Yoel Tenne
Chi-Keong Goh (Eds.)

ADAPTATION, LEARNING, AND OPTIMIZATION Volume 2

Computational Intelligence in Expensive Optimization Problems

Springer

Adaptation, Learning, and Optimization, Volume 2

Series Editor-in-Chief

Meng-Hiot Lim
Nanyang Technological University, Singapore
E-mail: emhlim@ntu.edu.sg

Yew-Soon Ong
Nanyang Technological University, Singapore
E-mail: asysong@ntu.edu.sg

Further volumes of this series can be found on our homepage: springer.com

Vol. 1. Jingqiao Zhang and Arthur C. Sanderson
Adaptive Differential Evolution, 2009
ISBN 978-3-642-01526-7

Vol. 2. Yoel Tenne and Chi-Keong Goh (Eds.)
Computational Intelligence in
Expensive Optimization Problems, 2010
ISBN 978-3-642-10700-9

Yoel Tenne and Chi-Keong Goh (Eds.)

Computational Intelligence in Expensive Optimization Problems

Springer

Dr. Yoel Tenne
Department of Mechanical Engineering and
Science-Faculty of Engineering,
Kyoto University, Yoshida-honmachi, Sakyo-Ku,
Kyoto 606-8501, Japan
E-mail: yoel.tenne@ky3.ecs.kyoto-u.ac.jp
Formerly: School of Aerospace Mechanical and Mechatronic Engineering,
Sydney University, NSW 2006, Australia

Dr. Chi-Keong Goh
Advanced Technology Centre,
Rolls-Royce Singapore Pte Ltd
50 Nanyang Avenue, Block N2,
Level B3C, Unit 05-08, Singapore 639798
E-mail: chi.keong.goh@rolls-royce.com

ISBN 978-3-642-10700-9 e-ISBN 978-3-642-10701-6

DOI 10.1007/978-3-642-10701-6

Adaptation, Learning, and Optimization ISSN 1867-4534

Library of Congress Control Number: 2009940123

Typeset & *Cover Design:* Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

To our families for their love and support.

Preface

Optimization is an essential part of research, both in science and in engineering. In many cases the research goal is an outcome of an optimization problem, for example, improving a vehicle's aerodynamics or a metal alloy's tensile strength.

Motivated by industrial demands, the process of design in science and engineering has undergone a major transformation. The advances in the fields of intelligent computing paradigm and the introduction of massive computing power have facilitated a move away from paper-based analytical systems towards digital models and computer simulations. Computer-aided design optimization is now involved in a wide range of design applications, ranging from large transatlantic airplanes to micro electro mechanical systems.

With the development of more powerful optimization techniques, the research community is continually seeking new optimization challenges and to solve increasingly more complicated problems. An emerging class of such challenging problems is known as the 'expensive optimization problems'. High computational cost can arise due to:

- Resource-intensive evaluations of the objective function: such problems arise when using 'computer-experiments', i.e., when a computer simulation replaces a real-world laboratory experiment during the optimization process. Such simulations can be prohibitory expensive (require anywhere from minutes to hours of evaluation time for each candidate solution). Also, there is no analytic expression for the objective function or its derivatives, requiring optimization algorithms which are derivative-free. Examples include wing shape optimization and electronic circuit design.
- Very high dimensional problems: in problems with hundreds or thousands of variables the 'curse of dimensionality' implies locating an optimum can be intractable due to the size of the search space. Examples include scheduling problems and image analysis.

On top of these difficulties, real-world optimization problems may exhibit additional challenges such as a complicated and non-smooth landscape,

multiple optima and discontinuities. Under these difficulties classical optimization methods may perform poorly or may even fail to obtain a satisfactory solution within the allocated resources (such as computer time). To circumvent this, researchers turn to computational intelligence methods such as agent-based algorithms, fuzzy logic and artificial neural networks. Such methods have shown to perform well in challenging scenarios and they can often handle a wide variety of problems when little or no-apriori knowledge is available. These nature- and biologically-inspired techniques are capable of 'learning' the problem features during the optimization and this can improve their performance and provide a better final solution.

However, the application of computational intelligence methods to expensive optimization problems is not straightforward. Their robustness, also referred to as the 'exploration-exploitation trade-off', implies they do not exploit domain knowledge efficiently and this can impair their convergence. For example, an evolutionary algorithm may require many thousands of function evaluations to obtain a satisfactory solution, which is unacceptable when each function evaluation requires hours of computer run-time. This necessitates the need to explore various methods to bridge the missing gaps before computational intelligence can be applied effectively to expensive problems.

Computational intelligence in Expensive Optimization Problems is a recent and emerging field which has received increasing attention in the last decade. This edited book represents the first endeavor to provide a snapshot of the current state-of-the-art in the field, covering both theory and practice. This edition consists of chapters contributed by leading researchers in the field, demonstrating the different methodology and practice to handle high computational cost of today's applications. This book is intended for wide readership and can be read by engineers, researchers, senior undergraduates and graduates who are interested in the development of computational intelligence techniques for expensive optimization problems.

This book is divided into 3 parts:

I Techniques for resource-intensive problems
II Techniques for high-dimensional problems
III Real-world applications

Part I considers the various methods to reduce the evaluation time, such as using models (also known as surrogate-models or meta-models, which are computationally cheaper approximations of the true expensive function) and parallelization. This section starts with two surveys on the current state-of-the-art. Shi and Rasheed survey a wide range of model-assisted algorithms, including frameworks for model-management in single objective optimization while Santana-Quintero *et al.* survey fitness approximations in multi-objective algorithms. Giannakoglou and Kampolis propose a flexible parallel multilevel evolutionary algorithm (EA) framework where each level can employ a different model, different search algorithm or different parametrization. They describe the performance of their approach with real-world expensive

aerodynamic shape optimization problems. Koziel and Bandler describe another approach which uses models of different fidelity, the 'space-mapping' method, to accelerate the optimization search. They apply their method to electronic circuit design. In another related study, Takahama and Sakai propose methods for model management which assesses the model accuracy and decides when a model needs to be improved. They implement their method in a differential evolution framework. Ginsbourger *et al.* parallelize the Efficient Global Optimization (EGO) algorithm which uses Kriging models and the expected improvement criterion. They propose statistical criteria for selecting multiple sites to evaluate for each iteration. Guimarães *et al.* propose a memetic algorithm for expensive design optimization problems. Their algorithm identifies promising regions and candidates from these regions are identified with a higher fidelity model and are given more weight by the algorithm. Ochoa also employs statistical criteria and proposes using Estimation of Distribution Algorithms (EDAs) to reduce the number of function evaluations. The study describes several approaches such as Boltzmann estimation and the Shrinkage EDAs. Also within the evolutionary computing framework, Fonseca *et al.* explore the use of similarity-based models (a nearest-neighbour approach) to extend the number of generations of an evolutionary algorithm in expensive optimization problems. Nakayama *et al.* and Bird and Li address the issues of expensive dynamic optimization problems. Nakayama *et al.* describe a model-predictive control algorithm for dynamic and expensive multiobjective optimization problems where they use a support-vector regression model. On the other hand, Bird and Li suggest a specialized particle swarm optimization (PSO) algorithm with least-squares regressors. The regressors locally approximate the objective function landscape and accelerate the convergence of the PSO to local optima.

In Part II, researchers explore sophisticated operators, such as those utilizing domain knowledge or which self-adapt during the search to combat the 'curse of dimensionality'. Caponio *et al.* implement a memetic algorithm which combines differential evolution (DE) with an adaptive local search which scales the DE vector, along with other algorithmic enhancements. Carvalho and Ferreira tackle the electric network distribution problem, which is a large scale combinatorial problem. They propose several hybrid Lamarckian evolutionary algorithms with specialized operators. dos Santos *et al.* tackle the traveling salesman problem (TSP) and propose a reinforcement learning metaheuristic for a specialized parallel hybrid EA. They show performance can be improved by using multiple search trajectories. Süral *et al.* also focus on the TSP and the TSP with back hauls problem and propose several evolutionary algorithms with specialized crossover and mutation and operators. They show that utilizing domain knowledge improves the algorithms performance. Cococcioni *et al.* study multiobjective genetic Takagi-Sugeno fuzzy systems in high-dimensional problems, which pose a challenge to such multiobjective EAs. They propose two enhancements to the multiobjective EA to accelerate the search. Davis-Moradkhan and Browne propose a specialized

evolutionary algorithm to tackle the multicriterion minimum spanning tree problem, a challenging combinatorial problem. They suggest several specialized operators as well as several algorithm variants to improve the spread of solutions along the Pareto front. Lastly, Shilane *et al.* present a specialized evolutionary algorithm to tackle the problem of risk-minimization in statistical parameter estimation, a multimodal high-dimensional problem. They demonstrate that their algorithm compares well with existing parameter-estimation methods while having the advantage that it can run in parallel.

Part III focuses on real-world applications. Successful application of computational intelligence methods to real-world problems is non-trivial and there are important insights and lessons to be learned from researchers' experience. Chen *et al.* use a particle swarm optimization algorithm for an expensive optimization problem of a transceiver design. They study a semi-blind joint maximum likelihood channel estimation and data detection for a receiver and the minimum bit-error-rate multiuser transmission. Results show their algorithm outperforms existing approaches. Donateo describes a multiobjective optimization of a diesel engine piston. The study used a multiobjective evolutionary algorithm which is parallelized over a cluster to reduce evaluation time. They obtained a more efficient engine with lower pollution level. Vasile and Croisard study the robust planning of a space mission, where the computational time grows exponentially with the number of uncertain variables. They use a multiobjective EA and apply the Evidence theory and an indirect approach to estimate the belief and plausibility functions. Kumar and Bauer propose a methodology to manage an expensive design process from the conceptual stage to a final design. They apply the methodology to the design of electrical drives and electrical circuits. Won *et al.* consider the problem of reliable network design and proposed a hybrid EA-ant colony system algorithm. They propose a multiring encoding scheme to combine the two and apply their algorithm to a variety of network design problems. Yamada and Berger describe the optimization of neural network for speech recognition using an EA. The structure of the EA changes from a random search to a steady state EA and finally to an elitist EA during the optimization. The algorithm reduces the high computational cost of the optimization by identifying a promising subset of variables and concentrating on it. Guichón and Castro tackle the expensive optimization problem of automatic image registration optimization by using a parallel evolutionary algorithm. The study describes an implementation of a fast and robust Internet subtraction service using a distributed evolutionary algorithm and a service-oriented architecture. Finally, Pilato *et al.* describe an expensive multiobjective optimization digital circuits, where the proposed algorithm uses fitness inheritance and approximation models to reduce the number of calls to the expensive simulation.

Overall, the chapters in this volume discuss a wide range of topics which reflect the broad spectrum of computational intelligence in expensive optimization problems. The chapters highlight both the current achievements and challenges and point to promising future research venues in this exciting field.

September 2009

Yoel Tenne
Chi-Keong Goh

Acknowledgement To Reviewers

We express our thanks to the expertise provided by our fellow researchers who have kindly reviewed for the edited book. Their assistance have been invaluable to our endeavors.

B. V. Babu
Will Browne
Pedro M. S. Carvalho
Jia Chen
Sheng Chen
Tsung-Che Chiang
Siang-Yew Chong
Antonio Della Ciopa
Carlos A. Coello Coello
Marco Cococcioni
Claudio De Stefano
Antonio Gaspar-Cunha
Kyriakos C. Giannakoglou
David Ginsbourger
Frederico Guimarães
Martin Holena
Amitay Issacs
Sameena S. Jayadeva
Wee Tat Koo
Slawomier Koziel
Jouni Lampinen
Xiaodong Li
Dudy Lim
Passi Luukka
Pramod Kumar Meher
Hirotaka Nakayama
Ferrante Neri
Thai Dung Nguyen
Alberto Ochoa
Yew-Soon Ong
Khaled Rasheed
Tapabrata Ray
Abdellah Salhi
Vui Ann Shim
Ofer M. Shir
Dimitri Solomatine
Sanjay Srivastava
Janusz Starzyk
Stephan Stilkerich
Haldun Süral
Mohammhed B. Trabia
Massimiliano Vasile
Lingfeng Wang
Chee How Wong

Contents

Part I
Techniques for Resource-Intensive Problems

Chapter 1
A Survey of Fitness Approximation Methods Applied in Evolutionary Algorithms

L. Shi and K. Rasheed

Abstract. Evolutionary algorithms (EAs) used in complex optimization domains usually need to perform a large number of fitness function evaluations in order to get near-optimal solutions. In real world application domains such as engineering design problems, such evaluations can be extremely computationally expensive. In some extreme cases there is no clear definition of the fitness function or the fitness function is too ambiguous to be deterministically evaluated. It is therefore common to estimate or approximate the fitness. A popular method is to construct a so-called surrogate or meta-model, which can simulate the behavior of the original fitness function, but can be evaluated much faster. An interesting trend is to use multiple surrogates to gain better performance in fitness approximation. In this chapter, an up-to-date survey of fitness approximation applied in evolutionary algorithms is presented. The main focus areas are the methods of fitness approximation, the working styles of fitness approximation, and the management of the approximation during the optimization process. To conclude, some open questions in this area are discussed.

1.1 Introduction

In recent years, EAs have been applied to many real-world application domains and gained much research interest. EAs proved to be powerful tools for optimization problems and were therefore used in a wide range of real-world applications, especially for engineering design domains. In such domains, the so-called fitness functions are sometimes discontinuous, non-differential, with many local optima, noisy

L. Shi
Applied Research, McAfee
e-mail: liang_shi@mcafee.com

K. Rasheed
Computer Science Department, University of Georgia
e-mail: Khaled@uga.edu

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 3–28.
springerlink.com

and ambiguous. It was found that EAs perform better than the conventional optimizers such as sequential quadratic programming and Simulated Annealing [2, 3, 4, 73].

Many challenges still arise in the application of EAs to real-world domains. For engineering design problems, a large number of objective evaluations may be required in order to obtain near-optimal solutions. Moreover, the search space can be complex, with many constraints and a small feasible region. However, determining the fitness of each point may involve the use of a simulator or analysis code that takes an extremely long time to execute. Therefore it would be difficult to be cavalier about the number of objective evaluations used for an optimization [5, 6]. For tasks like art design and music composition, no explicit fitness function exists; experienced human users are needed to do the evaluation. A human's ability to deal with a large number of evaluations is limited as humans easily get tired. Another challenge is that the environment of an EA can be noisy, which means that the exact fitness cannot be determined, and an approximate fitness must be assigned to each individual. An average fitness solution to the noise problem requires even more evaluations. For such problems surrogate-assisted evolution methods based on fitness approximation are preferable, as they can simulate the exact fitness at a much lower computational cost. A good fitness approximation method can still lead the EA process to find optimal or near-optimal solutions and is also tolerant to noise [7, 71].

In this chapter we further extend the discussion about fitness approximation by introducing more concepts in this area and by presenting new developments in recent years. Three main aspects of fitness approximation are our main focus areas. Those are the different types of fitness approximation methods, the working styles and the management schemes of the fitness approximation.

For the methods of fitness approximation, instance-based learning methods, machine learning methods and statistical learning methods are the most popular ones. Instance-based and machine learning methods include fitness inheritance, radial basis function models, the K-nearest-neighbor method, clustering techniques, and neural network methods. Statistical learning methods also known as functional models such as the polynomial models, the Kriging models, and the support vector machines are all widely used for fitness approximation in EAs. Comparative studies among these methods are presented in this chapter.

For the working styles of the fitness approximation, we discuss both direct and indirect fitness replacement strategies. The direct fitness replacement method is to use the approximate fitness to directly replace the original exact fitness during the course of the EA process. Thus individuals mostly have the approximate fitness during the optimization. The indirect fitness replacement method is to use the approximate fitness only for some but not all processes in the EA, such as population initialization and EA operators. Individuals have the exact fitness during most if not all of the optimization process.

With fitness approximation in EAs, the quality of an approximate model is always a concern for lack of training data and the often high dimensionality of the problem. Obtaining a perfect approximate model is not possible in such cases. Usually the original fitness function is used with the approximate method to solve this problem. The original fitness function can either correct some/all

individuals' fitness in some generations or improve the approximate model by giving the exact fitness. This is called the management of the fitness approximation or evolution control. In this chapter, different management methods of approximate fitness are presented, including online fitness update, offline model training, online model update, hierarchical models, and model migration. At the end of this charter, two real-world expensive optimizations by surrogate-assisted EAs are given.

1.2 Fitness Approximation Methods

The optimization problem usually deals with non-linear functions. There are some classic approximation methods, such as transforming the original function to a simpler one. This method entails transforming the original functions to linear ones, and then using a linear programming technique, such as the Frank-Wolfe method [8] or Powell's quadratic approximation [9]. Other classical methods like the Fourier approximation and Walsh function approximation use a set of basis functions and find a weighted sum of such basis functions to use as an approximation. These techniques have been used to first transform the original problem into an easy one, and then apply the EA to find the optima of the easier version of the original fitness function [10, 11].

Another class of methods determines a function's approximation using a chosen set of evaluated points extracted from the whole design space or from the evaluation history. This class includes in-stance-based learning methods (also known as lazy learning methods), machine learning methods and statistical learning methods. The relationships between the different types of fitness approximation methods are shown in Fig. 1.1. Many instancebased learning and other machine learning methods have been used for fitness approximation in EAs. Several of the most popular of these methods are reviewed next.

1.2.1 Instance-Based Learning Methods

1.2.1.1 Fitness Inheritance (FI)

Fitness inheritance techniques are one of the main subclasses of fitness approximation techniques. One such technique simply assigns the fitness of a new solution (child) based on the average fitness of its parents or a weighted average based on how similar the child is to each parent [12]. To deal with a noisy fitness function, a resampling method combined with a simple average fitness inheritance method is used to reduce the computational cost in [15]. Another approach is to divide the population into building blocks according to certain schemata. Under this approach, an individual obtains its fitness from the average fitness of all the members in its building block [13]. More sophisticated methods such as conditional probability tables and decision trees are used in [14] for fitness inheritance.

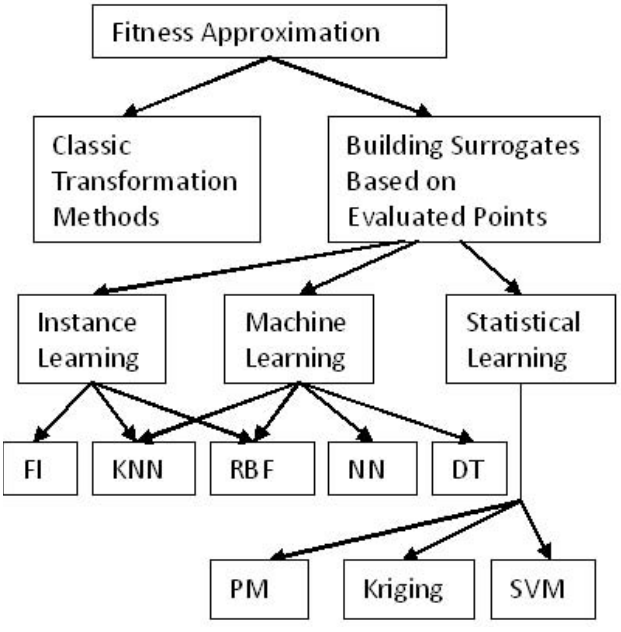


Fig. 1.1 Fitness approximation methods, FI: Fitness Inheritance KNN: K-Nearest Neighbors RBF: Radial Basis Functions NN: Neural Networks DT: Decision Tree PM: Polynomial Model SVM: Support Vector Machines

1.2.1.2 Radial Basis Function Model (RBF)

The Radial Basis Function (RBF) model is another instance-based learning method. RBF networks can also be viewed as a type of neural networks. Since it is a very popular technique for fitness approximation in EAs [3, 19, 31], it is worthy of being introduced independently from the normal multilayer neural networks.

An RBF network consists of an input layer with the same number of input units as the problem dimension, a single hidden layer of k nonlinear processing units and an output layer of linear weights w_I (Fig. 1.2). The size of the hidden layer (k) can be equal to the sample size if the sample size is small. In the case of a larger sample size, k is usually smaller than the sample size to avoid excessive calculations. This RBF network is called the generalized RBF network. The output $y(x)$ of the RBF network is given as a linear combination of a set of radial basis functions expressed in the following way:

$$y(x) = w_0 + \sum_{i=1}^{k} w_i \phi_i (\|x - c_i\|) \tag{1.1}$$

where w_0 and w_i are the unknown coefficients to be learned. The term $\phi_i (\|x - c_i\|)$, also called the kernel, represents the ith radial basis function. It evaluates the distance between the input x and the center c_i. For the generalized RBF network, the

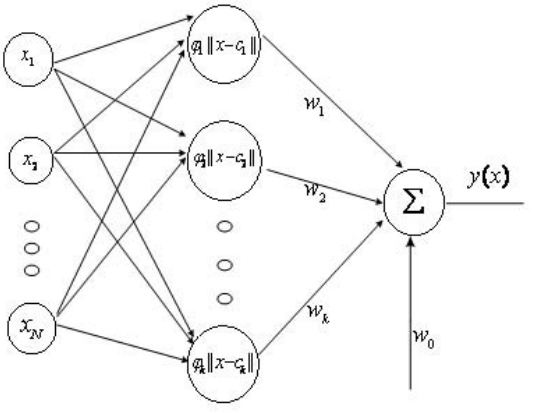


Fig. 1.2 Structure of RBF network models

centers c_i are also unknown and have to be learned by other methods such as the k-means method.

Typical choices for the kernel include linear splines, cubic splines, multi-quadratics, thin-plate splines, and Gaussian kernels. A Gaussian kernel is the most commonly used in practice, having the form:

$$\phi_i(\|x - c_i\|) = \exp\left(-\frac{\|x - c_i\|}{2\sigma^2}\right) \tag{1.2}$$

A detailed comprehensive description of RBF networks can be found in [32].

1.2.2 Machine Learning Methods

1.2.2.1 Clustering Techniques

Clustering algorithms include hierarchical clustering (such as single-linkage, complete-linkage, average-linkage and Ward's method), partition clustering (such as Hard C-Means and K-Means algorithm), and overlapping clustering (such as Fuzzy C-Means and B-Clump algorithm). Among them, the K-Means algorithm appears to be the most popular one for application to EAs due to its relative simplicity and low computational cost. In [16, 17], the entire population is divided into many clusters, and only the center of each cluster is evaluated. Other individuals' fitness values in the clusters are computed using their distance from these centers. Another approach is to build an approximate model based on sample points composed of the cluster centers. Every other individual's fitness is estimated by this approximate model, which may be a neural network model [18] or an RBF model [19]. Another interesting clustering approach applied in EAs is to divide the population into

several clusters and then build an approximate model for each cluster. The motivation is that multiple approximate models are believed to utilize more local information about the search space and fit the original fitness function better than a single model [5, 20, 21].

1.2.2.2 Multilayer Perceptron Neural Networks (MLPNNs)

Multilayer Perceptron Neural Networks (MLPNNs) usually utilize the back-propagation algorithm. MLPNNs have been proven to be powerful tools for fitness approximation. A MLPNN model is generally used to accelerate the convergence by replacing the original fitness function [34, 36]. In engineering design domains and drug design, MLPNNs have been used to reduce the evaluation times of complex fitness functions [35, 37, 38]. In [68], MLPNNs are used as surrogates to speed-up the process of an expensive blade design problem.

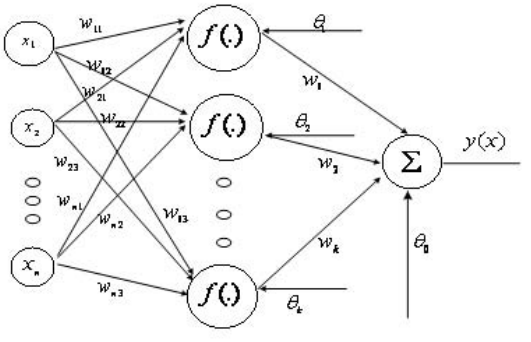


Fig. 1.3 Structure of the feed-forward MLPNN model

A simple feed-forward MLPNN with one input layer, one hidden layer and one output layer can be expressed as:

$$y(x) = \sum_{j=1}^{K} w_j f\left(\sum_{i=1}^{n} w_{ij} x_i + \theta_j\right) + \theta_0 \tag{1.3}$$

where n is the number of input neurons (which is usually equal to the problem dimension), K is the number of nodes of the hidden layer, and the function f is called the activation function. The structure of a feed-forward MLPNN is shown in Fig. 1.3. W and θ are the unknown weights to be learned. The most commonly used activation function is the logistic function, which has the form:

$$f(x) = \frac{1}{1 + \exp(-cx)} \tag{1.4}$$

where c is a constant. A comprehensive study can be found in [32].

1.2.2.3 Other Machine Learning Techniques

Other machine learning techniques were also applied for fitness approximation in EAs. An individual's fitness can be estimated by its neighbors using the K-nearest-neighbor algorithm [22]. The screening technique has been used for pre-selection [2, 5]. Decision Tree (DT) is another machine learning technique which has been used in [14].

1.2.3 Statistical Learning Methods

Statistical Learning methods for fitness approximation (basically statistical learning models) as applied to EAs have gained much interest among researchers, and have been used in several successful GA packages. In these methods, single or multiple models are built during the optimization process to approximate the original fitness function. These models are also referred to as approximate models, surrogates or meta-models. Among these models, Polynomial Models, Kriging Models, and Support Vector Machines (SVM) are the most commonly used.

1.2.3.1 Polynomial Models

Polynomial models (PM) are sometimes called Response Surfaces. Commonly used quadratic polynomial models have the form:

$$\hat{F}(\overline{X}) = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1,j=1}^{n,n} a_{ij} x_i x_j \tag{1.5}$$

Where a_0, a_i and a_{ij} are the coefficients to be fitted, n is the dimension of the problem and x_i is design variable number i.

Usually the least-squares approximation method is used to fit the unknown coefficients a_0, a_i and a_{ij}. The main limitation of the least-squares method is that the number of sample points (N) must exceed (n+1)(n+2)/2 for a second-order polynomial model. Even if this condition is satisfied, the fitting cannot be guaranteed because the singularity problem may still arise. Another drawback of the least-squares method is that its computational complexity grows quickly with the problem's dimension which can be unacceptable. The gradient method is introduced to address the problems of the least-squares method. More implementation details for using these two methods to fit a polynomial model can be found in [23]. Polynomial models are widely used as surrogates. One application can be found in [69], which uses a response surface method to approximate the Pareto front in NSGA-II for an expensive liquid-rocket engine design problem.

1.2.3.2 Kriging Models

The Kriging model consists of two component models which can be mathematically expressed as:

$$y(x) = f(x) + Z(x) \tag{1.6}$$

Where $f(x)$ represents a global model and $Z(x)$ is the realization of a stationary Gaussian random function that creates a localized deviation from the global model. Typically $f(x)$ is a polynomial and can be as simple as an underlying constant β in many cases, and then equation (1.6) becomes:

$$y(x) = \beta + Z(x) \tag{1.7}$$

The estimated model of equation (1.7) is given as:

$$\hat{y} = \hat{\beta} + r^T(x)R^{-1}(y - f\hat{\beta}) \tag{1.8}$$

Where y is a vector of length N as defined in equation (1.8), $\hat{y}$ is the estimated value of y given the current input x, f is a column vector which is filled with ones, and R is the correlation matrix which can be obtained by computing the correlation function between any two sampled data points. The form of the correlation function is specified by the user. Gaussian exponential correlation functions are commonly used, which is why the Kriging model is also sometimes called a Gaussian process:

$$R(x^i, x^j) = \exp\left[-\sum_{k=1}^{n} \theta_k \left| x_k^i - x_k^j \right|^2\right] \tag{1.9}$$

The correlation vector between x and the sampled data points is expressed as:

$$r^T(x) = \left[R(x, x^1), R(x, x^2), \ldots, R(x, x^n)\right]^T \tag{1.10}$$

Estimation of the parameters is often carried out using the generalized least squares method or the maximum likelihood method. Detailed implementations can be found in [24, 25].

In addition to the approximate values, the Kriging method can also provide accuracy information about the fitting in the form of confidence intervals for the estimated values without additional computational cost. In [6, 28], a Kriging model is used to build the global models because it is believed to be a good solution for fitting complex surfaces. A Kriging model is used to pre-select the most promising solutions in [29]. In [26, 27, 30], a Kriging model is used to accelerate the optimization or reduce the expensive computational cost of the original fitness function. In [67], a Kriging model with a pattern search technique is used to approximate the original expensive function. In [70], a Gaussian process method is used for landscape search in a multi-objective optimization problem that gives promising performance. One disadvantage of the Kriging method is that it is sensitive to the problem's dimension. The computational cost is unacceptable when the dimension of the problem is high.

1.2.3.3 Support Vector Machines (SVM)

The SVM model is primarily a classifier that performs classification tasks by constructing hyper-planes in a multidimensional space to separate cases with different

class labels. Contemporary SVM models support both regression and classification tasks and can handle multiple continuous and categorical variables. A detailed description of SVM models can be found in [40, 41]. SVM models compare favorably to many other approximation models because they are not sensitive to local optima, their optimization process does not depend on the problem dimensions, and overfitting is seldom an issue. Applications of SVM for fitness approximation can be found in [42]. The regression SVM is used for constructing approximate models. There are two types of regression SVMs: epsilon-SVM regression and nu-SVM regression. The epsilon-SVM regression model is more commonly used for fitness approximation, where the linear epsilon-insensitive loss function is defined by:

$$L^{\varepsilon}(x,y,f) = |y - f(x)| = \max(0, |y - f(x)| - \varepsilon) \tag{1.11}$$

The sum of the linear epsilon-insensitive losses must be minimized:

$$\frac{1}{2} w^T w + C \sum_{i=1}^{N} L^{\varepsilon}(x_i, y_i, f) \tag{1.12}$$

This is equivalent to a constrained minimization problem having the form:

$$\frac{1}{2} w^T w + C \sum_{i=1}^{N} \zeta_i + C \sum_{i=1}^{N} \zeta_i^* \tag{1.13}$$

Subject to the following constraints:

$$w^T \phi(x_i) + b - y_i \leq \varepsilon + \zeta_i^* \tag{1.14}$$

$$y_i - w^T \phi(x_i) - b_i \leq \varepsilon + \zeta_i \tag{1.15}$$

$$\zeta_i, \zeta_i^* \geq 0, i = 1, \ldots, N \tag{1.16}$$

Where $\phi(x)$ is called the kernel function. It may have the forms of linear, polynomial, Gaussian, RBF and sigmoid functions. The RBF is by far the most popular choice of kernel type used in SVMs. This is mainly because of their localized and finite responses across the entire range of the real x-axis. This optimization problem can be solved by using quadratic programming techniques.

1.2.4 Existing Research in Multi-surrogate Assisted EAs

For some real world applications, special approximation methods have been used. For example, a one dimensional approximation of the Kubelka Munk model is used to replace the expensive Monte Carlo method in an EA for analyzing colon tissue structure [55]. In [58], a classifier with confidence information is evolved to replace time consuming evaluations during tournament selection.

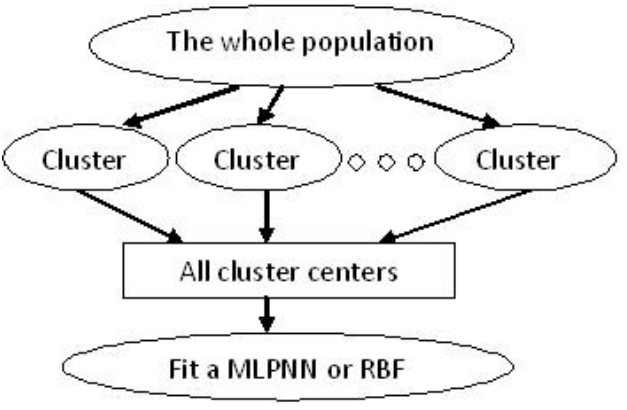


Fig. 1.4 A multiple surrogate model structure used in [18, 33]

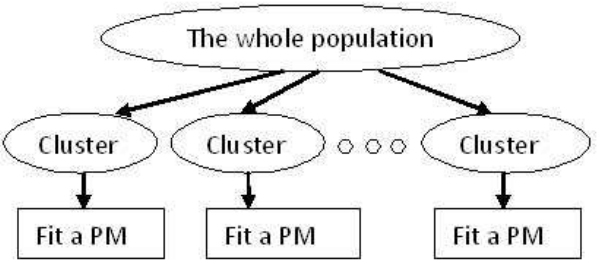


Fig. 1.5 A multiple surrogate model structure used in [5]

In some applications, several approximation methods have been combined to construct a type of fitness approximation model known as a Multi-surrogate. In [18, 33] the MLPNN model was combined with clustering methods for constructing approximate models (shown in Fig. 1.4). Fig. 1.5 shows another strategy using clustering techniques and polynomial models together [5]. A trained RBF model was used to generate sample points for the construction of polynomial models for fitness approximation in [39]. In [28, 51], the Kriging method was used to construct a global approximate model for pre-selection then RBF models were built using those pre-selected sample points for further fitness approximation. Fig. 1.6 shows the structure of this model. Multiple approximate models formed in a hierarchical structure have been used to assist the fitness evaluations in 1.11. In [59, 62], multiple local approximate models are built for each individual, and then these local models are aggregated into an average or weighted average of all approximate models. In [64, 65], multiple surrogates are built, and then the best surrogate is used [64] or the weighted sum of all surrogates is used, where the weights associated with each surrogate are determined based on the accuracy of the surrogate [65]. Fig. 1.7 shows this model in detail.

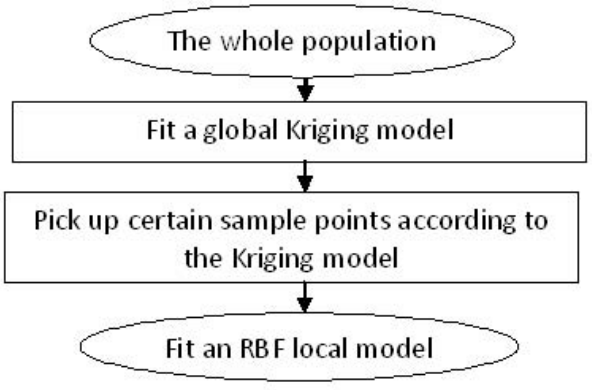


Fig. 1.6 A multiple surrogate model structure used in [51]

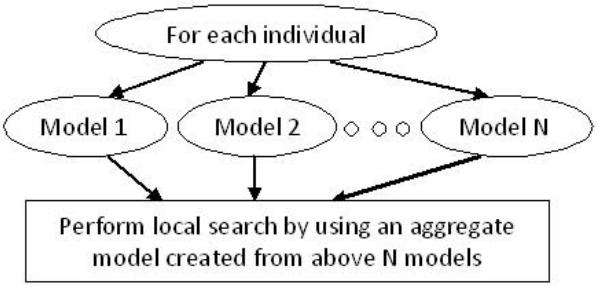


Fig. 1.7 A multiple surrogate model structure used in [59]

Multi-surrogates are also used for multi-objective optimization problems. In [63], the NSGA-II algorithm, a multi-objective EA using PM and RBF surrogates together is presented. A local surrogate-assisted evolution strategy using KNN and RBF models is introduced in [61]. For each new offspring in this strategy, a cubic RBF surrogate is built using the k-nearest previously evaluated points. This local RBF surrogate is then used to estimate the new off-spring's fitness. Fig. 1.8 shows the model structure used in [61]. In [66], Polynomial Models are used to estimate the coefficients of the fitness surrogate. Thus the surrogate is made adaptive to characteristics of the specific optimization problems.

A recent trend is to use multiple approximate models adaptively. In [60], both global and local (for each cluster) surrogate models are used. The global model adaptively evolves from a simple average of the fitness of all individuals in a population, all the way to a Support Vector Machine (SVM) model. The local models follow a similar path but do not exceed quadratic polynomials. The model evolution depends on the time complexity as well as model accuracy for each model. A

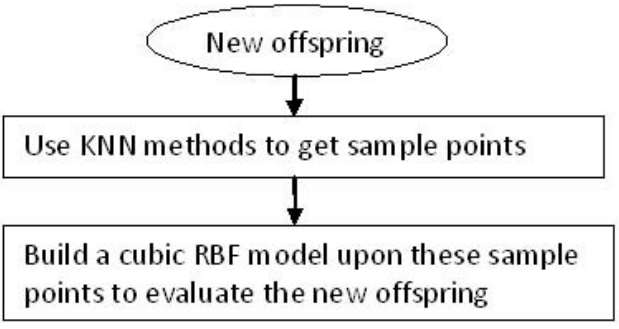


Fig. 1.8 A Multiple surrogates structure used in [61]

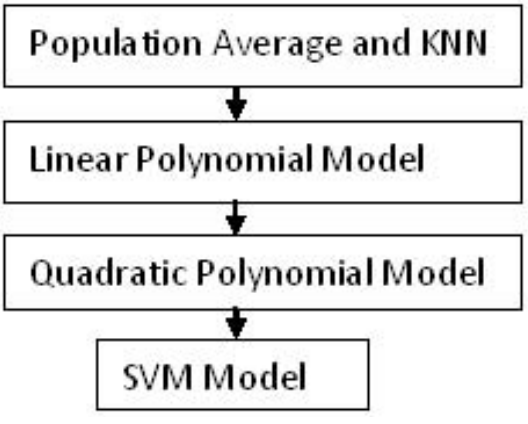


Fig. 1.9 Global approximate model evolution path in [60]

formula is used to decide whether to continue using the same type of model or switch to the next at any time. Fig. 1.9 shows the evolution path.

1.3 Comparative Studies for Different Approximate Models

Many approximation methods have been introduced for special problem domains. Even though these methods are claimed to save many function evaluations and to be nearly as good as the original fitness function, they are bound to their special domains, and thus no comparative studies have been conducted on them. On the other hand, the performance of many general-purpose approximation methods has been compared in early papers, especially for popular methods such as statistical learning methods.

The neural network model and the polynomial model were compared in [46, 72]. The study concluded that the performance of the two types of approximation was comparable in terms of the number of function evaluations required to build the approximations and the number of undetermined parameters associated with the approximations. However, the polynomial model had a much lower construction cost. In [72], after evaluating both methods in several applications, the authors concluded that both of them can perform comparably for modest data. In [43], a quadratic polynomial model was found to be the best method among the polynomial model, RBF network, and the Quick-prop neural network when the models were built for regions created by clustering techniques. The authors were in favor of the polynomial model because they found that it formed approximations more than an order of magnitude faster than the other methods and did not require any tuning of parameters. The authors also pointed out that the polynomial approximation was in a mathematical form which could be algebraically analyzed and manipulated, as opposed to the black-box results that neural networks give.

The Kriging model and the neural network model were compared using benchmark problems in [47]. However, no clear conclusion was drawn about which model is better. Instead, the author showed that optimization with a meta-model could lead to degraded performance. Another comparison was presented in [45] between the polynomial model and the Kriging model. By testing these two models on a real-world engineering design problem, the author found that the polynomial and Kriging approximations yielded comparable results with minimal difference in predictive capability. Comparisons between several approximate models were presented in [44], which compared the performance of the polynomial model, the multivariate adaptive splines' model, the RBF model, and the Kriging model using 14 test problems with different scales and nonlinearities. Their conclusion was that the polynomial model is the best for low-order nonlinear problems, and the RBF model is the best for dealing with high-order nonlinear problems (details shown in Table 1.1). In [59], four types of approximate models - Gaussian Process (Kriging), RBF, Polynomial model and Extreme Learning Machine Neural Network (ELMNN) - were compared on artificial unconstrained benchmark domains. Polynomial Models (PM) were found to be the best for final solution quality and RBF was found to be the best when considering correlation coefficients between the exact fitness and estimated fitness. Table 1.2 shows the performance ranks of these four models in terms of the quality of the final solution.

So far different approximate models have been compared based on their performance, but the word performance itself has not been clearly defined. This is because the definition of performance may depend on the problem to be addressed, and multiple criteria need to be considered. Model accuracy is probably the most important criterion, since approximate models with a low accuracy may lead the optimization process to local optima. Model accuracy also should be based on new sample points instead of the training data set points. The reason for this is that for some models such as the neural network, overfitting is a common problem. In the case of overfitting, the model works very well on training data, yielding good model accuracy,

Table 1.1 Summary of best methods in [44]

	Low-order Nonlinearity	High-order Nonlinearity
Small Scale	Polynomial	RBF
Large Scale	Kriging	RBF
Overall	Polynomial	RBF

Table 1.2 Final quality measures for Kriging, PM. RBF and ELMNN approximate models in [59]

Benchmark domain	Method			
	Kriging	PM	RBF	ELMNN
Ackley	2	1	4	3
Griewank	3	1	2	4
Rosenbrock	1	3	2	4
Step	3	1	2	4

but may perform poorly on new sample points. The optimization process could easily go in the wrong direction if it is assisted by a model suffering from overfitting. There are other important criteria to be considered, including robustness, efficiency, and time spent on model construction and updating. A fair comparison would consider the model accuracy as well as all of these criteria.

It is difficult to draw a clear conclusion on which model is the best for the reasons stated above, though the polynomial model seems to be the best choice for a local model when dealing with local regions or clusters and enough sample points are available [43]. In such cases, the fitting problem usually has low-order nonlinearity and the polynomial model is the best candidate according to [44]. The polynomial model is also believed to perform the best for problems with noise [44]. As for high-order nonlinear problems the RBF model is believed to be the best and it is the least sensitive to the sample size and has the most robustness [44]. So the RBF model is a good choice for a global model with or without many samples. In [72], NN is found to perform significantly better than PM when search space is very complex and the parameters are correctly set.

The SVM model is a powerful fitting tool that belongs to the class of kernel methods. Because of the beneficial features of SVMs stated above, the SVM model becomes a good choice for constructing a global model, especially for problems with high dimension and many local optima, provided that a large sample of points exists.

1.4 The Working Styles of Fitness Approximation

There are two categories of surrogate incorporation mechanisms in EAs, as shown in Fig. 1.10. In one category the original fitness is directly replaced by the estimated fitness when the individual is evaluated throughout the optimization. Only a few individuals have their exact fitness calculated for control purposes. In the other category, the original fitness is kept for each individual and the approximate fitness is not used to directly replace the original fitness. These two methods are reviewed next.

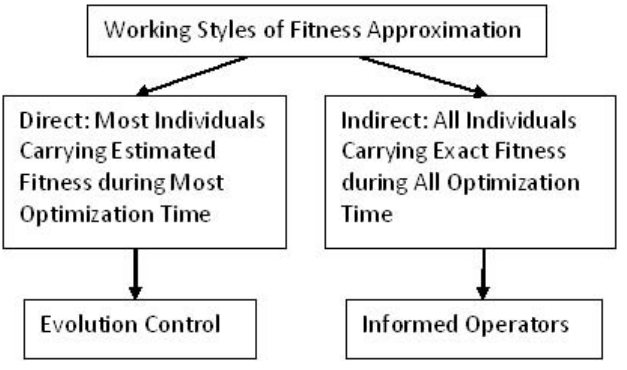


Fig. 1.10 Working styles of fitness approximation

1.4.1 Direct Fitness Replacement Methods

Direct fitness replacement is straightforward. Individuals are evaluated by surrogates and then the estimated fitness is assigned to each individual. During the course of the EA process, the approximate fitness assumes the role of the original fitness. This method has been used in numerous research efforts [6, 12, 13, 14, 15, 16, 18, 19, 22, 26, 27, 28, 29, 30, 31, 34, 35, 36, 37, 42, 48, 56]. The obvious draw-back is that the inaccuracy of the approximate fitness may lead the EA to inferior local optima. Consequently, the direct fitness replacement method needs a continuous calibration process called Evolution Control (described below). Even with Evolution Control, convergence to true optima cannot be guaranteed.

1.4.2 Indirect Fitness Approximation Methods

The indirect surrogate method computes the exact fitness for each individual during an EA process and the approximate fitness is used in other ways. For example, the approximate fitness can be used for population pre-selection. In this method, instead of generating a random initial population, an individual for the initial population

can be generated by selecting the best individual from a number of uniformly distributed random individuals in the design space according to the approximate fitness [5, 43, 49].

Approximate fitness can also be used for crossover or mutation in a similar manner, through a technique known as Informed Operators [5, 17, 43, 49]. Under this approach, the approximate models are used to evaluate candidates only during the crossover and/or mutation process. After the crossover and/or mutation process, the exact fitness is still computed for the newly created candidate solutions. Using the approximate fitness indirectly in the form of Informed Operators - rather than direct evaluation - is expected to keep the optimization moving toward the true global optima and to reduce the risk of convergence to suboptimal solutions because each individual in the population is still assigned its exact fitness [49]. Experimental results have shown that a surrogate-assisted informed operator-based multi objective GA can outperform state-of-art multi objective GAs for several benchmark problems [5]. Informed Operators also make it easy to use surrogates adaptively, as the number of candidates can be adaptively determined. Some of the informed operators used in [49] are explained as follows:

- Informed initialization: Approximate fitness is used for population pre-selection. Instead of generating a random initial population, an individual for the initial population can be generated by selecting the best individuals from a number of uniformly distributed random individuals in the design space according to the approximate fitness.
- Informed mutation: To perform informed mutation, several random mutations of the base point are generated. The mutation with the best approximate fitness value is returned as the result.
- Informed crossover: Two parents are selected at random according to the usual selection strategy. These two parents are not changed in the course of the informed crossover operation. Several crossovers are conducted by randomly selecting a crossover method, randomly selecting its internal parameters and applying it to the two parents to generate a potential child. The surrogate is used to evaluate every potential child, and the best child is selected as the outcome.

1.5 The Management of Fitness Approximation

For direct fitness replacement methods the management of the fitness approximation is necessary to drive the EA to converge to global optima with the cost reduced as much as possible. There are several ways to conduct the model management, as shown in Fig. 1.11.

1.5.1 Evolution Control

Evolution Control uses surrogates together with original fitness functions in an EA process where the original fitness functions are used to evaluate some/all

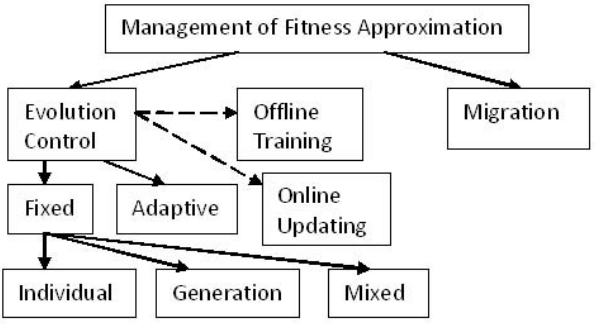


Fig. 1.11 Management of fitness approximation

individuals in some/all generations. There are two categories of Evolution Control methods: Fixed Evolution Control and Adaptive Evolution Control. For fixed evolution control, there are individual-based and generation-based methods. In individual-based evolution control, only some selected individuals are evaluated by the exact fitness function. The individual selection can be random or using some strategy, e.g., selecting the best individual (according to the surrogate) for evolution control. In generation-based Evolution Control, all individuals in a selected generation will be evaluated by the original fitness function, the generation selection can be random or with a fixed frequency. The adaptive Evolution Control adjusts the frequency of control according to the fidelity of the surrogates.

1.5.2 Offline Model Training

Offline model training constructs surrogates based on human evaluation or previous optimization history data. In this case, either the approximate model is of high fidelity or the original fitness cannot be easily evaluated during an EA process such as evolutionary art, so the original fitness is never used. An example of this method can be found in [50].

1.5.3 Online Model Updating

Fitness approximation may be constructed at an early stage of the EA process. Because of the limited sample points, a surrogate may concentrate on the region spanned by the existing sample points and not cover the rest of the search space well. As the EA continues and new individuals enter into the population, the accuracy of the previously built surrogate model will decrease. Thus the surrogate model needs to be reformed using the old sample points together with the new sample points. This technique is known as online surrogate up-dating. There has been considerable research with this method [3, 5, 6, 16, 19, 20, 26, 27, 28, 29, 30, 31].

1.5.4 Hierarchical Approximate Models and Model Migration

The hierarchical surrogates' method builds multiple models with a hierarchical structure during the course of an EA process [51, 52]. In [51], a Gaussian process model is built for the so-called global model. A user-specified percentage of the best individuals according to the global model are selected to form a local search space. Then Lamarckian evolution is performed involving a trust region-enabled gradient-based search strategy that employs RBF local approximate models to accelerate convergence in the local search space. In [52] the whole population is divided into several sub-populations. Each sub-population constructs its own surrogate. At a certain interval, the individuals in the different sub-populations can migrate into other sub-populations. This is called an Island Model. To gain a balance between the model performance and the population diversity, the selection of migration size and migration interval is important. It has been found that the migration interval plays a more dominant role than migration size [58].

1.6 Case Studies: Two Surrogate-Assisted EA Real-World Applications

1.6.1 The Welded Beam Design Domain

The Welded Beam Design problem is illustrated in Fig. 1.12. The goal for this design is to use the least material to sustain a certain weight. It has four design variables $\mathbf{x} = (h, l, t, b)$; the definition of Welded Beam Design problem can be found in the appendix. The known optimal solution is 2.38116 with $h = 0.2444$, $l = 6.2187$, $t = 8.2915$, and $b = 0.2444$.

We demonstrate it with three GA methods, GADO, GADO-R and ASAGA. GADO [2] stands for Genetic Algorithm for Design Optimization, a GA that has proved to

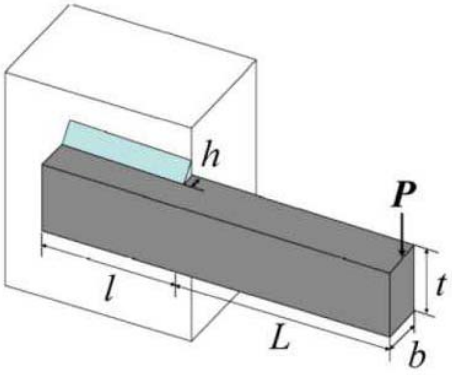


Fig. 1.12 Welded Beam Structure

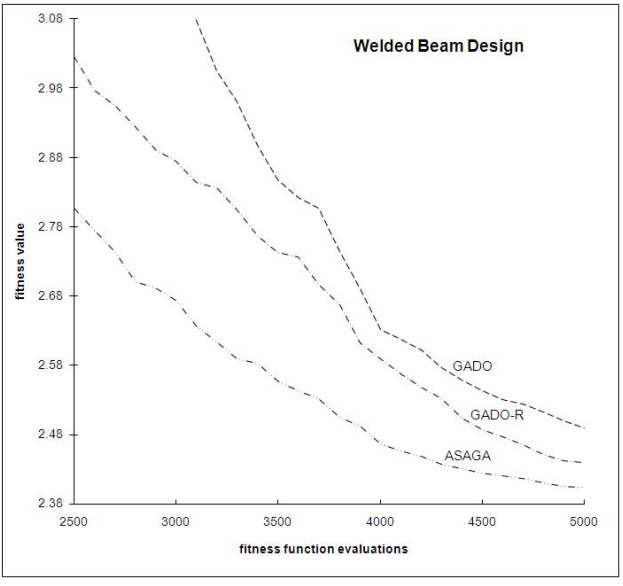


Fig. 1.13 Welded Beam design with global optima 2.38116

be powerful for solving engineering design problems. GADO-R is based on GADO, and includes global and local polynomial surrogate models structured by clustering techniques. ASAGA [60] is an adaptive multi-surrogate assisted EA with a backbone of a GADO. GADO was used with no approximate model assistance. GADO-R incorporates fixed quadratic polynomial surrogates through Informed Operators [45, 59]. All three methods ran 30 times with different random starting populations. The average best fitness values of the 30 runs with corresponding number of actual fitness evaluations are shown in Fig. 1.13. The figure shows that the surrogate-assisted GA outperforms the GA with no surrogate assistance and the adaptive surrogate-assisted GA further outperforms the non-adaptive surrogate-assisted GA.

1.6.2 Supersonic Aircraft Design Domain

This domain concerns the conceptual design of supersonic transport aircraft. It is summarized briefly here and is described in more detail in [74]. Fig. 1.14 shows a diagram of a typical airplane automatically designed by the software system. The GA attempts to find a good design for a particular mission by varying twelve of the aircraft conceptual design parameters over a continuous range of values. An optimizer evaluates candidate designs using a multidisciplinary simulator. The optimizer's goal is to minimize the takeoff mass of the aircraft, a measure of merit commonly used in the aircraft industry at the conceptual design stage. Takeoff mass is the sum of fuel mass, which provides a rough approximation of the operating cost

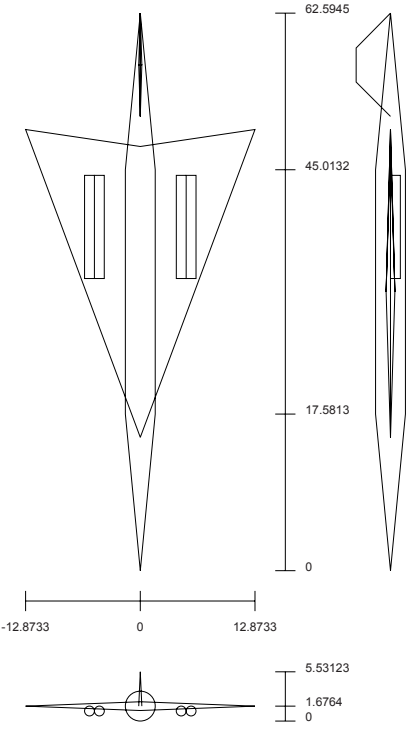


Fig. 1.14 Supersonic aircraft design problem

of the aircraft, and "dry"mass, which provides a rough approximation of the cost of building the aircraft. In summary, the problem has 12 parameters and 37 inequality constraints and only 0.6% of the search space is evaluable.

Fig. 1.15 shows a performance comparison in this domain. Each curve in the figure shows the average of 15 runs of GADO starting from random initial populations. The experiments were done once for each surrogate: Least Square PM (LS), Quick-Prop NN (QP) and RBF in addition to one without the surrogate-assisted informed operators altogether, with all other parameters kept the same. Fig. 1.15 demonstrates the performance with each of the three surrogate-assisted methods as well as performance with no approximation at all (the solid line). The figure plots the average (over the 15 runs) of the best measure of merit found so far in the optimization as a function of the number of iterations. The figure shows that all surrogate-assisted methods are better than the plain GADO and the LS approximation method gave the best performance in all stages of the search in this domain.

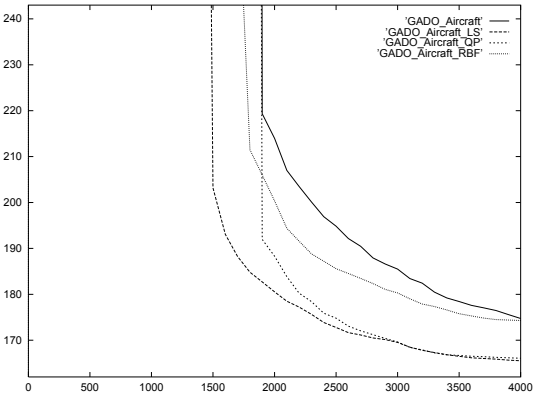


Fig. 1.15 Four GA methods comparison in supersonic aircraft design domain, landscape shows numbers of fitness function evaluations and vertical direction shows fitness values

1.7 Final Remarks

Using fitness approximation methods to assist GAs and other Evolutionary Algorithms has gained increasing popularity in recent years. This chapter presented a survey of the popular and recent trends in approximation methods, control strategies and management approaches. An interesting question in this area is: what is the best model for fitness approximation? Though the answer depends on the problem and user requirements, we propose an interesting generic solution, which is to try the simplest model first. If the performance is not satisfactory or degrades with time, more sophisticated models can be used. So far many researchers use only one type of approximation model, as in [26]. Some researchers use multiple models for different levels of approximation, but the approximate model itself is still fixed [18, 33, 51]. An interesting new direction [60] is to use an adaptive modeling method. In this method, first a simple approximation can be used such as the fitness inheritance or K-nearest neighbor method. If the fitting is not satisfactory, a more sophisticated model can be used such as the polynomial model. There are several levels inside the polynomial model method itself from the linear polynomial model to cubic polynomial model. They can be applied following a simple to complex direction. If this level of approximation is still inadequate, more complex models should be introduced such as the RBF, Kriging or SVM models. Usually the more complex models give better fitting accuracy but need more construction time. This adaptive method can provide the best trade-off between model performance and efficiency by adaptively adjusting the fitness approximation.

Appendix: Definitions of Welded Beam Design

$$\text{Minimize} f_{wielded_beam}(\mathbf{x}) = 1.10471h^2l + 0.04811tb(14+l) \quad (1.17)$$

$$\text{Subject to} 13600 - \tau(\mathbf{x}) \geq 0 \quad (1.18)$$

$$30000 - \sigma(\mathbf{x}) \geq 0 \quad (1.19)$$

$$b - h \geq 0 \quad (1.20)$$

$$P_c(\mathbf{x}) - 6000 \geq 0 \quad (1.21)$$

$$0.25 - \delta(\mathbf{x}) \geq 0 \quad (1.22)$$

$$0.125 \leq h \leq 10 \quad (1.23)$$

$$0.1 \leq l,t,b \leq 10 \quad (1.24)$$

The terms $\tau(\mathbf{x})$, $\sigma(\mathbf{x})$, and $\delta(\mathbf{x})$ are given below:

$$\tau(\mathbf{x}) = \sqrt{\tau'(\mathbf{x})^2 + \tau''(\mathbf{x})^2 + l\tau'(\mathbf{x})\tau''(\mathbf{x})/\sqrt{0.24(l^2+(h+t)^2)}} \quad (1.25)$$

$$\sigma(\mathbf{x}) = 504000/(t^2b) \quad (1.26)$$

$$P_c(\mathbf{x}) = 64746.022(1-0.0282346t)tb^3 \quad (1.27)$$

$$\delta(\mathbf{x}) = 2.1952/(t^3b) \quad (1.28)$$

where

$$\tau'(\mathbf{x}) = 6000/(\sqrt{2}hl) \quad (1.29)$$

$$\tau''(\mathbf{x}) = \frac{6000(14+0.5l)\sqrt{0.25(l^w+(h+t)^2)}}{2(0.707hl(l^2/12+0.25(h+t)^2))} \quad (1.30)$$

References

[1] Abuthinien, M., Chen, S., Hanzo, L.: Semi-blind joint maximum likelihood channel estimation and data detection for MIMO systems. IEEE Signal Processing Letters 15, 202–205 (2008)

[2] Rasheed, K.: GADO: A genetic algorithm for continuous design optimization. Technical Report DCS-TR-352, Department of Computer Science, Rutgers University. Ph.D. Thesis (1998)

[3] Ong, Y.S., Nair, P.B., Keane, A.J., Wong, K.W.: Surrogate-Assisted Evolutionary Optimization Frameworks for High-Fidelity Engineering Design Problems. In: Jin, Y. (ed.) Knowledge Incorporation in Evolutionary Computation. Studies in Fuzziness and Soft Computing, pp. 307–332. Springer, Heidelberg (2004)

[4] Schwefel, H.-P.: Evolution and Optimum Seeking. Wiley, Chichester (1995)

[5] Chafekar, D., Shi, L., Rasheed, K., Xuan, J.: Multi-objective GA optimization using reduced models. IEEE Trans. on Systems, Man, and Cybernetics: Part C 9(2), 261–265 (2005)

[6] Chung, H.-S., Alonso, J.J.: Multi-objective optimization using approximation model-based genetic algorithms. Technical report 2004-4325, AIAA (2004)

[7] Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing Journal 9(1), 3–12 (2005)
[8] Reklaitis, G.V., Ravindran, A., Ragsdell, K.M.: Engineering Optimization Methods and Application. Wiley, New York (1983)
[9] Deb, K.: Optimization for Engineering Design: Algorithms and Examples. Prentice-Hall, New Delhi (1995)
[10] Weinberger, E.D.: Fourier and Taylor series on fitness landscapes. Biological Cybernetics 65(55), 321–330 (1991)
[11] Hordijk, W., Stadler, P.F.: Amplitude Spectra of Fitness Landscapes. J. Complex Systems 1, 39–66 (1998)
[12] Smith, R., Dike, B., Stegmann, S.: Fitness inheritance in genetic algorithms. In: Proceedings of ACM Symposiums on Applied Computing, pp. 345–350. ACM, New York (1995)
[13] Sastry, K., Goldberg, D.E., Pelikan, M.: Don't evaluate, inherit. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 551–558. Morgan Kaufmann, San Francisco (2001)
[14] Pelikan, M., Sastry, K.: Fitness Inheritance in the Bayesian Optimization Algorithm. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 48–59. Springer, Heidelberg (2004)
[15] Bui, L.T., Abbass, H.A., Essam, D.: Fitness inheritance for noisy evolutionary multi-objective optimization. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 779–785 (2005)
[16] Kim, H.-S., Cho, S.-B.: An efficient genetic algorithm with less fitness evaluation by clustering. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 887–894. IEEE, Los Alamitos (2001)
[17] Elliott, L., Ingham, D.B., Kyne, A.G., Mera, N.S., Pourkashanian, M., Wilson, C.W.: An informed operator based genetic algorithm for tuning the reaction rate parameters of chemical kinetics mechanisms. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 945–956. Springer, Heidelberg (2004)
[18] Jin, Y., Sendhoff, B.: Reducing Fitness Evaluations Using Clustering Techniques and Neural Network Ensembles. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 688–699. Springer, Heidelberg (2004)
[19] Ong, Y.S., Keane, A.J., Nair, P.B.: Surrogate-Assisted Coevolutionary Search. In: 9th International Conference on Neural Information Processing, Special Session on Trends in Global Optimization, Singapore, pp. 2195–2199 (2002)
[20] Rasheed, K.: An incremental-approximate-clustering approach for developing dynamic reduced models for design optimization. In: Proceedings of the Congress on Evolutionary Computation (CEC 2002), pp. 986–993 (2000)
[21] Pelikan, M., Sastry, K., Goldberg, D.E.: Multiobjective hBOA, clustering, and scalability. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, Washington DC, USA, pp. 663–670 (2005)
[22] Takagi, H.: Interactive evolutionary computation. Fusion of the capabilities of EC optimization and human evaluation. Proceedings of the IEEE 89(9), 1275–1296 (2001)
[23] Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C: the Art of Scientific Computing, 2nd edn. Cambridge University Press, Cambridge (1992)
[24] Gibbs, M., MacKay, D.J.C.: Efficient Implementation of Gaussian Processes. Cavendish Laboratory, Cambridge (1997) (unpublished manuscript)

[25] Williams, C.K.I., Rasmussen, C.E.: Gaussian Processes for regression. In: Touretzky, D.S., Mozer, M.C., Hasselmo, M.E. (eds.) Advances in Neural Information Processing Systems, vol. 8. MIT Press, Cambridge (1996)
[26] Emmerich, M., Giotis, A., Özdemir, M., Bäck, T., Giannakoglou, K.: Metamodel-assisted evolution strategies. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 361–380. Springer, Heidelberg (2002)
[27] El-Beltagy, M.A., Keane, A.J.: Evolutionary optimization for computationally expensive problems using Gaussian processes. In: Proceedings of International Conference on Artificial Intelligence, pp. 708–714. CSREA (2001)
[28] Zhou, Z., Ong, Y.S., Nair, P.B.: Hierarchical surrogate-assisted evolutionary optimization framework. In: Congress on Evolutionary Computation, pp. 1586–1593. IEEE, Los Alamitos (2004)
[29] Ulmer, H., Streichert, F., Zell, A.: Evolution startegies assisted by Gaussian processes with improved pre-selection criterion. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 692–699 (2003)
[30] Bueche, D., Schraudolph, N.N., Koumoutsakos, P.: Accelerating evolutionary algorithms with Gaussian process fitness function models. IEEE Trans. on Systems, Man, and Cybernetics: Part C 35(2), 183–194 (2005)
[31] Ulmer, H., Streicher, F., Zell, A.: Model-assisted steady-state evolution strategies. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 610–621. Springer, Heidelberg (2003)
[32] Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
[33] Graening, L., Jin, Y., Sendhoff, B.: Efficient evolutionary optimization using individual-based evolution control and neural networks: A comparative study. In: European Symposium on Artificial Neural Networks, pp. 273–278 (2005)
[34] Hong, Y.-S., Lee, H., Tahk, M.-J.: Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks. Engineering Optimization 35(1), 91–102 (2003)
[35] Hüscken, M., Jin, Y., Sendhoff, B.: Structure optimization of neural networks for aerodynamic optimization. Soft Computing Journal 9(1), 21–28 (2005)
[36] Jin, Y., Hüsken, M., Olhofer, M., Sendhoff, B.: Neural networks for fitness approximation in evolutionary optimization. In: Jin, Y. (ed.) Knowledge Incorporation in Evolutionary Computation, pp. 281–305. Springer, Berlin (2004)
[37] Papadrakakis, M., Lagaros, N., Tsompanakis, Y.: Optimization of large-scale 3D trusses using Evolution Strategies and Neural Networks. Int. J. Space Structures 14(3), 211–223 (1999)
[38] Schneider, G.: Neural networks are useful tools for drug design. Neural Networks 13, 15–16 (2000)
[39] Shyy, W., Tucker, P.K., Vaidyanathan, R.: Response surface and neural network techniques for rocket engine injector optimization. Technical report 99-2455, AIAA (1999)
[40] Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge Press (2000)
[41] Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Patter Analysis. Cambridge Press (2004)

[42] Llorà, X., Sastry, K., Goldberg, D.E., Gupta, A., Lakshmi, L.: Combating User Fatigue in iGAs: Partial Ordering, Support Vector Machines, and Synthetic Fitness. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 1363–1370 (2005)
[43] Rasheed, K., Ni, X., Vattam, S.: Comparison of Methods for Developing Dynamic Reduced Models for Design Optimization. Soft Computing Journal 9(1), 29–37 (2005)
[44] Jin, R., Chen, W., Simpson, T.W.: Comparative studies of metamodeling techniques under multiple modeling criteria. Technical report 2000-4801, AIAA (2000)
[45] Simpson, T., Mauery, T., Korte, J., Mistree, F.: Comparison of response surface and Kriging models for multidiscilinary design optimization. Technical report 98-4755, AIAA (1998)
[46] Carpenter, W., Barthelemy, J.-F.: A comparison of polynomial approximation and artificial neural nets as response surface. Technical report 92-2247, AIAA (1992)
[47] Willmes, L., Baeck, T., Jin, Y., Sendhoff, B.: Comparing neural networks and kriging for fitness approximation in evolutionary optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 663–670 (2003)
[48] Branke, J., Schmidt, C.: Fast convergence by means of fitness estimation. Soft Computing Journal 9(1), 13–20 (2005)
[49] Rasheed, K., Hirsh, H.: Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000), pp. 628–635 (2000)
[50] Biles, J.A.: GenJam: A genetic algorithm for generating jazz solos. In: Proceedings of International Computer Music Conference, pp. 131–137 (1994)
[51] Zhou, Z.Z., Ong, Y.S., Nair, P.B., Keane, A.J., Lum, K.Y.: Combining Global and Local Surrogate Models to Accelerate Evolutionary Optimization. IEEE Transactions on Systems, Man and Cybernetics - Part C 37(1), 66–76 (2007)
[52] Sefrioui, M., Periaux, J.: A hierarchical genetic algorithm using multiple models for optimization. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 879–888. Springer, Heidelberg (2000)
[53] Skolicki, Z., De Jong, K.: The influence of migration sizes and intervals on island models. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 1295–1302 (2005)
[54] Rasheed, K., Hirsh, H.: Learning to be selective in genetic-algorithm-based design optimization. Artificial Intelligence in Engineering, Design, Analysis and Manufacturing 13, 157–169 (1999)
[55] Hidović, D., Rowe, J.E.: Validating a model of colon colouration using an evolution strategy with adaptive approximations. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 1005–1016. Springer, Heidelberg (2004)
[56] Ziegler, J., Banzhaf, W.: Decreasing the number of evaluations in evolutionary algorithms by using a meta-model of the fitness function. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 264–275. Springer, Heidelberg (2003)
[57] Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments: A survey. IEEE Transactions on Evolutionary Computation 9(3), 303–317 (2005)
[58] Ziegler, J., Banzhaf, W.: Decreasing the number of evaluations in evolutionary algorithms by using a meta-model of the fitness function. In: Ryan, C., Soule, T., Keijzer, M., Tsang, E.P.K., Poli, R., Costa, E. (eds.) EuroGP 2003. LNCS, vol. 2610, pp. 264–275. Springer, Heidelberg (2003)

[59] Lim, D., Ong, Y.S., Jin, Y., Sendhoff, B.: A Study on Metamodeling Techniques, Ensembles, and Multi-Surrogates in Evolutionary Computation. In: Genetic and Evolutionary Computation Conference, London, UK, pp. 1288–1295. ACM Press, New York (2007)
[60] Shi, L., Rasheed, K.: ASAGA: An Adaptive Surrogate-Assisted Genetic Algorithm. In: Genetic and Evolutionary Computation Conference (GECCO 2008), pp. 1049–1056. ACM Press, New York (2008)
[61] Regis, R.G., Shoemaker, C.A.: Local Function Approximation in Evolutionary Algorithms for the Optimization of Costly Functions. IEEE Transactions on Evolutionary Computation 8(5), 490–505 (2004)
[62] Zerpa, L.E., Queipo, N.V., Pintos, S., Salager, J.-L.: An Optimization Methodology of Alkaline-surfactant-polymer Flooding Processes Using Field Scale Numerical Simulation and Multiple Surrogates. Journal of Petroleum Science and Engineering 47, 197–208 (2005)
[63] Lundström, D., Staffan, S., Shyy, W.: Hydraulic Turbine Diffuser Shape Optimization by Multiple Surrogate Model Approximations of Pareto Fronts. Journal of Fluids Engineering 129(9), 1228–1240 (2007)
[64] Zhou, Z., Ong, Y.S., Lim, M.H., Lee, B.S.: Memetic Algorithm Using Multi-surrogates for Computationally Expensive Optimization Problems. Soft Computing 11(10), 957–971 (2007)
[65] Goel, T., Haftka, R.T., Shyy, W., Queipo, N.V.: Ensemble of Surrogates? Structural and Multidisciplinary Optimization 33, 199–216 (2007)
[66] Sastry, K., Lima, C.F., Goldberg, D.E.: Evaluation Relaxation Using Substructural Information and Linear Estimation. In: Proceedings of the 8th annual conference on Genetic and Evolutionary Computation Conference (2006)
[67] Torczon, V., Trosset, M.: Using approximations to accelerate engineering design optimization. NASA/CR-1998-208460 (or ICASE Report No. 98-33) (1998)
[68] Pierret, S., Braembussche, R.A.V.: Turbomachinery Blade Design Using a Navier-Stokes Solver and ANN. Journal of Turbomachinery (ASME) 121(2) (1999)
[69] Goel, T., Vaidyanathan, R., Haftka, R.T., Shyy, W., Queipo, N.V., Tucker, K.: Response surface approximation of Pareto optimal front in multi-objective optimization. Computer Methods in Applied Mechanics and Engineering (2007)
[70] Knowles, J.: ParEGO: A Hybrid Algorithm with On-Line Landscape Approximation for Expensive Multiobjective Optimization Problems. IEEE Transactions on Evolutionary Computation 10(1) (February 2005)
[71] Giannakoglou, K.C.: Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. Progress in Aerospace Sciences 38(1) (2000)
[72] Shyy, W., Papila, N., Vaidyanathan, R., Tucker, K.: Global design optimization for aerodynamics and rocket propulsion components. Progress in Aerospace Sciences 37 (2001)
[73] Quagliarella, D., Periaux, J., Poloni, C., Winter, G. (eds.): Genetic Algorithms and Evolution Strategies in Engineering and Computer Science. Recent Advances and Industrial Applications, ch. 13, pp. 267–288. John Wiley and Sons, West Sussex (1997)
[74] Gelsey, A., Schwabacher, M., Smith, D.: Using modeling knowledge to guide design space search. In: Fourth International Conference on Artificial Intelligence in Design 1996 (1996)

Chapter 2
A Review of Techniques for Handling Expensive Functions in Evolutionary Multi-Objective Optimization

Luis V. Santana-Quintero, Alfredo Arias Montaño,
and Carlos A. Coello Coello*

Abstract. Evolutionary algorithms have been very popular for solving multi-objective optimization problems, mainly because of their ease of use, and their wide applicability. However, multi-objective evolutionary algorithms (MOEAs) tend to consume an important number of objective function evaluations, in order to achieve a reasonably good approximation of the Pareto front. This is a major concern when attempting to use MOEAs for real-world applications, since we can normally afford only a fairly limited number of fitness function evaluations in such cases. Despite these concerns, relatively few efforts have been reported in the literature to reduce the computational cost of MOEAs. It has been until relatively recently, that researchers have developed techniques to achieve an effective reduction of fitness function evaluations by exploiting knowledge acquired during the search. In this chapter, we analyze different proposals currently available in the specialized literature to deal with expensive functions in evolutionary multi-objective optimization. Additionally, we review some real-world applications of these methods, which can be seen as case studies in which such techniques led to a substantial reduction in the computational cost of the MOEA adopted. Finally, we also indicate some of the potential paths for future research in this area.

2.1 Introduction

In many disciplines, optimization problems have, in a natural form, two or more objectives that we aim to minimize simultaneously, and which are normally in conflict with each other. These problems are called "multi-objective", and their solution

Luis V. Santana-Quintero · Alfredo Arias Montaño · Carlos A. Coello Coello
CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D.F. 07360, Mexico
e-mail: lvspenny@hotmail.com, aarias@computacion.cs.cinvestav.mx, ccoello@cs.cinvestav.mx

* The third author is also affiliated to the UMI-LAFMIA 3175 CNRS.

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 29–59.
springerlink.com

gives rise not to one, but to a set of solutions representing the best possible trade-offs among the objectives (the so-called *Pareto optimal set*). In the absence of user's preferences, all the solutions contained in the Pareto optimal set are equally good. When plotted in objective function space, the contents of the Pareto optimal set produces the so-called *Pareto front*.

Evolutionary algorithms (EAs) have become a popular search engine for solving multi-objective optimization problems [17, 21], mainly because they are very easy to use and have a wide applicability. However, multi-objective evolutionary algorithms (MOEAs) normally require a significant number of objective function evaluations, in order to achieve a reasonably good approximation of the Pareto front, even when dealing with problems of low dimensionality. This is a major concern when attempting to use MOEAs for real-world applications, since in many of them, we can only afford a fairly limited number of fitness function evaluations.

Despite these concerns, relatively little efforts have been reported in the literature to reduce the computational cost of MOEAs, and several of them only focus on algorithmic complexity (see for example [36]), in which little else can be done because of the theoretical bounds related to nondominance checking [45].

It has been until relatively recently, that researchers have developed techniques to achieve a reduction of fitness function evaluations by exploiting knowledge acquired during the search [42]. Knowledge of past evaluations can also be used to build an empirical model that approximates the fitness function to optimize. This approximation can then be used to predict promising new solutions at a smaller evaluation cost than that of the original problem [40, 42]. Current functional approximation models include Polynomials (response surface methodologies [30, 65]), neural networks (e.g., multi-layer perceptrons (MLPs) [33, 34, 62]), radial-basis function (RBF) networks [60, 77, 83], support vector machines (SVMs) [4, 71], Gaussian processes [6, 78], and Kriging [24, 66] models. Other authors have adopted fitness inheritance [67] or cultural algorithms [46] for the same purposes.

In this chapter several possible schemes are described, in which the use of the knowledge from past solutions can help to guide the search of the new solutions, with particular emphasis on MOEAs. The remainder of this chapter is organized as follows. In Section 2.2, we present basic concepts related to multi-objective optimization. Then, in Section 2.3 we discuss several schemes that incorporate knowledge into the fitness evaluations of an evolutionary algorithm, providing a brief explanation of the surrogate models that have been used to approximate the fitness function. Next in Section 2.4 some selected research works are discussed. Such works are related to real–world engineering optimization problems, and can be considered as case studies in which the use of the described techniques led to a substantial reduction in the computational cost of the MOEA adopted. Finally, in Section 2.5, our conclusions and some potential paths for future research in this area are indicated.

2.2 Basic Concepts

The general multi-objective optimization problem (MOP) can be formally defined as the problem of finding:

$\mathbf{x}^* = (x_1^*, x_2^*, \ldots, x_n^*)^T$ which satisfies the m inequality constraints:

$$g_i(\overrightarrow{x}) \leq 0; \quad i = 1, \ldots, m$$

the p equality constraints:

$$h_j(\overrightarrow{x}) = 0; \quad j = 1, \ldots, p$$

and optimizes the vector function:

$$\mathbf{f}(\mathbf{x}) = [f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_k(\overrightarrow{x})]^T$$

In other words, we aim to determine from among the set S of all vectors (points) which satisfy the constraints those that yield the optimum values for all the k objective functions simultaneously. The constraints define the feasible region S and any point $\overrightarrow{x}$ in the feasible region is called a feasible point.

2.2.1 Pareto Dominance

Pareto dominance is formally defined as follows:

A vector $\overrightarrow{u} = (u_1, \ldots, u_k)$ is said to dominate a vector $\overrightarrow{v} = (v_1, \ldots, v_k)$ if and only if $\overrightarrow{u}$ is partially less than $\overrightarrow{v}$, i.e., $\forall i \in \{1, \ldots, k\}, u_i \leq v_i \wedge \exists i \in \{1, \ldots, k\} : u_i < v_i$ (assuming minimization).

In order to say that a solution dominates another one, it needs to be strictly better in at least one objective, and not worse in any of them. So, when we are comparing two different solutions A and B, there are 3 possible outcomes:

- A dominates B.
- A is dominated by B.
- A and B are incomparable.

2.2.2 Pareto Optimality

The formal definition of *Pareto optimality* is provided next:

A solution $\overrightarrow{x_u} \in S$ (where S is the feasible region) is said to be Pareto optimal if and only if there is no $\overrightarrow{x_v} \in S$ for which $\mathbf{v} = f(\mathbf{x}_v) = (v_1, \ldots, v_k)$ dominates $\mathbf{u} = f(\mathbf{x}_u) = (u_1, \ldots, u_k)$, where k is the number of objectives.

In words, this definition says that $\mathbf{x}_u$ is Pareto optimal if there exists no feasible vector $\mathbf{x_v}$ which would decrease some objective without causing a simultaneous increase in at least one other objective (assuming minimization).

This definition does not provide us a single solution (in decision variable space), but a set of solutions which form the so-called *Pareto Optimal Set* (P^*). The vectors that correspond to the solutions included in the Pareto optimal set are *nondominated*.

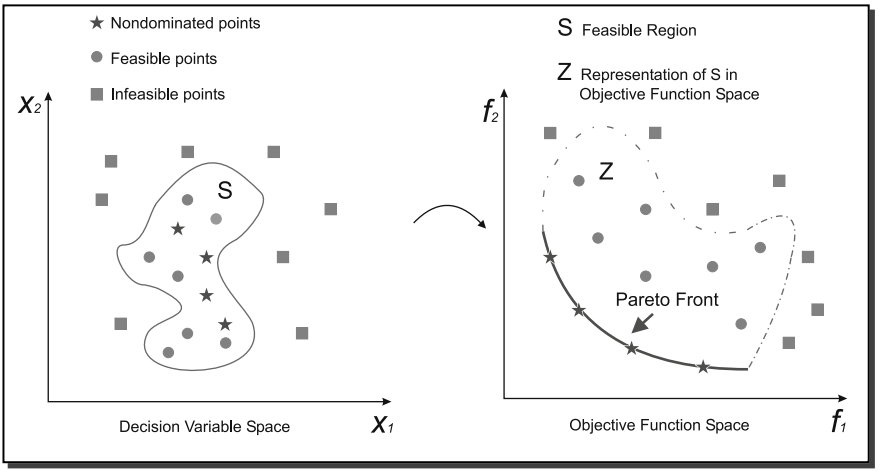


Fig. 2.1 Mapping of the Pareto optimal solutions to the objective function space

2.2.3 Pareto Front

When all nondominated solutions are plotted in objective function space, the nondominated vectors are collectively known as the *Pareto Front* (PF^*). Formally:

$$PF^* := \{\overrightarrow{f}(\mathbf{x}) = [f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})]^T \,|\mathbf{x} \in P^*\}$$

It is, in general, impossible to find an analytical expression that defines the Pareto front of a MOP, so the most common way to get the Pareto front is to compute a sufficient number of points in the feasible region, and then filter out the nondominated vectors from them.

The previous definitions are graphically depicted in Figure 2.1, showing the *Pareto front*, the *Pareto optimal set* and the *dominance* relations among solutions.

2.3 Knowledge Incorporation

From the many techniques adopted to solve such multi-objective optimization problems, evolutionary algorithms are among the most popular mainly because of their population-based nature, which is very useful to generate several nondominated solutions in a single run. However, dealing with a large population size and a large number of generations make MOEAs an unaffordable choice (computationally speaking) in certain applications, even when parallelism is adopted. In general, MOEAs can be unaffordable for an application when:

- The evaluation of the fitness functions is computationally expensive (i.e., it takes from minutes to hours).
- The fitness functions cannot be defined in an algebraic form (e.g., when the fitness functions are generated by a simulator).
- The total number of evaluations of the fitness functions is limited by financial constraints (i.e., there is a financial cost involved in computing the fitness functions).

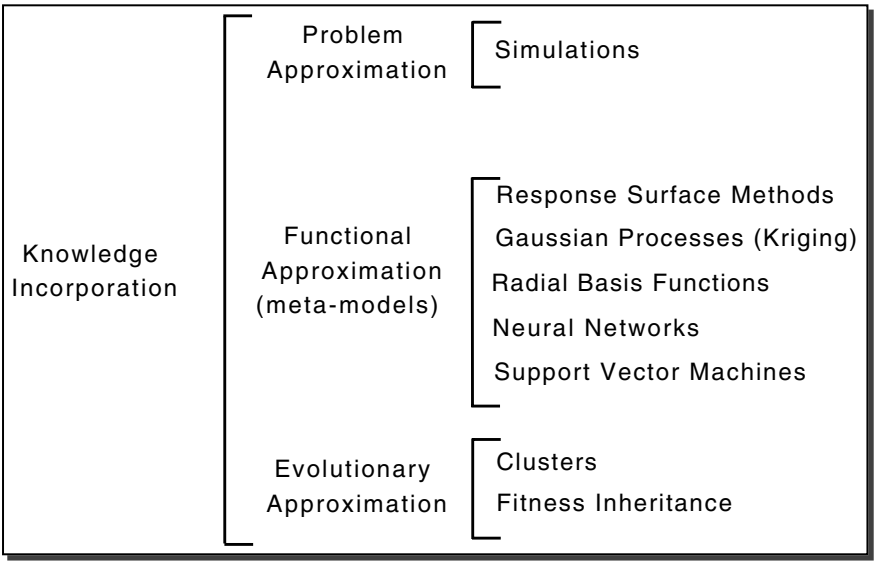


Fig. 2.2 A taxonomy of approaches for incorporating knowledge into evolutionary algorithms

Jin et al. [40] presented a taxonomy of approaches which incorporate knowledege into EAs (see Figure 2.2). From this taxonomy, we can distinguish three main types of strategies or approaches to deal with expensive fitness functions:

Problem Approximation: Tries to replace the original statement of the problem by one which is approximately the same as the original problem but which is easier to solve. To save the cost of the experiments, numerical simulations instead of physical experiments are used to pseudo-evaluate the performance of a design.

Functional Approximation: In this case, a new expression is constructed for the objective function based on previous data obtained from the real objective functions. The models obtained from the available data are often known as *meta-models* or *surrogates* (see Section 2.3.1).

Evolutionary Approximation: This approximation is specific for EAs and tries to save function evaluations by estimating an individual's fitness from other similar individuals. Two popular subclasses in this category are fitness inheritance and clustering.

2.3.1 Surrogates

In many practical engineering problems, we have black-box objective functions whose algebraic definitions are not known. In order to construct an approximation function, it is required to have a set of sample points that help us to build a meta-model of the problem. The objective of such meta-model is to reduce the total number of evaluations performed on the real objective functions, while maintaining

a reasonably good quality of the results obtained. Thus, such meta-model is used to predict promising new solutions at a smaller evaluation cost than that of the original problem.

The accuracy of the surrogate model relies on the number of samples provided in the search space, as well as on the selection of the appropriate model to represent the objective functions. There exist a variety of techniques for constructing surrogate models (see for example [79]). One example is least-square regression using low-order polynomials, also known as response surface methods. Comparisons of several surrogate modeling techniques have been presented by Giunta and Watson [27] and by Jin et al. [39].

A surrogate model is built when the objective functions are to be estimated. This local model is built using a set of data points that lie on the local neighborhood of the design. Since surrogate models will probably be built thousands of times during the search, computational efficiency becomes a major issue of their construction process.

In [43], Knowles and Nakayama present a survey of meta-modeling approaches to solve specific problems. The authors discuss the problem on how to model each objective function and how to improve the Pareto approximation set using a trade-off method proposed by Nakayama et al. [56]. In multi-objective optimization problems, the trade-off method tries to satisfy an aspiration level at the k-th iteration, with the help of a trade-off operator which changes the k-th level if the decision maker (DM) is not satisfied with the solution. So, they combine the satisficing trade-off method and meta-modeling for supporting the DM to get a final solution with a low number of fitness function evaluations. They use the $\mu - \nu$ Support Vector Regression method [57] as their meta-model and include two real-world multi-objective optimization problems, using also a Radial Basis Function Network with a Genetic Algorithm in searching the optimal value of the predicted objective function [58]. The proposed approach obtains good solutions within 1/10 or less analysis time than a conventional optimization approach based on a quasi-Newton method with approximated differentials.

2.3.2 *Polynomials: Response Surface Methods (RSM)*

The response surface methodology comprises three main components: (1) regression surface fitting, in order to obtain approximate responses, (2) design of experiments in order to obtain minimum variances of the responses and (3) optimizations using the approximated responses.

An advantage of this technique is that the fitness of the approximated response surfaces can be evaluated using powerful statistical tools. Additionally, the minimum variances of the response surfaces can be obtained using design of experiments with a small number of experiments.

For most response surfaces, the functions adopted for the approximations are polynomials because of their simplicity, although other types of functions are, of

course, possible. For the cases of quadratic polynomials, the response surface is described as follows:

$$\hat{y} = (\beta_0) + \sum_{i=1}^{n} (\beta_i \cdot x_i) + \sum_{i,j=1, i \leq j}^{n} (\beta_{i,j} \cdot x_i \cdot x_j) \tag{2.1}$$

where n is the number of variables, and β_0 and β_i are the coefficients to be calculated. To estimate the unknown coefficients of the polynomial model, both the least squares method (LSM) and the gradient method can be used, but either of them requires at least the same number of samples of the real objective function than the β_i coefficients in order to obtain good results.

2.3.3 Gaussian Process or Kriging

An alternative approach for constructing surrogate models is to use a Gaussian Process Model (also known as Kriging), which is also referred to as "Design and Analysis of Computer Experiments" (DACE) model [68] and Gaussian process regression [82]. This approach builds probability models through sample data and estimates the function values at every untested point with a Gaussian distribution.

In Kriging, the meta-model prediction is formed by adding up two different models as follows:

$$y(\overrightarrow{x}) = a(\overrightarrow{x}) + b(\overrightarrow{x})$$

where $a(\overrightarrow{x})$ represents the "average" long-term range behavior and the expected value of the true function. This function can be modeled in various ways, such as with polynomials or with trigonometric series as:

$$a(\overrightarrow{x}) = a_0 + \sum_{i=1}^{L} \sum_{j=1}^{R} a_{ij} (x_i)^j$$

where: R is the polynomial order with L dimensions and $b(\overrightarrow{x})$ stands for a local deviation term. $b(\overrightarrow{x})$ is a Gaussian random function with zero mean and non-zero covariance that represents a localized deviation from the global model. This function represents a short-distance influence of every data point over the global model. The general formulation for $b(\overrightarrow{x})$ is a weighted sum of N functions, $K_n(\mathbf{x})$ that represent the covariance functions between the n^{th} data point and any point $\mathbf{x}$:

$$b(\overrightarrow{x}) = \sum_{n=1}^{N} b_n K(h(x, x_n)) \text{ and } h(x, x_n) = \sqrt{\sum_{i=1}^{L} \left(\frac{x_i - x_{in}}{x_i^{max} - x_i^{min}} \right)^2}$$

where x_i^{min} and x_i^{max} are the lower an upper bounds of the search space and x_{in} denotes the $i-th$ component of the data point x_n. However, the shape of $K(h)$ has a strong influence on the resulting aspect of the statistical model. That is the reason why it is said that Kriging is used as an estimator or an interpolator.

Table 2.1 Radial basis functions

Type of Radial Function		
LS	linear splines	$\|r\|$
CS	cubic splines	$\|r\|^3$
MQS	multiquadrics splines	$\sqrt{1+(\varepsilon r)^2}$
TPS	thin plate splines	$\|r\|^{2m+1}\ln\|r\|$
GA	Gaussian	$e^{-(\varepsilon r)^2}$

2.3.4 Radial Basis Functions

Radial Basis Functions (RBFs) were first introduced by R. Hardy in 1971 [32]. Let's suppose we have certain points (called centers) $\overrightarrow{x}_1,\ldots,\overrightarrow{x}_n \in \mathbb{R}^d$. The linear combination of the function g centered at the points $\overrightarrow{x}$ is given by:

$$f:\mathbb{R}^d \mapsto \mathbb{R} : \overrightarrow{x} \mapsto \sum_{i=1}^{n} \lambda_i g(\overrightarrow{x}-\overrightarrow{x_i}) = \sum_{i=1}^{n} \lambda_i \phi(\|\overrightarrow{x}-\overrightarrow{x_i}\|) \tag{2.2}$$

where $\|\overrightarrow{x}-\overrightarrow{x_i}\|$ is the Euclidean distance between the points $\overrightarrow{x}$ and $\overrightarrow{x}_i$. So, f becomes a function which is in the finite dimensional space spanned by the basis functions:

$$g_i : \overrightarrow{x} \mapsto g(\|\overrightarrow{x}-\overrightarrow{x_i}\|)$$

Now, let's suppose that we already know the values of a certain function $H:\mathbb{R}^d \mapsto \mathbb{R}$ at a set of fixed locations $\overrightarrow{x_i},\ldots,\overrightarrow{x_n}$. These values are named $f_j = H(\overrightarrow{x_j})$, so we try to use the $\overrightarrow{x_j}$ as centers in the equation 2.2. If we want to force the function f to take the values f_j at the different points $\overrightarrow{x_j}$, then we have to put some conditions on the λ_i. This implies the following:

$$\forall j \in \{1,\ldots,n\}\ f_j = f(\overrightarrow{x_j}) = \sum_{i=1}^{n} (\lambda_i \cdot \phi(\|\overrightarrow{x_j}-\overrightarrow{x_i}\|))$$

In these equations, only the λ_i are unknown, and the equations are linear in their unknowns. Therefore, we can write these equations in matrix form:

$$\begin{bmatrix} \phi(0) & \phi(\|x_1-x_2\|) & \ldots & \phi(\|x_1-x_n\|) \\ \phi(\|x_2-x_1\|) & \phi(0) & \ldots & \phi(\|x_2-x_n\|) \\ \vdots & \vdots & & \vdots \\ \phi(\|x_n-x_1\|) & \phi(\|x_n-x_2\|) & \ldots & \phi(0) \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \tag{2.3}$$

Typical choices for the basis function $g(\mathbf{x})$ include linear splines, cubic splines, multiquadrics, thin-plate splines and Gaussian functions as shown in Table 2.1.

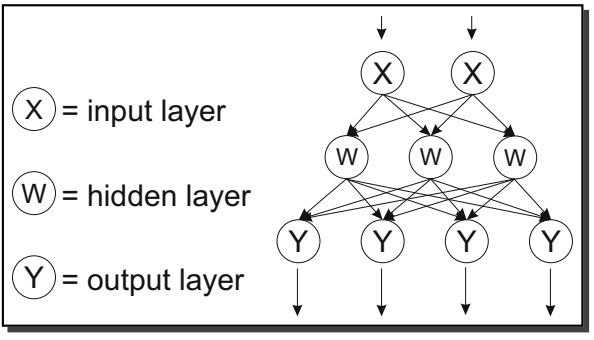


Fig. 2.3 A graphical representation of an MLP network with one hidden layer

2.3.5 Artificial Neural Networks

An ANN basically builds a map between a set of inputs and the corresponding outputs, and are good to deal with nonlinear regression analysis with noisy signals [5]. A multilayer feedforward neural network consists of an array of input nodes connected to an array of output nodes through successive intermediate layers. Each connection between nodes has a weight, which initially has a random value, and that is adjusted during a training process. The output of each node of a specific layer is a function of the sum on the weighted signals coming from the previous layer. The crucial points in the construction of an ANN are the selection of inputs and outputs, the architecture of the ANN, that is, the number of layers and the number of nodes in each layer, and finally, the training algorithm.

The multi-layer perceptron (MLP) is a multilayered feedforward network that has been widely used in function approximation problems, because it has been often found to provide compact representations of mappings in real-world problems. An MLP is composed of neurons and the output (y) of each neuron is thus:

$$y = \phi\left(\sum_{i=1}^{n} w_i \cdot a_i + b\right)$$

where a_i are the inputs of the neuron, and w_i is the weight associated with the i^{th} input. The nonlinear function ϕ is called the activation function as it determines the activation level of the neuron.

In Figure 2.3, we show an MLP network with one layer of linear output neurons and one layer of nonlinear neurons between the input and output neurons. The middle layers are usually called *hidden layers*.

To learn a mapping $\mathbb{R}^n \rightarrow \mathbb{R}^m$ by an MLP, its architecture should be the following: it should have n input nodes and m output nodes with a single or multiple hidden layer. The number of nodes in each hidden layer is generally a design decision.

2.3.5.1 Training an ANN

In general terms, supervised training consists of presenting to the network patterns whose output we know (the training set) finding the output of the net and adjusting the weights so as to make the actual output more like the desired (or teaching signal). The two most useful training protocols are: off-line and on-line. In off-line learning, all the data are stored and can be accessed repeatedly. In on-line learning, each case is discarded after it is processed and the weights are updated. With off-line learning, we can compute the objective function for any fixed set of weights, so we can see whether or not we are making progress in training.

Error backpropagation is the simplest and most widely used algorithm to train feedforward neural networks. In this algorithm the training is performed by minimizing a loss function, usually the sum of square errors over the N elements of the training set. In this case, it is adopted a generalization of the square error function given by:

$$J(W) = \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{c}(t_{ki} - z_{ki})^2 = \frac{1}{2}\sum_{i=1}^{N}||\overrightarrow{t_i} - \overrightarrow{z_i}||^2$$

where $\mathbf{t_i}$ and $\mathbf{z_i}$ are the i^{th}-target and the i^{th}-network output vectors of length c, respectively; W represents all the weights in the network. The backpropagation learning rule is based on a gradient descent. The weights are initialized with random values, and are changed in a direction to reduce the error following the next rule:

$$W_{new} = W_{old} - \eta\frac{\partial J}{\partial W}$$

The weight update for the hidden-output weights is given by:

$$\partial W_{kj} = \eta(t_k - z_k)f'(net_k)y_j$$

and the input-to-hidden weights learning rule is:

$$\partial W_{ji} = \eta \cdot x_i \cdot f'(net_j)\sum_{k=1}^{n} w_{kj}\partial_k$$

where η is the learning rate, i, j, k are the corresponding node indexes for each layer and net_j is the inner product of the input layer with the weights w_{ji} at the hidden unit.

2.3.6 Support Vector Machines

Support vector machines (SVM) have become popular in recent years for solving problems in classification, regression and novelty detection. An important property of support vector machines is that the determination of the model parameters corresponds to a convex optimization problem, and thus, any local solution found is also a global optimum. In *SVM* regression, our goal is to find a function $f(x)$ that has at most an ε deviation from the obtained targets y_i for all the training data, and

at the same time is as flat as possible. Let's suppose we are given training data $\chi = (x_t, y_t)_{t=1}^{N}$ where $y_t \in \mathbb{R}$. Then, the $f(x)$ is given by:

$$f(x) = \langle w, x \rangle + b \; with \; w \in \mathbb{R}^d, x \in \mathbb{R}^d, b \in \mathbb{R}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product in χ. A small w means that the regression is flat. One way to ensure this, is to minimize the norm, $||w||^2 = \langle w, w \rangle$. The problem can be written as a convex optimization problem:

$$\begin{array}{ll} minimize & \frac{1}{2}||w||^2 \\ subject\ to & \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon \end{cases} \end{array} \tag{2.4}$$

And one can introduce two slack variables ξ_i, ξ_i^*, for positive and negative deviations, $\xi_i \geq 0$ and $\xi_i^* \geq 0$, where $\xi_i > 0$ corresponds to a point for which $\langle w, x_i \rangle + b > y_i + \varepsilon$ and $\xi_i^* > 0$ corresponds to a point for which $\langle w, x_i \rangle + b < y_i - \varepsilon$ (as in Figure 2.4):

$$\begin{array}{ll} minimize & C\sum_{i=1}^{l}(\xi_i + \xi_i^*) + \frac{1}{2}||w||^2 \\ subject\ to & \begin{cases} y_i - \langle w, x_i \rangle - b \leq \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \end{array} \tag{2.5}$$

The constant $C > 0$ determines the trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated. The ε-insensitive loss function [80] (see equation (2.6)) means that we tolerate errors up to ε and also that errors beyond that value have a linear rather than a quadratic effect. This error function is therefore more tolerant to noise and is thus, more robust.

$$|\xi|_\varepsilon = \begin{cases} 0, & \text{if } |\xi| \leq \varepsilon; \\ |\xi| - \varepsilon, & \text{otherwise.} \end{cases} \tag{2.6}$$

Figure 2.4, shows a plot of the ε-insensitive loss function. Note that only the points outside the shaded region contribute to the cost of the function. It turns out that in most cases, the optimization problem defined by equation (2.5) can be solved more easily in its dual formulation. The dual formulation also provides the capability for extending SVM to nonlinear functions using a standard dualization method based on Lagrange multipliers, as described by Fletcher [25]. So, optimizing the Lagrangian and substituting $t_i = \langle w, x_i \rangle$ for simplicity, we have:

$$\begin{aligned} L = C\sum_{i=1}^{N}(\xi_i + \xi_i^*) + \frac{1}{2}||w||^2 - \sum_{i=1}^{N}(\mu_i\xi_i + \mu_i^*\xi_i^*) \\ - \sum_{i=1}^{N}\alpha_i(\varepsilon + \xi_i + y_n - t_n) - \sum_{i=1}^{N}\alpha_i^*(\varepsilon + \xi_i^* + y_n - t_n) \end{aligned} \tag{2.7}$$

Then, we can substitute for $y(x)$ using the linear model equation: $y(x) = w^T \phi(x) + b$ and set the derivatives of the Lagrangian with respect to 1) w, 2) b, 3) ξ_i and 4) ξ_i^* to zero, giving:

$$\frac{\partial L}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*)\phi(x_i) \tag{2.8}$$

$$\frac{\partial L}{\partial b} = 0 \Rightarrow \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) = 0 \tag{2.9}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow \alpha_i + \mu_i = C \tag{2.10}$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow \alpha_i^* + \mu_i^* = C \tag{2.11}$$

Using these results to eliminate the corresponding variables from the Lagrangian, we see that the dual problem involves maximizing:

$$L'(a, a^*) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*) k(x_i, x_j) - \varepsilon \sum_{i=1}^{N} (\alpha_i + \alpha_i^*) + \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) t_n \tag{2.12}$$

with respect to α_i and α_i^*, where $k(x_i, x_j) = \phi(x_i)^T \cdot \phi(x_j)$ is the kernel function. So, the problem becomes a constrained maximization problem with the box constraints:

$$0 \leq \alpha_i \leq C$$

$$0 \leq \alpha_i^* \leq C$$

And the predictions for new inputs can be made using:

$$y(x) = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) k(x, x_i) + b \tag{2.13}$$

The support vectors are those data points that contribute to predictions given by equation 2.13, in other words those for which either $\alpha_i \neq 0$ or $\alpha_i^* \neq 0$. These are points that lie on the boundary of the ε-tube or outside the tube. All points within the tube have $\alpha_i = \alpha_i^* = 0$.

2.3.7 Clustering

Clustering is the unsupervised classification of patterns into groups (or clusters). The clustering problem has been addressed in many contexts and by researchers in many disciplines [35].

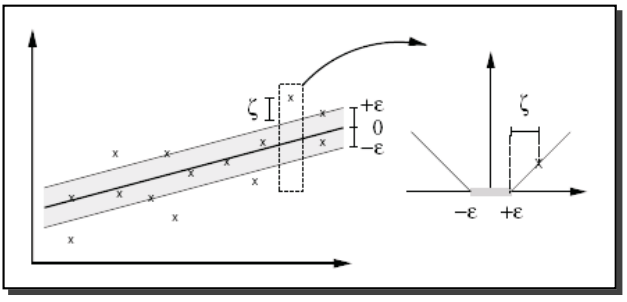


Fig. 2.4 ε-insensitive loss function for SVM

Typical pattern clustering involves the following steps:

(1)**Pattern Representation:** it refers to the number of classes, number of patterns, and features available to the clustering algorithm.
(2)**Definition of a Pattern Proximity:** it is usually measured by a distance function defined on pairs. A simple distance measure such as the Euclidean distance can often be used to reflect dissimilarity between two patterns.
(3)**Clustering or Grouping:** it can usually be hard (a partition of the data into well-defined groups) or fuzzy (where each pattern belongs in certain degree to each of the output clusters).
(4)**Data Abstraction** (if necessary)**:** it is the process of extracting a simple representation of a data set, and a compact description of each cluster, such as the centroid.
(5)**Assessment of Output** (if necessary)**:** it distinguishes a good clustering result from a poor one, it attempts to study the cluster tendency, and it analyzes the clustering result with a specific criterion of optimality.

Although, there is no specific approach that uses only clustering to deal with the problem of reducing the number of objective function evaluations of a problem, clustering techniques are commonly used in combination with surrogates. The computational cost of a surrogate method can become prohibitively high when the size of the training data set is very large, because of the time that it could require to process the data set. In such cases, it is common to cluster the whole data set into several small clusters and then try to build an independent local model from them.

2.3.8 Fitness Inheritance

Fitness Inheritance is a technique that was introduced by Smith et al. [72], whose main motivation is to reduce the total number of fitness function evaluations performed by an evolutionary algorithm. The mechanism works as follows: when assigning the fitness to an individual, some times we evaluate the objective function as usual, but the rest of the time, we assign fitness as an average of the fitness of the

parents. This saves one fitness function evaluation, and is based on the assumption of similarity of an offspring to its parents.

Fitness inheritance must not be always applied, since the algorithm needs to use the true fitness function several times, in order to obtain enough information to guide the search. The percentage of time in which fitness inheritance is applied is called *inheritance proportion*. If this inheritance proportion is 1, the algorithm is most likely to prematurely converge [8].

It is important to mention that some researchers consider this mechanism not so useful in complex or real world problems, under the argument that it has been only applied in "easy" problems. For example, Ducheyne et al. [23] tested the original scheme of fitness inheritance on a standard binary genetic algorithm and the Zitzler-Deb-Thiele (ZDT) [84] multiobjective test problems, concluding that fitness inheritance was not useful when dealing with difficult shapes of the Pareto front. Other authors, however, have successfully applied fitness inheritance to the ZDT and other (more complicated) test problems (see for example [67]).

2.4 Real-World Applications

In this section, we present some selected research work in which a real-world multiobjective engineering optimization problem was solved using a MOEA coupled to a technique for reducing the computational cost involved. There are many engineering disciplines which require expensive function evaluations. From them, we chose aeronautical/aerospace engineering, because it presents problems having high CPU time demand, high nonlinearity and, some times, also high dimensionality. All of these features are also common in other engineering optimization problems, and we consider them representative of the main sources of difficulty in engineering optimization in general.

Aeronautical and aerospace engineering are disciplines in which the solution of multi-objective/multi-disciplinary problems is a standard practice. During the last three decades, the process of engineering design in these industries has been revolutionized as computational simulation has come to play an increasingly dominant role. The increasing demand of optimal and robust designs, driven by time to market, economics and environmental constraints, along with the increasing computing power available, has changed the role of computational simulations from being used only as analysis tools to be used as design optimization tools.

Among the problems with expensive evaluations identified in these disciplines are the following:

- **Aerodynamic Shape Optimization:** This type of optimization problem ranges from $2D$ to complex $3D$ shapes. Typical optimization applications for $2D$ problems comprise Wing and Turbine Airfoil Shape Optimization as well as Inlet/Nozzle design optimization, whereas for $3D$ problems, turbine blade, Wing Shape and Wing-Body configuration design optimizations are typical example applications.

- **Structural Optimization:** The aeronautical/aerospace design philosophy focuses on the design of structures with minimum weight that are strong enough to withstand certain design loads. These two objectives are conflicting in nature and, therefore, the aim of structural optimization is to find the best possible compromise between them. Typical applications for this type of problems comprise structural shape and topology optimization, robust structural design and structural weight optimization.
- **Multidisciplinary Design Optimization:** aeronautical/aerospace design has a multidisciplinary nature, since in many practical design applications, two or more disciplines are involved, each one with specific performances to accomplish. Typical applications for this type of problems are the aeroelastic applications in which aerodynamics and structural engineering are the interacting disciplines.

For all the optimization problems indicated above, the objective function evaluations are routinarily done by using complex computational simulations such as CFD (Computational Fluid Dynamics) in the case of aerodynamic problems, CAA (Computational Aero-Acoustics) for aero-acoustic problems, CSM (Computational Structural Mechanics, by means of Finite Element Method software) for Structural Optimization Problems, or a combination of them in the case of multidisciplinary design optimization problems. Because of their nature, any of these computational simulations have a high computational cost (since they solve, in an iterative manner, the set of partial differential equation governing the physics of the problem) and evaluating the objective functions for the kind of problems indicated above, can take from minutes to hours for a single candidate solution, depending on the fidelity of the simulation.

Nowadays in aeronautical/aerospace industries, MOEAs have gained popularity and are considered as a mature and reliable numerical optimization tool, since they provide to the designers not only with one design solution, but with a set of them from which the tradeoff between the competing objectives can be assessed. This last situation can help decision makers to select a compromise design according to his/her own preferences. Given the high computational cost required for the computational simulations and the population based nature of MOEAs, the use of hybrid methods or meta-models is a natural choice in order to reduce the computational cost of the design optimization process, as indicated by some representative research works that will be described next.

2.4.1 Use of Problem Approximation

As indicated in Section 2.3, this approach tries to replace the original problem by one which is approximately the same as the original one but which is easier to solve. In the context of aeronautical/aerospace engineering problems, where complex CFD, CAA and CSD are employed, the problem can be approximated by using different resolutions in the flow or structural simulation by using either coarse or fine grids. In the case of CFD simulations another level of approximation can be

obtained by solving Euler flows or potential flows instead of Navier-Stokes flow simulations. Some of these techniques are used in the following research works.

Chiba et al. [9, 10] addressed the problem of multidisciplinary wing shape optimization using the ARMOGA (Adaptive Range Multi-Objective Genetic Algorithm) [69] and CFD and CSD Simulations. Three objective functions are minimized: (i) Block Fuel, (ii) Maximum takeoff weight, and (iii) Difference in the drag coefficient between transonic and subsonic flight conditions. In this work, and during the optimization process, an iterative aeroelastic solution is performed in order to minimize the wing weight, with constraints on flutter and strength requirements. For this iterative process, Euler flow solutions (instead of Navier-Stokes flow solutions) are used as a problem approximation in order to reduce the computational cost. Also, a flight envelope analysis is done, which uses high-fidelity CFD Navier-Stokes flow solutions for various flight conditions. The whole optimization process evolves a population of 8 individuals during 16 generations. Authors indicate that they use on the order of 70 Euler and 90 Navier-Stokes simulations per generation of their MOEA.

Sasaki et al. [69, 70] and Obayashi and Sasaki [59], solved a supersonic wing shape optimization problem minimizing four objective functions: (i) drag coefficient at transonic cruise, (ii) drag coefficient at supersonic cruise, (iii) bending moment at the wing root at supersonic cruise condition, and (iv) pitching moment at supersonic cruise condition. In this research study, which also makes use of the ARMOGA algorithm, no iterative aeroelastic analysis is performed, aiming at reducing the associated computational cost. The objective associated with the bending moment at wing root, is approximated by numerical integration of the pressure distribution over the wing surface, as obtained by the CFD analysis.

Lee et al. [48, 51] presented the application of the HAPMOEA (Hierarchical Asynchronous Parallel Multi-Objective Evolutionary Algorithm) [31] to the robust design optimization of an ONERA M6 wing shape. The optimization problem is solved considering uncertainties in the design environment, related to the flow Mach number. The Taguchi method is employed to transform the problem into one with two objectives: (i) minimization of the mean value of an objective function with respect to variability of the operating conditions, and (ii) minimization of the variance of the objective function of each solution candidate, with respect to its mean value. HAPMOEA is based on evolution strategies, incorporating the concept of the Covariance Matrix Adaptation (CMA). It also incorporates a Distance Dependent Mutation (DDM) operator, and a hierarchical set of CFD models (varying the grid resolution of the solver) and populations; small populations are evolved using fine mesh CFD solutions (exploitation of solutions) while large populations are evolved with coarse mesh CFD solutions (exploration of solutions). Good solutions from the coarse mesh populations are transferred to the fine mesh populations. The use of a hierarchical set of CFD models can be seen as different levels of fitness approximation; low-quality fitness approximations are obtained by using coarse mesh grids at low computational cost, while high-quality fitness approximations are obtained by using a fine mesh grid with its associated higher computational cost.

Lee et al. [49, 50] made use of a generic framework for multidisciplinary design and optimization [31] to explore the application of a robust MOEA-based algorithm for improving the aerodynamic and radar cross section characteristics of an UCAV (Unmanned Combat Aerial Vehicle). In both applications, two disciplines are considered, the first concerning the aerodynamic efficiency and the second one dealing with the visual and radar signature of an UCAV airplane. The evolutionary Algorithm employed corresponds to the HAPMOEA indicated above. In this case, the minimization of three objective functions is considered: (i) inverse of the lift/drag ratio at ingress condition, (ii) inverse of the lift/drag ratio at cruise condition, and (iii) frontal area. The problem has, approximately, 100 decision variables, and the first two objective functions are evaluated using a potential flow solver (FLO22) coupled to FRICTION code for obtaining the viscous drag. The use of these last two codes approximates the Navier-Stokes flow solution, considerably reducing the computational cost. The evolutionary system evaluates a total of 1600 solution candidates from which, a Pareto set containing 30 members is obtained. From these nondominated solutions, a single compromise solution is obtained. The authors reported a solution time of 200 hours on a single processor.

2.4.2 Use of RSM by Polynomial Approximation

Lian and Liou [52] used a multi-objective genetic algorithm coupled to a second-order polynomial response surface model for the multiobjective optimization of a three-dimensional rotor blade. The optimization problem consisted of the redesign of the NASA rotor 67 compressor blade, a transonic axial-flow fan rotor which acts as the first stage of a two-stage compressor fan. Two objectives are considered: (i) maximization of the stage pressure raise, and (ii) minimization of the entropy generation. A constraint is imposed on the mass flow rate to have a difference less than 0.1% between the new and the reference design. Blade geometry is constructed from airfoil shapes defined at four span stations, with 32 total design variables. The quadratic response surface model is constructed with 1,024 sampling design candidates and using the IHS (Improved Hypercube Sampling) algorithm [3]. The authors noted that the evaluation of the 1,024 sampling individuals took approximately 128 hours (5.3 days) using eight processors and a Reynolds-Averaged Navier-Stokes CFD simulation. The optimization process for this application is done for 200 generations with a population size of 320 individuals, where objective functions are obtained from the approximation model. Following the optimization process, 12 design solutions are selected from the obtained response surface method Pareto front, and verified with the high fidelity CFD simulation. Objective functions differ slightly from those obtained using the approximation model, and all selected solutions are better in both objective functions than the reference design. A similar research work is presented by Lian and Liou [53, 54], but minimizing the blade weight instead of entropy generation.

Goel et al. [29] used a quintic polynomial response surface method for solving a liquid-rocket injector multiobjective optimization design problem. Four

competing objectives are considered: i) combustion length, ii) injector face temperature, iii) injector wall temperature, and iv) injector tip temperature. In this research, the NSGA-IIa (referred to as archiving NSGA-II [22]), and a local search strategy called "$\varepsilon - constraint$" are adopted to generate a solution set that is used for approximating the Pareto optimal front by a response surface method (RSM). Once the Pareto optimal solutions are obtained, a clustering technique is used to select representative tradeoff design solutions.

Pagano et al. [61] presented an application for three-dimensional aerodynamic shape optimization, particularly the aerodynamic shape of an aircraft propeller. The aim of this multiobjective optimization is to improve an actual propeller performance. The authors considered two conflicting objectives: (i) minimize noise emission level, and (ii) maximize aerodynamic propeller efficiency. For this industrial problem, several disciplines are considered and, therefore the objective function evaluations consider: (a) aerodynamics, (b) structural behavior, and (c) aeroacustics. For each of these, specialized computer simulation codes are employed. Every calculation comprises an iterative coupling procedure (fluids-structures-acoustics) among these simulation codes in order to evaluate a more realistic operating condition. As a consequence, the optimization process becomes computationally demanding. In order to reduce the burden of this high computational cost, the authors made use of design of experiment techniques (DOE), and a quadratic response surface method (RSM) for efficiently exploring the design space. The geometry for the propeller blade is parameterized using a total of 14 design variables. The optimization problem contains constraints on the geometry design variables and on propeller shaft power at two flight conditions; takeoff and cruise, respectively. The evolutionary algorithm employed corresponds to the NSEA+ (Nondominated Sorting Evolutionary Algorithm) as implemented in the OPTIMUS commercial code which is adopted by the authors. The population size for the evolutionary algorithm is set to 20 individuals, and the optimization is run using the DOE and RSM methods. Afterwards, the Pareto front solutions obtained are evaluated using the high fidelity simulation codes. The authors indicated that a total of 340 designs were evaluated using high fidelity simulations. From them, approximately 20 Pareto solutions were obtained, all of them being better than the reference design in the two objectives considered.

2.4.3 Use of Artificial Neural Networks

Rai [64] addressed the problem of multiobjective robust design of a turbine blade airfoil, considering performance degradation due to manufacturing uncertainties. For this problem, the objectives are: (i) minimize the variance of the pressure distribution over the airfoil's surface, and (ii) maximize the wedge angle at the trailing edge. Both objectives must be met subject to the constraint that the required flow turning angle is achieved. Objectives are evaluated by means of a model that modifies the geometry of the airfoil surface following a probability density function that is observed for manufacturing tolerances, and with a CFD simulation for obtaining the flow pressure

distribution. The blade geometry is defined by eight design parameters, but only two of them are varied during the optimization process. The evolutionary algorithm used in this research correspond to a multiobjective version of the differential evolution algorithm previously implemented by the same author and described in [63]. In order to cope with the associated calculation time of the CFD simulations required to evaluate the objective functions, the authors used a *hybrid neural network* comprised of 10 individual *single–hidden–layer feed forward networks*. The optimization is run with a small population size of 10 individuals and during 25 generations.

Arabnia and Ghaly [2] presented a strategy that makes use of multi-objective evolutionary algorithms for aerodynamic shape optimization of turbine stages in three-dimensional fluid flow. The NSGA [74] is used and coupled to an artificial neural network (ANN) based response surface method (RSM) in order to reduce the overall computational cost. The blade geometry, both for rotor and stator blades, is based on the E/TU-3 turbine which is used as a reference design to compare the optimization results to. The multi-objective optimization consists of finding the best distribution of 2D blade sections in the radial and circumferential directions. For this, a quadratic rational Bèzier curve, with 5 control points, is used for each of the two blades. The objective functions to be optimized include: (i) maximization of isentropic efficiency for the stage, and (ii) minimization of the streamwise vorticity. Both objective functions are evaluated using a 3D CFD flow simulation with constraints on: (1) inlet total pressure and temperature, (2) exit pressure, (3) axial chord and spacing, (4) inlet and exit flow angles, and (5) mass flow rate. The authors noted that one CFD simulation took approximately 10 hours. Therefore they resorted to an ANN based RSM. The ANN model with backpropagation, containing a single hidden layer with 50 nodes, was trained and tested with 23 CFD simulations, sampling the design space using the latin hypercubes sampling technique. The optimization process used the ANN model to estimate the objective functions, and the constraints values as well. The population size used in the NSGA was set to 50 individuals, and was run for 150 generations. Finally, the Pareto solutions were evaluated with the CFD flow simulation. From their results, the authors indicated that they obtained design solutions which were better in comparison to the reference turbine design. Indeed, they attained a 1.2% improvement in stage efficiency, which is remarkable considering the small number of design variables used in the optimization process.

Alonso et al. [1] described a procedure for the multi-objective optimization design of a generic supersonic aircraft. The competing design objectives considered were two: i) maximization of aircraft range, and ii) minimization of the perceived loudness of the ground boom signature. Constraints were set for aircraft's structural integrity, take-off field length and landing field length. The objective functions were evaluated using CFD with various fidelity (approximation) levels. In this work, the authors made use of a neural network (NN) based response surface method. The prototype for the NN is a single hidden layer perceptron with sigmoid activation functions, providing a general nonlinear model, which is useful for the high non-linearities present in the objective functions landscapes associated to this problem. The neural network was trained with 300 sampling design solutions, obtained with low fidelity simulations in order to reduce the

computational cost. In their optimization cycle, authors used high fidelity simulations only in promising regions of the design space to do a local exploration. The problem comprised 10 design variables and the NSGA-II [22] was used as the search engine with a population size of 64 and was run for 1000 generations using the surrogate-based objective function.

2.4.4 Use of a Gaussian Process or Kriging

D'Angelo and Minisci [20] used an evolutionary algorithm based on MOPED [19], which is a multi-objective optimization algorithm for continuous problems that uses the Parzen method to build a probabilistic representation of Pareto solutions, with multivariate dependencies among variables. The authors included three modifications to improve a previous implementation of MOPED: (a) use of a kriging model by which solutions are evaluated without resorting to costly computational simulations, (b) use of evolution control, which is adopted to avoid the evolution to converge to a false minima; the mechanism of this technique is to evaluate a subset of individuals or the whole actual generation, with the real simulation model, for a continuous kriging model update, and (c) hybridization of the algorithm; in this case, the selection and ranking of the individuals is different from the original algorithm and some mechanisms borrowed from the NSGA-II algorithm are adopted as well. In their optimization examples, subsonic airfoil shape optimization was performed. The optimization problem considered two objective functions: (i) drag force coefficient, and (ii) lift force coefficient difference with respect to a reference value. Both objectives were minimized. The airfoil geometry is parameterized using Bèzier curves both for its camber line and thickness distribution. In total, 5 design variables were used and constraints were imposed on the objective functions extreme values. The authors indicated that the subsonic airfoil shape optimization presented several difficulties. For example, the true Pareto front was discontinuous and partially converged solutions from the aerodynamic simulation code introduced irregularities in the objective function. It is important to note that the approximation model used (*kriging*) reduced the number of real evaluations to only 2300, considering that the evolution system comprised a population size of 100 individuals and a total of 150 generations.

Song and Keane [73] applied a multi-objective genetic algorithm for studying the shape optimization of a civil aircraft engine nacelle. The primary goal of the study was to identify the tradeoff between aerodynamic performance and noise effects associated with various geometric features for the nacelle. The geometry was parameterized using 40 parameters, 33 of which were considered as design variables. In their study, the authors used NSGA-II [22] as the multi-objective search engine, while a commercial software was used for the CFD evaluations of the three-dimensional flow. Due to the large size of the design space to be explored, as well as the simulations being very time consuming, a kriging surrogate model was adopted in order to keep to a minimum the number of designs being evaluated with the CFD tool. The kriging model was continuosly updated, adding sampling solutions

from the Pareto front obtained using the kriging model and evaluated with the CFD tool. In their research, the authors reported difficulties in obtaining a converged Pareto front (there exist large discrepancies between the approximated and the real Pareto fronts). They attributed this behavior to the large number of variables in the design problem, and to the associated difficulties in obtaining an accurate kriging model for these situations. In order to alleviate this situation, they performed an ANOVA (Analysis of Variance) test to find the variables that contributed the most to the objective function values. After this test, they presented results with a reduced kriging surrogate model, employing only 7 variables. The authors argued that they obtained a similar design with this reduced kriging model at a considerably lower computational effort.

Jeong et al. [38] investigated the improvement of the lateral dynamic characteristics of a lifting-body type re-entry vehicle in transonic flight condition. The problem was posed as a multi-objective optimization problem in which two objectives were minimized: (i) derivative of the yawing moment, and (ii) derivative of the rolling moment. Due to the geometry of the lifting body and the operating flow condition of interest, namely high Mach number and strong vortex formation, the evaluation of the objectives was done by means of a full Navier-Stokes CFD simulation. Since the objectives were derivatives, multiple flow solutions were required to determine their values in a discrete manner through the use of finite differencing techniques. This considerably increased the total computational time due to a large number of calls for the CFD code. The optimization problem considered 4 design variables, and two solutions were sought: the first one without constraints, and the second one constraining the L/D ratio for the lifting-body type reentry vehicle. the authors used the EGOMOP (Efficient Global Optimization for Multi-Objective Problems) algorithm developed by Jeong et al. [37]. Such algorithm was built upon the ideas borrowed from the EGO and ParEGO algorithms from Jone et al. [41] and Knowles et al. [42], respectively. EGOMOP adopts the use of the kriging model as a response surface model, for predicting the function value and its uncertainty. For the exploration of the Pareto solutions, Fonseca's MOGA [26] was used. The initial kriging model was built by using the latin hypercube sampling method for uniformly covering the design space, and the model was continuosly updated.

Voutchkov et al. [81] used the NSGA-II [22] to perform a robust structural design of a simplified FEM jet engine model. This application aimed at finding the best jet engine structural configuration minimizing: the variation of reacting forces under a range of external loads, the mass for the engine and the engine's fuel consumption. These objectives are competing with each other and, therefore, the authors used a multi-objective optimization technique to explore the design space looking for trade-offs among them. The evaluation of the structural response was done in parallel by means of finite element simulations. The FEM model comprised a set of 22 groups of shell elements. The thickness for 15 of these groups were considered as the design variables. Computational time was reduced by using a kriging based response surface method. The optimization problem was posed as a MOP, comprising four objectives (all to be minimized): (i) standard deviation of the internal reaction forces, (ii) mean value of the internal reaction forces, (iii) engine's mass, and (iv) mean value

of the specific fuel consumption. The first two objectives were computed over 200 external load variations. The authors noted that for this class of problem which comprises huge combinations of loads and finite element thicknesses, the multiobjective optimization problem would take on the order of one year of computational time on a single 1 GHZ CPU. Also, they indicated that by using the surrogate model and parallel processing, the optimization time was reduced to about 26 hours in a cluster with 30 PEs (processing elements).

Todoroki and Sekishiro [75, 76] proposed a new optimization method for composite structural components. This approach is based on the use of a multi-objective genetic algorithm coupled to a kriging model, in order to reduce the number of objective function evaluations, and to a FBB (Fractal Branch and Bound) method for the stacking sequence optimization needed in laminar composite structures. The problem consisted of two objectives: (i) minimize the structural weight of a hat-stiffened wing panel, subject to buckling load constraints, and (ii) maximize the probability of satisfying a predefined buckling load. The variables for the problem are a set of mixed real/discrete variables. Real variables correspond to the stiffener geometry definition, while discrete variables correspond to the number of plies for the composite panel. Constraints were imposed on the dimensions of the stiffener, but they were automatically satisfied in the definition of the variables ranges. The authors noted that the buckling load constraint demanded a large computational cost, since it needed a FEM (Finite Element Analysis). For this reason a kriging model was adopted and initialized with sampling points obtained by the LHS (Latin Hypercube Sampling) technique. The optimization cycle consisted of two layers. The upper one driven by the multi-objective genetic algorithm and the kriging model, in which the optimization of the structural dimensions was performed. In the lower layer, the stacking sequences of the stiffener and panels were optimized by means of the FBB method. The evolutionary algorithm was run for 300 generations with a population of 100 individuals, and every 50 generations some nondominated solutions were evaluated with the FEM model, in order to update the kriging model. The authors obtained a Pareto Front that was discontinuous. Also, from the results obtained, a comparison of different designs was made. The solution obtained with the evolutionary algorithm was 3% heavier than a previous design obtained with a conventional method (deterministic), but obtained after only 301 FEM analyses compared to the tens of thousands required by the conventional method.

Choi et al. [11] used the NSGA-II [22] in the solution of a multidisciplinary supersonic business jet design. In this case, the disciplines involved were (i) aerodynamics and, (ii) aeroacoustics. The main objective of this particular problem was to obtain a compromise design having good aerodynamic performance while minimizing the intensity of the sonic boom signature at the ground level. Multiobjective optimization was used to obtain tradeoffs among the following objectives: (i) the aircraft drag coefficient, (ii) initial pressure raise (boom overpressure), and (iii) ground perceived noise level. All the objectives were minimized. The geometry of the aircraft was defined by 17 design variables, involving the modification of the wing platform, its position along the fuselage, and some cross sections and camber for the fuselage. For evaluating the objective functions, a high fidelity Euler

solution was obtained with a very fine grid close to the aircraft's surface. In order to reduce the computational time required for the optimization cycle, a kriging model was employed. Its initial definition was formed with a latin hypercube sampling of the design space with 232 initial solutions, including both feasible and infeasible candidates. Following a kriging based optimization cycle, the Pareto optimal solutions were evaluated with high fidelity simulation tools and used to update the kriging model. In the example, constraints were imposed on some geometry parameters, and on the aircrfat's operational conditions. No special constraint-handling mechanism was adopted other than discarding the solution candidates that did not satisfy the constraints, which were mostly geometrical. From their results, the authors noted that after the first design cycle using the kriging based NSGA-II, 59 feasible solutions were obtained. It is important to note that all the solutions obtained were better in both objectives compared to a base design. Another important issue in this particular application was that the kriging model did not perform as well as in other applications. The reason for this behavior was the high nonlinear physics involved in the two disciplines considered, which required, in consequence, more design cycles in the optimization.

In related work, Chung and Alonso [12] and Chung et al. [13] solved the same previously defined multidisciplinary problem, but using the μ-GA Algorithm, from Coello Coello and Toscano Pulido [15, 16]. This change was aimed at reducing the total number of function evaluations during the optimization process. This μ-GA algorithm used a population size of 3 to 6 individuals and an external file to keep track of the nondominated solutions obtained so far. In the study reported in [12], the design cycles were performed using a kriging model. Two design cycles were executed, each one consisting of 150 solution candidates using the latin hypercube sampling technique applied around a base design in the first cycle. For the second cycle, the sampling was applied around the best solution obtained in the previous cycle. The authors reported that they obtained a very promising Pareto front estimation with only 300 functions evaluations. In the second study, reported in [13], the authors proposed an tested the GEMOGA (Gradient Enhanced Multiobjective Genetic Algorithm). The basic idea of this algorithm is to enhance the Pareto solutions with a gradient based search. One important feature of the algorithm is that gradient information is obtained from the kriging model. With this, the computational cost is not considerably increased.

Kumano et al. [44] used Fonseca's MOGA [26] for the multidisciplinary design optimization of wing shape for a small jet aircraft. In this study, four objectives were considered: (i) drag at the cruise condition, (ii) drag divergence between cruising and off-design condition, (iii) pitching moment at the cruising condition, and (iv) structural weight of the main wing. All these objectives were minimized. In this study, the optimization process was also performed by means of a kriging model, and such model was continuosly updated after a certain prescribed number of iterations (defined by the user), adding new nondominated points obtained from the optimization steps.

2.4.5 Use of Clustering

Langer et al. [47] applied an integrated approach using CAD (Computer Aided Design) modeling with a MOEA for structural shape and topology optimization problems. The application presented in this research, dealt with the structural optimization of a typical instrument panel of a satellite, and considered two objectives: (i) minimize the instrument panel mass, and (ii) maximize the first eigenfrecuency. The problem contained a mixed continuous/discrete set of variables. 17 design variables were used, from which 3 were discrete, which consider the number of stringers to use in the panel, as well as the plate and stringer materials. The authors solved the optimization problem for three shape and topology optimization cases: (a) a panel without instruments, (b) a panel with instruments at fixed positions, and (c) a panel with instrumental placing. They made use of polynomial based response surface methods in order to reduce the computational cost. Multiple local approximation models were constructed using a clustering technique. In all the examples included, the population size was set to 200 and was evolved for 20 generations. The evaluation of the objective functions comprised four load cases: (a) quasi-static acceleration, (b) modal analysis, (c) sinusoidal vibration loads, and (d) 'pseudo temperature' load. This latter load case, constrained the positioning of the instruments on the panel, since it imposed a limiting operating temperature for a specific instrument. The first three load cases were evaluated in parallel using FEM (Finite Element Method) simulations on a cluster of 32 workstations.

2.4.6 Use of Radial Basis Functions

Cinnella et al. [14] presented the airfoil shape optimization for transonic flows of BZT (Bethe-Zel'dovich-Thompson) fluids, by using a multi-objective genetic algorithm. This application explored the design of airfoil shapes in turbine cascades which could exploit the benefits of BZT transonic flows past airfoils. In the application, the authors proposed two optimization problems which aimed at finding optimal airfoil geometries both for (i) non–lifting airfoils, and (ii) lifting airfoils. In both cases GA-Based approaches were used as search engines. In the second case, the optimization problem considered two design objectives: (i) maximize lift at BZT subcritical conditions, and (ii) Minimize wave drag while maximizing lift for supercritical BZT flow conditions. Therefore, a bi-objective problem was solved, and the evolutionary algorithm helped the designers to find trade-off solutions between these two design points. The multi-objective genetic algorithm used in the second case was the NSGA [74]. In previous related work [18], a population size of 36 and 24 generations were used (totaling 864 objective function evaluations obtained from CFD), based on the constraint that the whole CFD calculation time had to be kept inferior to one week (the evaluation time for each individual varied from 5 to 10 min in a PC equipped with a Pentium Processor). In order to reduce the computational cost, the authors included an ANN (Artificial Neural Network) based on radial basis functions, formed by an input layer, an intermediate layer, and an output layer.

The weights of the linear combinations were determined through a training procedure. The number of neurons involved was taken as the number of individuals in the training set. The first training set was formed with all the solutions obtained from the first two generations. Afterwards, the objective functions were approximated with the ANN-RBF model, and the training set was updated by adding a 30% of "exactly evalauted" individuals per generation. With this technique the authors obtained similar design solutions with approximately 60% less computational cost.

Kampolis and Giannakoglou [7] solved the inverse design of an isolated airfoil at two operating conditions. For this design problem, two reference airfoil and operating conditions were defined (these solutions could be seen as the extreme portions of the Pareto front), and a MOEA was used to find the tradeoff solutions between them. The MOEA adopted was SPEA-2 [85]. In their approach, the authors proposed the use of a radial basis function meta-model.

2.5 Conclusions and Future Research Paths

We have described several techniques which have been coupled to MOEAs, aiming to reduce the computational cost of evaluating the objective functions of a multi-objective optimization problem. Additionally, some selected real-world applications of such techniques were also presented as case studies in which these hybrid schemes led to substantial reductions in the computational cost. The main aim of this review was to provide a general overview of this area, which we believe that may be of interest both for MOEA researchers who may be looking for new algorithmic design challenges, and for practitioners, who may benefit from combining MOEAs with surrogate methods or any other approximation techniques that they normally use to reduce the computational cost of their simulations.

From the application examples reviewed here, we observed that the most preferred methods seem to be problem approximation, kriging and polynomial interpolation, followed by the use of neural networks and radial basis functions. Our study of the small sample of real-world applications presented here, also led us to outline some of the future research paths that seem promising within this area:

- **Model Selection Guidelines:** Since the high computational cost involved in applications such as those described here preclude us from any exhaustive experimentation, the existence of guidelines that allow us to identify which sort of method could be a good choice for a given problem would be of great help. To the authors' best knowledge no guidelines of this sort have even been reported in the specialized literature.
- **Hybridization:** Approximation models can be used not only to replace the objective function evaluations, but also to estimate first-order information (e.g., gradient information). This could lead to the use of hybrids of MOEAs with gradient-based methods. An example of this type of approach is presented in Chung et al. [13], where solutions are improved by the use of gradient information obtained from a kriging model. This sort of hybridization scheme is, however, relatively scarce in the literature until now.

- **Use of Multiple Approximation Models:** Most authors report the use of a single approximation model. However, it may be worth exploring the combination of several of them for exploiting either their global or their local nature. This idea has been explored in the past, for example, by Mack et al. [55], by using a combination of polynomial respose surface methods and radial basis functions, for performing global sensitivity analysis and shape optimization of bluff bodies. Also, Glaz et al. [28] adopted three approximation models, namely polynomial, kriging, and radial basis functions. This combined approach, adopted a weighted estimation from the different models, which was used to reduce the vibration for a helicopter rotor blade. To the authors' best knowledge, no similar combination of approaches has ever been reported when using MOEAs.
- **Automatic Switching:** Considering that every approximation model has particular properties in terms of global or local accuracy, and that the selection of the "best" approximation method to use for a particular application can also be considered a difficult task, one promising research area is to develop mechanisms allowing to automatically switch from one approximation method to a different one, as the optimization process is being executed. For example, a global approximation method (i.e., coarse-grained) could be used for exploration of the design space, while a more locally accurate method (i.e., fine-grained) might be used for solution exploitation.
- **Sampling Techniques:** The accuracy of the approximation highly depends on the sampling and updating technique used. In most cases, the initial sampling is defined by a latin hypercube sampling, aiming at covering as much as possible the design space. This can be considered as a general technique. Another possibility is to use application-dependent sampling techniques, where the initial sampling design points are selected on the basis of reference or similar solutions. One example of this sort of situation is reported by Chung et al. [13] and by Chung and Alonso [12], where the initial approximation models are built around a reference design in decision variable space.

References

[1] Alonso, J., LeGresley, P., Pereyra, V.: Aircraft Design Optimization. Mathematics and Computer in Simulation 79, 1948–1958 (2008)
[2] Arabnia, M., Ghaly, W.: A strategy for multi-objective shape optimization of turbine stages in three-dimensional flow. In: 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, British Columbia, Canada (2008)
[3] Beachkofski, B.K., Grandhi, R.V.: Improved Distributed Hypercube Sampling. In: 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Denver, CO, USA (2002)
[4] Bhattacharya, M., Lu, G.: A dynamic approximate fitness based hybrid ea for optimization problems. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 1879–1886 (2003)
[5] Bishop, C.M.: Neural Networks for Pattern Recognition. Oxford University Press, UK (1995)

[6] Bueche, D., Schraudolph, N., Koumoutsakos, P.: Accelerating evolutionary algorithms with gaussian process fitness function models. IEEE Transactions on Systems, Man, and Cybernetics: Part C 35(2), 183–194 (2005)
[7] Kampolis, I.C., Giannakoglou, K.C.: A multilevel approach to single- and multiobjective aerodynamic optimization. Computer Methods in Applied Mechanics and Engineering 197, 2963–2975 (2008)
[8] Chen, J.H., Goldberg, D., Ho, S.Y., Sastry, K.: Fitness inheritance in multi-objective optimization. In: Proceedings of Genetic and Evolutionary Computation Conference. Morgan Kaufmann, San Francisco (2002)
[9] Chiba, K., Obayashi, S.: Data mining for multidisciplinary design space of regional-jet wing. AIAA Journal of Aerospace Computing, Information, and Communication 4(11), 1019–1036 (2007)
[10] Chiba, K., Obayashi, S., Nakahashi, K., Morino, H.: High-Fidelity Multidisciplinary Design Optimization of Wing Shape for Regional Jet Aircraft. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 621–635. Springer, Heidelberg (2005)
[11] Choi, S., Alonso, J.J., Chung, H.S.: Design of a low-boom supersonic business jet using evolutionary algorithms and an adaptive unstructured mesh method. In: AIAA Paper 2004-1758, 45th AIAA/ASME/ASCE/AHS/ASC Structure, Structural Dynamics and Materials Conference, Palm Springs, CA, USA (2004)
[12] Chung, H.S., Alonso, J.J.: Multiobjective optimization using approximation model-based genetic algorithms. In: AIAA Paper 2004-4325, 10th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Albany, New York, USA (2004)
[13] Chung, H.S., Choi, S., Alonso, J.J.: Supersonic business jet design using a knowledge-based genetic algorithm with an adaptive, unstructured grid methodology. In: AIAA Paper 2003-3791, 21st Applied Aerodynamics Conference, Orlando, Florida, USA (2003)
[14] Cinnella, P., Congedo, P.M.: Optimal Airfoil Shapes for Viscous Transonic Flows of Dense Gases. In: AIAA Paper 2006-3881, 36th AIAA Fluid Dynamics Conference and Exhibit, San Francisco, California, USA (2006)
[15] Coello Coello, C.A., Toscano Pulido, G.: A Micro-Genetic Algorithm for Multiobjective Optimization. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 126–140. Springer, Heidelberg (2001)
[16] Coello Coello, C.A., Toscano Pulido, G.: Multiobjective Optimization using a Micro-Genetic Algorithm. In: Spector, L., Goodman, E.D., Wu, A., Langdon, W., Voigt, H.M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M.H., Burke, E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001), pp. 274–282. Morgan Kaufmann Publishers, San Francisco (2001b)
[17] Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007)
[18] Congedo, P.M., Corre, C., Cinnella, P.: Airfoil Shape Optimization for Transonic Flows of Bethe-Zel'dovich-Thompson Fluids. AIAA Journal 45(6), 1303–1316 (2007)
[19] Costa, M., Minisci, E.: MOPED: A multi-objective parzen-based estimation of distribution algorithm for continuous problems. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 282–294. Springer, Heidelberg (2003)
[20] D'Angelo, S., Minisci, E.A.: Multi-objective evolutionary optimization of subsonic airfoils by kriging approximation and evolutionary control. In: 2005 IEEE Congress on Evolutionary Computation (CEC 2005), Edinburg, Scotland, vol. 2, pp. 1262–1267 (2005)

[21] Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
[22] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
[23] Ducheyne, E.I., De Baets, B., De Wulf, R.: Is Fitness Inheritance Useful for Real-World Applications? In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 31–42. Springer, Heidelberg (2003)
[24] Emmerich, M., Giotis, A., Özdenir, M., Bäck, T., Giannakoglou, K.: Metamodel-assisted evolution strategies. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 371–380. Springer, Heidelberg (2002)
[25] Fletcher, R.: Practical Methods of Optimization. John Wiley and Sons, New York (1989)
[26] Fonseca, C.M., Fleming, P.J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In: Forrest, S. (ed.) Proceedings of the Fifth International Conference on Genetic Algorithms, University of Illinois at Urbana-Champaign, pp. 416–423. Morgan Kauffman Publishers, San Mateo (1993)
[27] Giunta, A., Watson, L.: A comparison of approximation modeling techniques: Polynomial versus interpolating models. Tech. Rep. 98-4758, AIAA (1998)
[28] Glaz, B., Goel, T., Liu, L., Friedmann, P.P., Haftka, R.T.: Application of a Weighted Average Surrogate Approach to Helicopter Rotor Blade Vibration. In: AIAA Paper 2007–1898, 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Honolulu, Hawaii, USA (2007)
[29] Goel, T., Vaidyanathan, R., Haftka, R.T., Shyy, W., Queipo, N.V., Tucker, K.: Response Surface Approximtion of Pareto Optimal Front in Multi-Objective Optimization. Computer Methods in Applied Mechanics and Engineering 196, 879–893 (2007)
[30] Goel, T., Vaidyanathan, R., Haftka, R., Shyy, W., Queipo, N., Tucker, K.: Response surface approximation of pareto optimal front in multiobjective optimization. Tech. Rep. 2004-4501, AIAA (2004)
[31] Gonzalez, L.F., Périaux, J., Srinivas, K., Whitney, E.J.: A generic framework for the design optimisation of multidisciplinary uav intelligent systems using evolutionary computing. In: AIAA Paper 2006-1475, 44th AIAA Aerospace Science Meeting and Exhibit, Reno, Nevada, USA (2006)
[32] Hardy, R.L.: Multiquadric equations of topography and other irregular surfaces. J. Geophys. Res. 76, 1905–1915 (1971)
[33] Hong, Y.S., Lee, H., Tahk, M.J.: Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks. Engineering Optimization 35(1), 91–102 (2003)
[34] Hüscken, M., Jin, Y., Sendhoff, B.: Structure optimization of neural networks for aerodynamic optimization. Soft Computing 9(1), 21–28 (2005)
[35] Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. 31(3), 264–323 (1999), http://doi.acm.org/10.1145/331499.331504
[36] Jensen, M.T.: Reducing the Run-Time Complexity of Multiobjective EAs: The NSGA-II and Other Algorithms. IEEE Transactions on Evolutionary Computation 7(5), 503–515 (2003)
[37] Jeong, S., Minemura, Y., Obayashi, S.: Optimization of combustion chamber for diesel engine using kriging model. Journal of Fluid Science and Technology 1, 138–146 (2006)

[38] Jeong, S., Suzuki, K., Obayashi, S., Kurita, M.: Improvement of nonlinear lateral characteristics of lifting-body type reentry vehicle using optimization algorithm. In: AIAA Paper 2007–2893, AIAA infotech@Aerospace 2007 Conference and Exhibit, Rohnert Park, California, USA (2007)
[39] Jin, R., Chen, W., Simpson, T.: Comparative studies of metamodeling techniques under miltiple modeling criteria. Tech. Rep. 2000-4801, AIAA (2000)
[40] Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing 9(1), 3–12 (2005)
[41] Jone, D., Schonlau, M., Welch, W.: Efficient global optimization of expensive black-box function. Journal of Global Optimization 13, 455–492 (1998)
[42] Knowles, J.: ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation 10(1), 50–66 (2006)
[43] Knowles, J., Nakayama, H.: Meta-modeling in multiobjective optimization. In: Branke, J., Deb, K., Miettinen, K., Slowinski, R. (eds.) Multiobjective Optimization-Interactive and Evolutionary Approaches, pp. 245–284. Springer, Heidelberg (2008)
[44] Kumano, T., Jeong, S., Obayashi, S., Ito, Y., Hatanaka, K., Morino, H.: Multidisciplinary design optimization of wing shape for a small jet aircraft using kriging model. In: AIAA Paper 2006-932, 44th AIAA Aerospace Science Meeting and Exhibit, Reno, Nevada, USA (2006)
[45] Kung, H., Luccio, F., Preparata, F.: On finding the maxima of a set of vectors. Journal of the Association for Computing Machinery 22(4), 469–476 (1975)
[46] Becerra, R.L., Coello, C.A.C.: Solving Hard Multiobjective Optimization Problems Using ε-Constraint with Cultured Differential Evolution. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 543–552. Springer, Heidelberg (2006)
[47] Langer, H., Pühlhofer, T., Baier, H.: A multi-objective evolutionary algorithm with integrated response surface functionalities for configuration optimization with discrete variables. In: AIAA Paper 2004–4326, 10th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization Conference, Albany, New York, USA (2004)
[48] Lee, D., Gonzalez, L., Periaux, J., Srinivas, K.: Robust design optimisation using multi-objective evolutionary algorithms. Computer & Fluids 37, 565–583 (2008)
[49] Lee, D., Gonzalez, L., Srinivas, K., Periaux, J.: Robust evolutionary algorithms for uav/ucav aerodynamic and rcs design optimisation. Computer & Fluids 37, 547–564 (2008b)
[50] Lee, D.S., Gonzalez, L.F., Srinivas, K., Auld, D.J., Wong, K.C.: Aerodynamics/rcs shape optimisation of unmanned aerial vehicles using hierarchical asynchronous parallel evolutionary algorithms. In: AIAA Paper 2006-3331, 24th AIAA Applied Aerodynamics Conference, San Francisco, California, USA (2006)
[51] Lee, D.S., Gonzalez, L.F., Srinivas, K., Periaux, J.: Multi-objective robust design optimisation using hierarchical asynchronous parallel evolutionary algorithms. In: AIAA Paper 2007-1169, 45th AIAA Aerospace Science Meeting and Exhibit, Reno, Nevada, USA (2007)
[52] Lian, Y., Liou, M.S.: Multiobjective Optimization Using Coupled Response Surface Model and Evolutinary Algorithm. In: AIAA Paper 2004–4323, 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York, USA (2004)

[53] Lian, Y., Liou, M.S.: Multi-Objective Optimization of a Transonic Compressor Blade Using Evolutionary Algorithm. In: AIAA Paper 2005–1816, 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference, Austin, Texas, USA (2005)
[54] Lian, Y., Liou, M.S.: Multi-objective Optimization of Transonic Compressor Blade Using Evolutionary Algorithm. Journal of Propulsion and Power 21(6), 979–987 (2005)
[55] Mack, Y., Goel, T., Shyy, W., Haftka, R., Queipo, N.: Multiple Surrogates for the Shape Optimization of Bluff Body-Facilitated Mixing. In: AIAA Paper 2005–333, 43rd AIAA Aerospace Sciences Metting and Exhibit, Reno, Nevada, USA (2005)
[56] Nakayama, H., Sawaragi, Y.: Satisficing trade-off method for multi-objective programming. In: Grauer, M., Wierzbicki, A. (eds.) Interactive Decision Analysis, pp. 113–122. Springer, Heidelberg (1984)
[57] Nakayama, H., Yun, Y.: Support vector regression based on goal programming and multi-objective programming. In: IEEE World Congress on Computational Intelligence (2006)
[58] Nakayama, H., Inoue, K., Yoshimori, Y.: Approximate optimization using computational intelligence and its application to reinforcement of cable-stayed bridges. In: Zha, X., Howlett, R. (eds.) Integrated Intelligent Systems for Engineering Design, pp. 289–304. IOS Press, Amsterdam (2006)
[59] Obayashi, S., Sasaki, D.: Self-organizing map of pareto solutions obtained from multi-objective supersonic wing design. In: AIAA Paper 2002–0991, 40th Aerospace Science Meeting and Exhibit, Reno, Nevada, USA (2002)
[60] Ong, Y.S., Nair, P.B., Keane, A.J., Wong, K.W.: Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In: Jin, Y. (ed.) Knowledge Incorporation in Evolutionary Computation. Studies in Fuzziness and Soft Computing, pp. 307–332. Springer, Heidelberg (2004)
[61] Pagano, A., Federico, L., Barbarino, M., GUida, F., Aversano, M.: Multi-objective Aeroacoustic Optimization of an Aircraft Propeller. In: 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, British Columbia Canada (2008)
[62] Pierret, S.: Turbomachinery blade design using a Navier-Stokes solver and artificial neural network. ASME Journal of Turbomachinery 121(3), 326–332 (1999)
[63] Rai, M.M.: Robust Optimal Aerodynamic Design Using Evolutionary Methods and Neural Networks. In: AIAA Paper 2004-778, 42nd AIAA Aerospace Science Meeting and Exhibit, Reno, Nevada, USA (2004)
[64] Rai, M.M.: Robust Optimal Design With Differential Evolution. In: AIAA Paper 2004-4588, 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, New York, USA (2004)
[65] Rasheed, K., Ni, X., Vattam, S.: Comparison of methods for developing dynamic reduced models for design optimization. Soft Computing 9(1), 29–37 (2005)
[66] Ratle, A.: Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 87–96. Springer, Heidelberg (1998)
[67] Reyes Sierra, M., Coello Coello, C.A.: A Study of Fitness Inheritance and Approximation Techniques for Multi-Objective Particle Swarm Optimization. In: 2005 IEEE Congress on Evolutionary Computation (CEC 2005), vol. 1, pp. 65–72. IEEE Service Center, Edinburgh (2005)
[68] Sacks, J., Welch, W., Mitchell, T., Wynn, H.: Design and analysis of computer experiments (with discussion). Statistical Science 4, 409–435 (1989)

[69] Sasaki, D., Obayashi, S., Nakahashi, K.: Navier stokes optimzation of supersonic wings with four objectives using evolutionary algorithms. In: AIAA Paper 2001–2531, 15th AIAA Computational Fluid Dynamics Confernce, Anaheim, CA, USA (2001)
[70] Sasaki, D., Obayashi, S., Nakahashi, K.: Navier-Stokes Optimization of Supersonic Wings with Four Objectives Using Evolutionary Algorithms. Journal of Aircraft 39(4), 621–629 (2002)
[71] Abboud, K., Schoenauer, M.: Surrogate deterministic mutation: Preliminary results. In: Collet, P., Fonlupt, C., Hao, J.-K., Lutton, E., Schoenauer, M. (eds.) EA 2001. LNCS, vol. 2310, pp. 104–115. Springer, Heidelberg (2002)
[72] Smith, R.E., Dike, B.A., Stegmann, S.A.: Fitness inheritance in genetic algorithms. In: SAC 1995: Proceedings of the 1995 ACM symposium on Applied computing, pp. 345–350. ACM Press, New York (1995)
[73] Song, W., Keane, A.J.: Surrogate-based aerodynamic shape optimization of a civil aircraft engine nacelle. AIAA Journal 45(10), 265–2574 (2007)
[74] Srinivas, N., Deb, K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation 2(3), 221–248 (1994)
[75] Todoroki, A., Sekishiro, M.: Dimensions and laminates optimization of hat-stiffened composite panel with buckling load constraint using multi-objective ga. In: AIAA Paper 2007–2880, AIAA infotech@Aerospace 2007 Conference and Exhibit, Rohnert Park, California, USA (2007)
[76] Todoroki, A., Sekishiro, M.: Modified efficient global optimization for a hat-stiffened composite panel with buckling constraint. AIAA Journal 46(9), 2257–2264 (2008)
[77] Ulmer, H., Streicher, F., Zell, A.: Model-assisted steady-state evolution strategies. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 610–621. Springer, Heidelberg (2003)
[78] Ulmer, H., Streichert, F., Zell, A.: Evolution startegies assisted by gaussian processes with improved pre-selection criterion. In: Proceedings of IEEE Congress on Evolutionary Computation, pp. 692–699 (2003)
[79] Vapnik, V.: Statistical Learning Theory. Wiley, Chichester (1998)
[80] Vapnik, V.N.: The Nature of Statistical Learning. Springer, Heidelberg (1995)
[81] Voutchkov, I., Keane, A.J., Fox, R.: Robust structural design of a simplified jet engine model, using multiobjective optimization. AIAA Paper 2006–7003, Portsmouth, Virginia, USA (2006)
[82] Williams, C.K.I., Rasmussen, C.E.: Gaussian processes for regression. In: Touretzky, D.S., Mozer, M.C., Hasselmo, M.E. (eds.) Advances in Neural Information Processing Systems, vol. 8. MIT Press, Cambridge (1996)
[83] Won, K., Ray, T.: Performance of kriging and cokriging based surrogate models within the unified framework for surrogate assisted optimization. In: Congress on Evolutionary Computation, pp. 1577–1585. IEEE, Los Alamitos (2004)
[84] Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2), 173–195 (2000)
[85] Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In: Giannakoglou, K., Tsahalis, D., Periaux, J., Papailou, P., Fogarty, T. (eds.) EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Athens, Greece, pp. 95–100 (2002)

Chapter 3
Multilevel Optimization Algorithms Based on Metamodel- and Fitness Inheritance-Assisted Evolutionary Algorithms

Kyriakos C. Giannakoglou and Ioannis C. Kampolis

Abstract. This chapter is concerned with the efficient use of metamodel–assisted evolutionary algorithms built in multilevel or hierarchical schemes for the solution of computationally expensive optimization problems. Existing methods developed by other researchers or the authors' group are overviewed and a new enhancement based on fitness inheritance is proposed. Whereas conventional evolutionary algorithms require a great number of calls to the evaluation software, the use of low cost surrogates or metamodels, trained on the fly on previously evaluated individuals for pre–evaluating the evolving populations, reduce noticeably the *CPU* cost of an optimization. Since, the metamodel training requires a minimum amount of previous evaluations, the starting population is evaluated on the problem–specific model. Fitness inheritance is introduced in this context so as to approximate the objective function values in place of metamodels. In addition, to profit of the availability of evaluation or parameterization models of lower fidelity and *CPU* cost and/or refinement methods, a multilevel search algorithm relying also on the use of metamodels is presented. The algorithm may optionally operate as hierarchical–distributed (many levels performing distributed optimization) or distributed–hierarchical (more than one sub–populations undergoing their own hierarchical optimizations) to further reduce the design cycle time. The proposed algorithms are generic and can be used to solve any kind of optimization problems. Here, aerodynamic shape optimization problems, including turbomachinery applications, are used to demonstrate the efficiency of the proposed methods. A new computationally demanding application, namely the optimization of a 3*D* compressor blade is also shown.

Kyriakos C. Giannakoglou · Ioannis C. Kampolis
National Technical University of Athens,
School of Mechanical Engineering,
Lab. of Thermal Turbomachines, Parallel CFD & Optimization Unit,
P.O. Box 64069, Athens 157 10, Greece
e-mail: kgianna@central.ntua.gr
http://velos0.ltt.mech.ntua.gr/research

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 61–84.
springerlink.com

3.1 Introduction

The term *computationally expensive optimization problems* is referred to design or optimization applications requiring an excessive number of calls to the costly evaluation software for locating the optimal solution(s). Typical examples are optimizations in which the evaluation of candidate solutions calls for the numerical solution of p.d.e.'s or is based on Monte–Carlo techniques to account for uncertainties. Without loss in generality, we will restrict ourselves to design–optimization problems with aerodynamic performance criteria. Therefore, the evaluation software might be any *Computational Fluid Dynamics* (*CFD*) code. Depending on the selected flow model, the problem dimension (2*D* or 3*D*) and the complexity of the flow domain (affecting the computational grid size, if such a grid is needed), the cost of running the *CFD* code may range from a couple of minutes to some hours on many *CPUs*.

In aerodynamic shape optimization, the use of either *gradient–based* methods or *global search metaheuristics* is steadily increasing, [48, 82]. Though they usually appear as "rival" methods, they can be hybridized to create more efficient optimization methods (see also [58], chapter 16). In the present chapter, an evolutionary algorithm (*EA*, [4, 34, 57]) is the key search method. *EAs* are assisted by metamodels and fitness inheritance, hybridized with gradient–based methods and structured as multilevel search algorithms. They are also used in distributed search schemes, properly adapted for use on multiprocessor platforms and may become Grid enabled.

EAs are gradient–free methods that may accommodate any ready-to-use analysis software, even a commercial–off–the–shelf one without having access to its source code. In aerodynamic optimization problems, they unfortunately become computationally demanding if (some, at least, of) the add–on features discussed in this chapter or elsewhere in this book are not used. This is due to the high number of candidate solutions that must be evaluated. For *EAs* to become routine industrial tools, much focus has been placed on methods reducing the number of evaluations required and, thus, their *CPU* cost. To this end, most of the existing papers rely on *surrogate evaluation models* or *metamodels*. The latter stand for evaluation methods of lower accuracy and *CPU* cost. The so–called *metamodel–assisted EAs* (*MAEAs*, [26, 67]) use both the exact and costly *problem–specific evaluation model* and the approximate and computationally cheap *metamodel*), according to coupling schemes to be discussed below.

In the so–called *MAEAs* with *off–line* trained metamodels [7, 8, 20, 25, 35, 38, 41, 62, 71, 73], the metamodel is trained in advance, i.e. separately from the evolution which is exclusively based on them. The problem–specific tool is used to evaluate a number of selected samples which the metamodel should be trained on and for cross–checking the outcome of the metamodel–based optimization.

In *MAEAs* with *on–line* trained metamodels [6, 18, 19, 26, 33, 38, 44, 66, 77, 83] the metamodel(s) and the problem–specific model are used in an interleaving way during the evolution. The metamodels may be local (valid over a part only of the design space) or global (valid over the entire design space). The more frequently metamodels are used (in place of the exact model), the greater the gain in *CPU* cost.

Another interesting alternative, regarding the possible coupling of metamodels and stochastic search techniques, lies in the so–called metamodel–assisted memetic algorithms (*MAMAs*) [24, 28, 68, 84]. In [24], the metamodels are (also) used for the refinement (local search) of selected population members in the spirit of Lamarckian learning. The hierarchical and/or distributed optimization schemes presented in this chapter can certainly accommodate metamodel–based local search processes but this is beyond the scope of this chapter.

The method discussed in this chapter is an extension of the *MAEA* originally proposed in [26, 29, 32], for single– (*SOO*) and multi–objective optimization (*MOO*) problems, [44]. A key feature of this method is the *inexact pre-evaluation* (*IPE*) technique. Apart from the starting population which is evaluated on the problem–specific model, the subsequent generations use local metamodels. These are trained on the fly on a subset of previously evaluated neighbors of each new population member. *MAEAs* can also be configured as distributed search methods (*DMAEA*), by handling intercommunicating sub–populations or *demes*, [45].

The evolutionary search can also be carried out via *hierarchical* or *multilevel* schemes, [39, 43, 46], by incorporating more than one *optimization levels* associated with different evaluation software, [17, 37, 39, 43, 46, 78], different search techniques, [49, 61, 76], and different chromosome sizes, [54], or numbers of design variables, [12, 16]. The communication between subsequent levels (one– or two–way migrations of individuals) is important. Gains from using the multilevel algorithm and the underlying hierarchy or *EAs* with improved evolution operators and/or metamodels are superimposed.

This chapter reviews the aforementioned hierarchical methods. Over and above, *fitness inheritance* is introduced so as to reduce the number of offspring that undergo exact evaluation during the first generations of a *MAEA*. In those generations the archived data are inadequate to train metamodels that generalize well. As a remedy fitness inheritance is employed and only the top approximately evaluated population members must be re–evaluated on the problem–specific model, according to the *IPE* concept. According to the method presented in [39, 43, 46], the distributed search of optimal solutions is structured in levels, each of which employs a *MAEA* or a gradient–based method; such a scheme will be referred to as a *hierarchical distributed MAEA* (*HDMAEA*, even if a gradient–based method is employed on the higher level). In a *HDMAEA*, the number of demes may differ from level to level, inter–level migrations do not depend on levels' partition into demes and intra–level (inter–deme) migrations also occur. Apart from *MAEAs*, gradient–based methods can also be used on any level but none of the levels is associated with a metamodel only. On the other hand, *distributed hierarchical EAs* (*DHEAs*) where the hierarchical search is carried out within each deme (this is why this is called *distributed hierarchical*, rather than the other way round), have also been devised. The lower pass evaluations rely on metamodels, so the abbreviation *DHEA* can be replaced by *DHMAEA*. Only a few best performing individuals migrate upwards, where problem–specific evaluations of increasing fidelity and CPU cost take place. The *DHMAEA* demes communicate regularly by exchanging top individuals.

3.2 Metamodel–Assisted *EAs* and Distributed *MAEAs*

This section presents a *MAEA*, [26, 31] and its distributed variant (*DMAEA*, [45]). Note that at least one of the levels of the multilevel algorithms presented below relies on either a *MAEA* or a *DMAEA*.

In a conventional (μ,λ) *EA*, with μ parents and λ offspring, each generation costs as many as λ calls to the problem–specific evaluation software. A metamodel–assisted variant (*MAEA*; see also [29] for *SOO* and [44] for *MOO* problems) starts exactly as the conventional *EA* by evaluating the first random population on the problem–specific tool and archiving paired inputs–outputs in a database. In all subsequent generations, the population members are inexactly pre–evaluated (*IPE*) using either fitness inheritance or local metamodels. More precisely, a few next generations employ fitness inheritance. Once the database exceeds a user–defined minimum number of entries, local metamodels are used in place of fitness inheritance. These are trained on the fly, separately for each new individual, on a small number of neighboring database entries. As with fitness inheritance, the most promising members are re–evaluated. Note that, to create the next generation, the evolution operators use mixed exact and approximate scores.

For the re–evaluation, the $\sigma\lambda$ top pre–evaluated population members are selected. The σ value may vary between user–defined lower and upper bounds, $0 < \sigma_{min} \leq \sigma \leq \sigma_{max} < 1$. In each generation, the $\sigma_{min}\lambda$ top individuals are unconditionally re–evaluated. Up to $(\sigma_{max} - \sigma_{min})\lambda$ more offspring which take on better fitness values (on the metamodel) than the current optimal solution can be re–evaluated, too.

In Pareto front seeking *EAs* for *MOO* problems, a scalar fitness value must be computed for each population member, based on dominance, niching etc criteria (*NSGA*, [10], *NSGA–2*, [11], *SPEA*, [85], *SPEA–2*, [86] to mention a few of them). The σ value is determined as in *SOO* problems. However, as explained in [44], the use of the *IPE* technique in *MOO* problems may become less efficient than in *SOO*. In *SOO*, the database of previously evaluated solutions (among which the training patterns are selected) becomes well populated close to the optimal solution and the metamodels give progressively better predictions. This is not the case in *MOO*, where the trained local metamodel must predict well along a front of solutions. In [44], the replacement of the conventional radial-basis function (*RBF*) network (used as metamodel) by its generalized counterpart, using less centers than training patterns and acting as approximation, rather than interpolation, method, greatly improves the performance of the *IPE* technique in *MOO*. The *RBF* centers are selected as described in section 3.3.2.

On the other hand, distributed *EAs* (*DEAs*), which handle a small number of medium–sized sub–populations (the so–called *demes* or *islands*), outperform single–population *EAs*. The evolution operators are restricted within each deme and inter–deme exchanges of promising individuals take place. Different evolution policies over different demes can be used, [13]. *DEA* variants can be devised by changing the communication topology (ring, grid, etc.), the migration frequency and/or the selection and replacement policies, [1]. The *IPE* algorithm is directly applicable

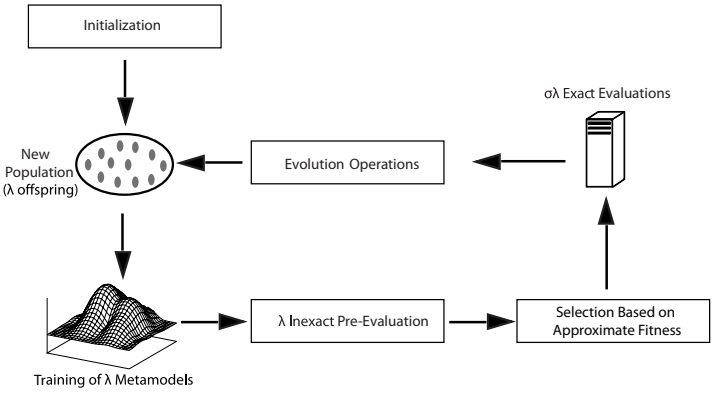


Fig. 3.1 Metamodel–assisted *EA* (*MAEA*), with the —IPE technique

to *DEAs*, as described in [45] where a distributed *MAEA* (*DMAEA*) was proposed. In non–hierarchical optimization, all demes share a single database archiving previously evaluated individuals.

DEAs belong to the class of multi–population parallel *EAs*, [1, 9, 52, 64], exhibiting high parallel efficiency. Each deme may use a cluster of processors for the concurrent evaluation of its offspring and different demes can be evaluated concurrently. *EAs* (in either conventional or distributed or even hierarchical form) are also enabled for deployment in the Grid environment, [53]. The present method, [27], is also Grid–enabled, using a middleware allowing concurrent offspring evaluations, [52]. Evaluation requests are queued to the Gridway metascheduler, [59], using the *DRMAA* library API. Passing through different layers of middleware from the Grid level down to the execution host, each job is executed on a remote resource. The Globus Toolkit, [21], and Ganglia, [55], are employed for discovery, user authentication and inter–connection of remote resources. Local resource management (submission and execution of jobs within the cluster) is accomplished using Condor, [81].

3.3 Surrogate Evaluation Models for *MAEAs*

Fitness inheritance, [15, 79], and the *RBF* networks, [36, 74], which are used herein as surrogate evaluation models, are discussed. We assume an optimization problem with N design variables and M objectives. Let $\mathbf{x} \in \mathscr{R}^N$ and $\mathbf{y} \in \mathscr{R}^M$ denote the design variable array and the corresponding array of fitness or cost values.

3.3.1 Fitness Inheritance

Fitness inheritance, as originally proposed in [79], computes the fitness of any offspring $\mathbf{x}^*$ from the fitness values of its parents $\mathbf{x}^{(*,p)}, p = 1, \ldots, \rho$ using either *average inheritance* (setting $\mathbf{y}^*$ equal to the average value of $\mathbf{y}^{(*,p)}$) or *proportional*

inheritance (setting $\mathbf{y}^*$ equal to the weighted average of $\mathbf{y}^{(*,p)}$, depending on the degree of similarity between $\mathbf{x}^*$ and $\mathbf{x}^{(*,p)}$). In the present method, each offspring inherits an approximate fitness value from its parents according to their distance in the design space. In specific, $\mathbf{y}^* = \sum_{p=1}^{\rho} w_p \mathbf{y}^{(*,p)}$, where

$$w_i = 1 - \frac{\left\| \mathbf{x}^{(*,i)} - \mathbf{x}^* \right\|}{\sum_{p=1}^{\rho} \left\| \mathbf{x}^{(*,p)} - \mathbf{x}^* \right\|} \tag{3.1}$$

3.3.2 Radial Basis Function (RBF) Networks

An *RBF* network, [36, 74], performs the mapping $\mathscr{R}^N \rightarrow \mathscr{R}^M$ using three layers of processing units: the input with N nodes where the input vectors are applied to, the hidden with K processing nodes and the output layer with M nodes where responses emerge. Signals propagate through the network in the forward direction. The K links connecting the hidden nodes to the output one (assume $M = 1$) are associated with synaptic weights $\mathbf{w}$ to be computed during the training process. The K hidden layer units are associated with the so–called *RBF* centers, $\mathbf{c}^{(k)} \in \mathscr{R}^N$, $k = 1, \ldots, K$, which are the centers of the neuron's nonlinear radial-basis activation function $\mathscr{G} : \mathscr{R}^N \mapsto \mathscr{R}$. Here, the Gaussian function $\mathscr{G}(\|\mathbf{x} - \mathbf{c}^{(k)}\|_2, r_k) = exp\left(-\frac{\|\mathbf{x}-\mathbf{c}^{(k)}\|_2^2}{r_k^2}\right)$ is used, where r_k are the *RBF* radii. The network response is

$$y(\mathbf{x}^*) = \sum_{k=1}^{K} w_k \mathscr{G}(\|\mathbf{x}^* - \mathbf{c}^{(k)}\|_2, r_k) \tag{3.2}$$

Therefore, assuming K centers $\mathbf{c}^{(k)}$ and $T > K$ training patterns $\mathbf{x}^{(t)}$, the network training requires the solution of the following system of linear equations

$$\sum_{k=1}^{K} w_t \mathscr{G}(\|\mathbf{x}^{(t)} - \mathbf{c}^{(k)}\|_2, r_k) = y^{(t)} \ , \quad t = 1, T$$

(where $y^{(t)}$ are the known responses) using the least squares algorithm.

The selection of the *RBF* centers is carried out as in [44]. It is based on *self–organizing maps* (*SOMs*, [22, 36] and an iterative scheme with both unsupervised and supervised learning. During the unsupervised learning, the *SOMs* classify the training patterns into K clusters. Each cluster gives a single *RBF* center $\mathbf{c}^{(k)}$ and the corresponding radius r_k, through heuristics based on distances between the centers, [5, 23, 36, 47, 60]. During the supervised learning, the synaptic weights are calculated by minimizing the approximation error over the training set.

Herein, a variant of *RBF* networks enhanced by *Importance Factors* (*IFs*, denoted by I_n, $n = 1, \ldots, N$), as proposed in [29], is used. The modified network incorporates the I_n factors which quantify how much the network response is affected by each design variable. The higher the I_n value the higher the response sensitivity with respect to the n–th input variable. A *weighted norm* defined by

$$\left\|\mathbf{x}^* - \mathbf{c}^{(k)}\right\|_{wei} = \sqrt{\sum_{n=1}^{N} I_n \left(x_n^* - c_n^{(k)}\right)^2}, \quad I_n = \frac{\left|\frac{\partial y^{(b)}}{\partial x_n}\right|}{\sum_{i=1}^{N}\left|\frac{\partial y^{(b)}}{\partial x_i}\right|} \tag{3.3}$$

is used instead of eq. 3.2. The computation of I_n is based on analytically computed derivatives according to the *RBF* network signal propagation formulas, [29] and the current best solution (index b) and must be updated whenever a new optimal solution is found.

The use of *RBF* networks trained on both fitness values and gradients is an interesting alternative, [30, 42, 65]; their presentation is beyond the scope of this chapter.

3.4 Assessment of *MAEA* and *DMAEA*

To justify the need for using *MAEAs* or *DMAEAs*, instead of conventional *EAs*, two examples follow. More examples can be found in [19, 26, 29, 44, 45].

The first case is concerned with the optimization of the *RAE*2822 (reference, *ref*) airfoil, at $M_\infty = 0.73$ and $a_\infty = 3.19^o$ (inviscid flow), for minimum drag coefficient C_D while maintaining the same lift (C_L). Thus, the cost function to be minimized was $F = (C_{L,ref} - C_L)^2 + 5 \cdot C_D$. A $(15,60)$ *EA* and a $3 \times (5,20)$ *DEA* (i.e. with three demes) were used, with and without metamodels. In the *DEA*, the demes were fully connected and the migration operator was employed every four generations; two top members migrated from each deme to all the rest. In both *MAEA* and *DMAEA*, σ was fixed to 0.1. So, in each generation, six offspring were re–evaluated on the *CFD* software per generation. In fig. 3.2, the convergence histories of the four algorithms in terms of the number of evaluations on the *CFD* model are shown. It is evident that the *DMAEA* outperforms all other algorithmic variants. In the same figure, the pressure coefficient C_P distribution over the optimal airfoil shows that the shock wave formed over the reference *RAE*2822 airfoil has been eliminated.

The second case demonstrates the gain expected in *MOO* problems using *MAEAs* instead of *EAs*. This problem is concerned with the design of an axial compressor cascade airfoil for $\alpha_{in} = 47^o$, $M_{in} = 0.6$, Reynolds number based on chord $Re_C = 8.5 \cdot 10^5$ and inlet turbulence intensity $\tau_u = 2.0\%$. The axial velocity density ratio was 1.1325 and the stagger angle 30^o. The two objectives were (a) minimization of the cascade total pressure loss coefficient ω and (b) maximization of the static pressure rise p_{out}/p_{in}. The airfoil was parameterized using Bézier curves and the control point coordinates were the design variables. The optimization was carried out using a $(60,15)$ *EA*, a $(60,15)$ *MAEA* using *RBF* networks for the *IPE* phase and the same *MAEA* using the enhanced *RBF* networks (i.e. the ones using *IFs*), both with $\sigma = 0.10$. The three methods are compared in fig. 3.3, illustrating Pareto front approximations computed at the same *CPU* cost. The *MAEA* using the *RBF* networks and the *IFs* provides the best approximation to the Pareto front.

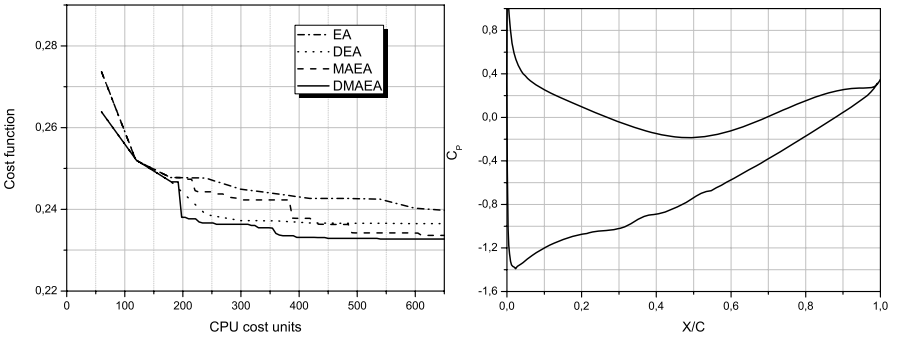


Fig. 3.2 Redesign of the *RAE*2822 airfoil. Convergence history plotted in terms of the number of evaluations on the problem–specific model (left). C_P distribution on the optimal airfoil (right)

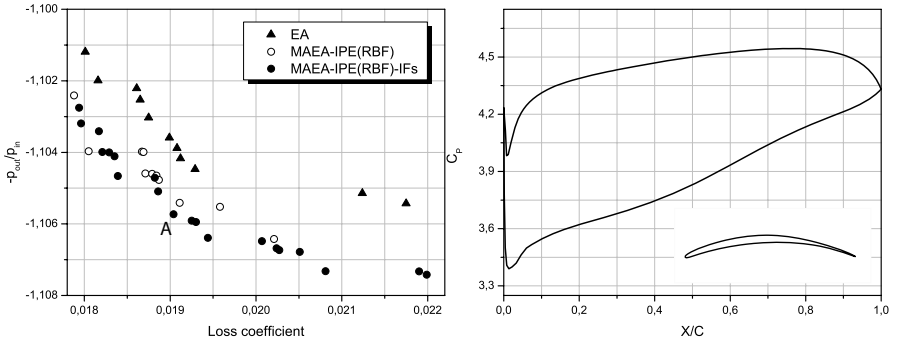


Fig. 3.3 Design of an axial compressor cascade airfoil. Pareto front approximations using a conventional *EA* and two *MAEAs*. The contour and pressure coefficient (C_P) distribution on the Pareto front member marked with *A* (right)

3.5 Multilevel Search Algorithms and the Underlying Hierarchy

In this section, the structure of an optimization algorithm in more than one levels, which is the backbone of the present method, is described. Three different ways of employing hierarchy within the multilevel scheme, fig. 3.4, are presented. These modes can be used either separately or in combination. Two level schemes ($L=2$) are described since expansion to $L>2$ is straightforward.

3.5.1 The Three Multilevel Modes – Defining a HDMAEA

(a) Multilevel Evaluation: In this mode, a *different evaluation software* is assigned to each level. Since the low level undertakes the detection of near–optimal solutions and delivers them to the high level for refinement, evaluation models of low cost

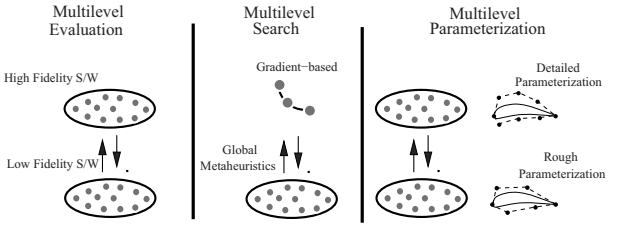


Fig. 3.4 The three modes of the multilevel (herein L=2) algorithm at a glance: (a) Multilevel Evaluation (b) Multilevel Search and (c) Multilevel Parameterization

and fidelity are associated with it. The problem–specific (high fidelity) evaluation model is employed on the high level. One– or two–way inter–level migrations can be used. In one–way migrations, a small number of best performing individuals is directed upwards, with no feedback at all. On the high level, immigrants replace badly performing and/or randomly selected population members (assume that an *EA* or a *MAEA* is used on both levels). In the two–way migration scheme, promising individuals from the high level may also move downwards to stimulate better search in their neighborhood.

The multilevel evaluation algorithm is often configured with different population sizes per level, usually a large population on the low level and a small one on the high level to compensate for the high *CPU* cost per evaluation and synchronize better with the low level. The latter certainly depends on the *CPU* cost ratio and the number of processors used.

The multilevel evaluation mode is appropriate for use in aerodynamic shape optimization problems. For instance, in flows dominated by viscous effects, either a Navier–Stokes solver coupled with wall–functions on a coarse grid or an integral boundary layer method can be employed on the low level. The high level must rely on a model with the desired accuracy, such as a Navier–Stokes solver with a low–Reynolds number turbulence model and a much finer grid. Alternatively, the same *CFD* tool running with different grids and/or employing different convergence criteria can be used on the two levels.

(b) Multilevel Search: In this mode, each level is associated with a *different search technique* (*EA*, conjugate gradient, Sequential Quadratic Programming *SQP*, etc., [63]). Stochastic search techniques, such as *EAs*, are preferably used on the low level to adequately explore the design space. On the high level, the refinement of promising solutions can be carried out through gradient–based methods or stochastic, individual–based methods (such as simulated annealing). The migration of promising solutions is, preferably, bi–directional to accentuate the exploration capabilities of low level *EAs*.

The coupling of stochastic, population–based methods and gradient–based algorithms is not new. In the literature, hybrid optimization methods are mostly restricted to *SOO* problems or *MOO* ones where the objectives are concatenated in a single

function. In [39], a "genuine" multilevel search methods for *MOO* problems was proposed. In this method, on the low level, the *EA* or *MAEA* computes approximations to the Pareto front using a known scalar utility assignment (*SPEA–2*, [86]). On the high level, the scalar utility gradient is computed for a few selected non–dominated solutions and a descent algorithm is used to improve them with respect to all objectives. This is carried out using the chain rule after replacing the derivative (delta function) of the non–differentiable *SPEA–2* utility function (terms that involve the Heaviside function) with a differentiable approximation.

In case a gradient–based search is used on the high level, the gradient of the objective function must be computed or approximated. To this end, in aerodynamic optimization, the adjoint approach can be used at about the cost of an additional flow solution, [2, 69, 70].

(c) Multilevel Parameterization: The third mode associates a *different set of design variables* with each level. On the low level, a problem with just a few design variables is solved. On the high level, the detailed problem parameterization is used. To support migrations, (exact or approximate) transformations between different parameterizations must be available. All immigrants must be transformed to the parameterization scheme of the destination level. Working with *NURBS* curves or surfaces, knot insertion and removal properties and formulas [72] must be used to switch between levels with different parameterizations.

In constrained problems, the term *different parameterization* may also imply that the constraints are handled differently on each level. For instance, on the low level, constraints may be relaxed or even ignored, allowing thus even infeasible but promising solutions to be sent to the high level.

The three modes can be used either separately or altogether. As mentioned above, the term *HDMAEA* is used to denote an optimization method of multilevel structure which may accommodate distributed search on all or some of its levels. In a *HDMAEA*, all levels using *DEAs* or *DMAEAs* regularly perform intra–level (i.e. inter–deme) migrations, over and above to the inter–level ones. In case a level has not yet reached the generation (or iteration, in gradient–based methods) marked for inter–level migration whereas the other did, the one ahead suspends evolution, waiting for synchronization. This is also valid for the intra–level migration between demes.

When ineffective inter–level migrations occur (i.e. when all immigrants perform worse than the destination level individuals) for a user–defined number of consecutive generations/iterations, the evolution on the lower level terminates.

3.5.2 Distributed Hierarchical Search – DHMAEA *vs.* HDMAEA

Apart from the previous *HDMAEA* modes, a *distributed hierarchical EA* (fig. 3.5) can also be devised using the same ingredients. Since metamodels can optionally be used during the lower pass, this will also be referred to as *distributed hierarchical metamodel–assisted EA* (*DHMAEA*). *DHMAEAs* clearly distinguish from *HDMAEAs* since, in the former, hierarchy is employed within each deme, [40].

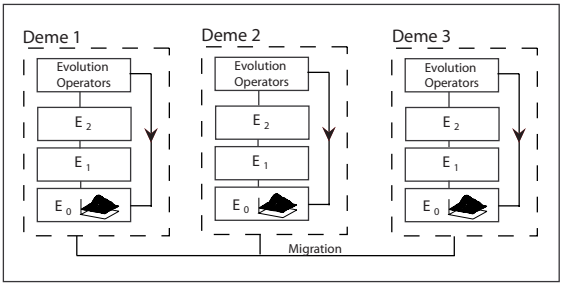


Fig. 3.5 The distributed hierarchical *EA* (*DHEAm*) with one metamodel (E_0) and two problem–specific tools (E_1, E_2)

Let E_s $(s = 0, ..., S)$ denote the $S+1$ evaluation tools. By convention, E_0 is the metamodel and E_S the high fidelity evaluation model. In the sake of simplicity, all of them use the same number of design variables. Schemes that use different (coarse and fine) parameterizations on each level have also been devised, see [39], but these will not be discussed further. The maximum number of offspring per generations and deme to be evaluated on E_s is $\lambda_s = \lambda \prod_{i=0}^{S} \pi_i$. The user–defined parameters $\pi_i \in [0,1]$ determine the percentage of individuals evaluated on E_{i-1} to be re–evaluated on E_i. The π_i values decrease with i, starting from $\pi_0 \equiv 1$ (i.e. $\lambda_0 \equiv \lambda$).

The only non–problem–specific evaluation model is E_0 (i.e. the metamodel). Training the metamodels requires a database of samples, shared among all demes. So, evaluations on E_0 are postponed until the database of previously evaluated individuals (using the low fidelity problem–specific model) exceeds a user–defined minimum number of entries. Upon completion of a few starting generations (during which $\pi_1 = 1$), metamodels are separately trained and used to approximate the objective vector value(s) of each new offspring.

3.6 Assessment of Multilevel–Hierarchical Optimization

All four multilevel schemes have been assessed on a number of applications. For some standard benchmarks as well as information on detailed parameter settings, the reader should refer to [39, 40, 43]. In what follows four of them are presented here. The first three demonstrate the expected gain from separately using the multilevel evaluation, search and parameterization algorithms whereas the last one demonstrates the use of *DHMAEA*.

The first study is concerned with the design of a $2D$ transonic compressor cascade airfoil with minimum total pressure losses and maximum static pressure rise (i.e. two objectives), as originally presented in [43]. The optimization was carried out for isentropic exit Mach number equal to $M_{out,is}$=0.6, a_{in}=55.4^o and Re_C=$1.7 \cdot 10^5$. Constraints were imposed on the minimum airfoil thickness at various chordwise positions, not allowing the airfoil to become unrealistically thin.

The *multilevel evaluation* mode, with two levels, was used. On the high level, a $(5,20)$ *MAEA* $(0.05 \leq \sigma \leq 0.15)$ and a Navier–Stokes equation solver with the

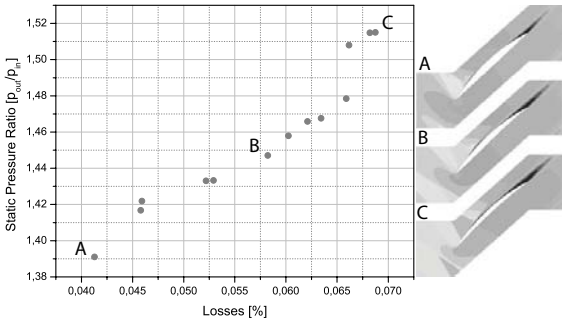


Fig. 3.6 Design of a 2*D* transonic compressor cascade using a *HDMAEA* (*multilevel evaluation*): Pareto front obtained at the cost of 311 cost units; airfoil shapes and iso–Mach number contours for three Pareto members. From [43]

Spalart–Allmaras turbulence model, [80], on a fine unstructured mesh were used. On the low level, search was carried out using a $3\times(5,20)$ *DMAEA* with the same σ bounds. Each evaluation on the low level (E_1) was based on an integral boundary layer method, [14] with *CPU* cost about 30 times less than that of an E_2–based evaluation. So the overall *CPU* cost of the optimization was $k_2+k_1/30$, where k_1 and k_2 are the numbers of evaluations on E_1 and E_2, respectively. On both levels, the fronts were populated with up to 35 non–dominated solutions, by screening out excess front members from overcrowded front parts. The *IPE* phase started after each level database archived 150 (on E_1) and 130 (on E_2) evaluated individuals. The inter–level migrations (with 6 individuals moving upwards and 5 downwards) were performed every 5 (high) and 20 (low level) generations. In addition, for the low level *DMAEA*, the inter–deme migration occurred every 4 generations. The Pareto front approximation shown in fig. 3.6 was obtained at the cost of 311 cost units. This corresponds to 209 and 1778 evaluations on E_2 and E_1, respectively.

The *multilevel search HDMAEA* mode (with two levels) is demonstrated on the second case where an isolated airfoil with optimal performance at two operating points: [O.P.1: $\alpha_\infty=1.0^\circ, M_\infty=0.5$] and [O.P.2: $\alpha_\infty=5.0^o, M_\infty=0.2$], was designed, [39]. The integrals of the deviation of the static pressure distribution along the airfoil contour from two target distributions were used as objectives. On the high level, the optimization used a gradient–based method (*SQP*) whereas a $3\times(5,20)$ *DMAEA* was employed on the low level. 10 individuals were simultaneously improved during one *SQP* iteration (or equivalent "high level generation"). Every 10 *SQP* iterations and 30 *DMAEA* generations, 10 promising individuals migrated upwards and incorporated into the high level on condition that they performed better than the current *SQP* "population". Simultaneously, 5 individuals migrated downwards to join the *DMAEA*. On the low level, the *IPE* phase (with $0.05\leq\sigma\leq0.20$) started once 150 evaluated individuals have been archived.

The same Euler equations solver was used on both levels. On the high level, the objective function gradients were computed by the adjoint method, [69]; thus, the cost per high level evaluation was twice as much as a low level one (i.e. two

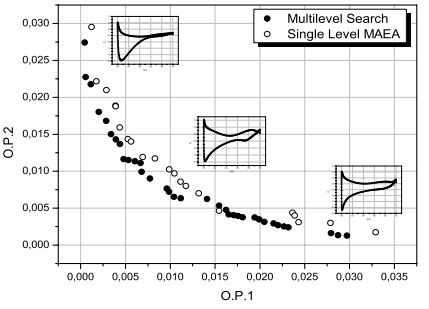


Fig. 3.7 Design of an isolated airfoil at two operating points. With the same *CPU* cost, the *multilevel search* variant of the *HD-MAEA* outperforms a single level *MAEA*. From [39]

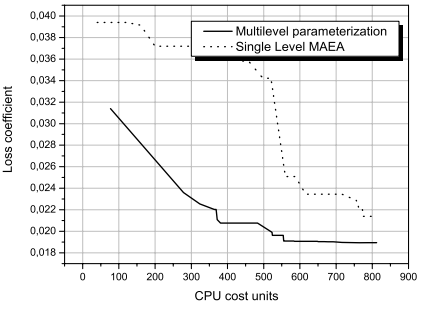


Fig. 3.8 Design of a compressor stator cascade. Convergence plot of a *multilevel parameterization* algorithm and a single–level *MAEA*. From [39]

equivalent flow solutions). As mentioned in a previous section, the gradient of the *SPEA–2* fitness assignment technique with respect to the design variables was computed and used by the descent method on the high level. Fig. 3.7 presents the computed Pareto front approximation at the cost of 1000 equivalent flow solutions and compares it with that computed by a single level *MAEA* at the same cost.

The third case demonstrates the gain achieved by the *multilevel parameterization* mode during the design of a compressor stator cascade. The objective was to minimize the total pressure loss coefficient ω, [39], with four constraints on the minimum allowed airfoil thickness. The flow conditions were $M_{out,is}=0.65$, $\alpha_{in}=57^o$ and $Re_C=6.0\cdot 10^5$. The axial velocity density ratio was equal to 1.124 and the stagger angle was 35^o. The airfoil pressure and suction sides were parameterized using Bézier curves with 7 and 17 control points, respectively.

New control points (knots) were inserted to the existing parametrization of individuals migrating upwards without changing their shape. Unfortunately, this was not the case of knot removal. The incremental knot insertion formula (from $N+1$ to $N+2$) used for the internal control points of a Bézier curve is

$$\mathbf{R}_i=\frac{i}{N+1}\mathbf{r}_{i-1}+\left(1-\frac{i}{N+1}\right)\mathbf{r}_i,\quad 0<i<N+1 \tag{3.4}$$

where $\mathbf{r}$ and $\mathbf{R}$ are the position vectors of the control points before and after the insertion, respectively. It is evident that $\mathbf{R}_0=\mathbf{r}_0$, $\mathbf{R}_{N+1}=\mathbf{r}_N$.

In this case, $(40,8)$ and $(25,5)$ *MAEAs* were used on the high and low levels, respectively, with the same *CFD* software. Fig. 3.8 presents the convergence behavior of the high level *MAEA* and compares it with a single level *MAEA*. The best solution achieved using the two–level parameterization algorithm had $\omega=1.895\%$ whereas the single level *MAEA* led to $\omega=2.14\%$, both at the same *CPU* cost.

The last case in this section stands for the two–objective design of a compressor cascade airfoil, aiming at minimum total pressure losses and maximum static

pressure rise, subject to airfoil thickness related constraints. The flow conditions were $M_{out,is}$=0.45, α_{in}=47^o, Re_C=8.41·10^5. Local metamodels (E_0), a high fidelity *CFD* model (E_2) and a low fidelity one (E_1 based on the same iterative solution methods with relaxed convergence criteria) were used. The *CPU* cost ratio of E_1 and E_2 was about 0.1 : 1,

In this case, three algorithms are compared:

1. A single–level $(15,60)$ *EA* using E_2–based evaluations.
2. A *DHEA* with three $(5,20)$ demes and two evaluation passes (on E_1 and E_2) within each deme. During the first pass, the 20 offspring were evaluated on E_1 and only the top two of them were re–evaluated on E_2.
3. A $3 \times (5,20)$ *DHMAEA* and three evaluation passes per deme. Upon completion of the 20 E_0–based evaluations, the 10 best among them were re–evaluated on E_1 and the 3 best of them on E_2. All metamodels were trained on the fly, using previously evaluated (on E_1) neighbors.

In both distributed algorithms, the migration operator was employed every 8 generations by exchanging two individuals between any pair of demes. The two emigrants of each deme were selected after ranking the 20 population members in terms of their fitness, irrespective of the evaluation tool used. In each destination deme, the immigrants replaced the worst performing members evaluated on the same or a lower fidelity model.

The three algorithms are compared in terms of the hypervolume indicator, [87], fig. 3.9, which quantifies the part of the objective space (up to a user–defined point) dominated by the front; larger indicator values correspond to better Pareto front approximations. The combined use of metamodels and hierarchical schemes achieves better performance. The same figure also shows the Pareto front approximation (at the cost of 1000 *CPU* cost units) computed by the *DHMAEA*.

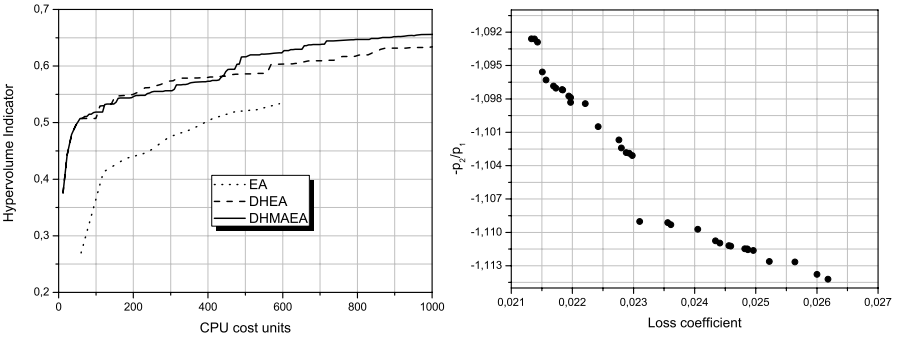


Fig. 3.9 Two–objective compressor cascade airfoil design: Evolution of the hypervolume indicator for the three tested algorithms (left) and the Pareto front approximation computed by the *DHMAEA* (right)

3.7 Optimization of an Annular Cascade

This section is dedicated to a real engineering application, with high *CPU* cost per evaluation. This new case is concerned with the optimization of a $3D$ annular compressor cascade. An existing cascade with 19 straight blades (chord length equal to C=0.1 m, stagger angle equal to 51.4^o) was used as the reference one. The blades were mounted on the casing by forming a 2.0 mm clearance with the stationary or rotating hub with radius equal to 0.244 m. This study was concerned only with the stationary hub. The facility layout and experimental measurements can be found in [56] and a numerical flow study in [75]. The inlet total pressure/temperature and peripheral/radial flow angle distributions at the cascade inlet were given. The cascade mass flow rate was 13.2 kg/s and the maximum inlet Mach number was about 0.6. The purpose of this study was to redesign the airfoil of the straight blade so as to achieve minimum mass-averaged pressure loss coefficient PLC_{ave}. PLC_{ave} results from the radial distribution of the circumferentially averaged pressure loss coefficient $PLC(r) = \frac{p_{t,inl} - p_t(r)}{p_{t,inl} - p_{inl}}$.

Each airfoil side was parameterized using 15 *NURBS* control points, 5 of which were allowed to vary. So the design variables were 20 in total (two coordinates per design variable). Geometrical constraints were imposed to ensure that the optimal airfoil would not become thinner than the reference one by more than 90%. In addition, the mean exit flow angle $\bar{\alpha}_{out}$ was not allowed to exceed 53^o.

The *CFD* software used was a Navier–Stokes solver employing a time–marching, vertex–centered, finite volume formulation for unstructured grids, [50]. Turbulence is modeled using the Spalart–Allmaras model, [80]. The inlet turbulence intensity was set to 1.5%.

The hierarchical optimization algorithm described in section 3.5.2 with a single deme was used. The two problem–specific evaluation models were both based on the aforementioned *CFD* software, using different turbulence modeling and grid sizes. The high fidelity model (E_2) used the low–Reynolds number Spalart–Allmaras model on a hybrid–unstructured grid of about 1.000.000 nodes, fig. 3.11. Hexahedral and prismatic elements were generated over the blade surface 400×95 structured–like grid and the casing; the distance of the first layer of nodes off the

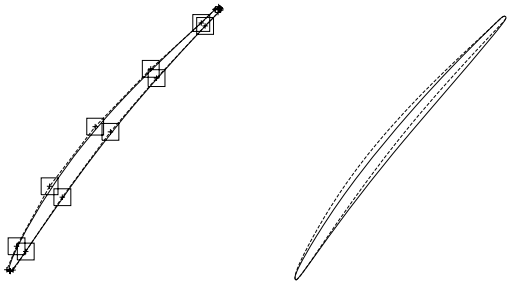


Fig. 3.10 Optimization of an annular cascade: Reference airfoil (continuous line), its control points' polygon (dashed line) and the design variables' bounds (left). Reference (dashed) and optimal (continuous) blade airfoil contours (right)

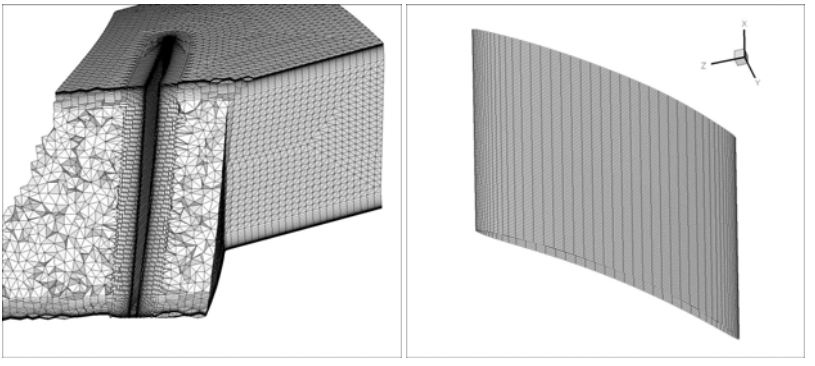


Fig. 3.11 Optimization of an annular cascade: Views of the hybrid fine grid used for E_2–based evaluations (left) and view of the surface grid over the reference blade (right)

walls satisfied the usual constraint on the nondimensional distance y^+ from the wall ($y^+ < 1$). All subsequent layers of elements were arranged using a geometrical progression law with ratio ω. A layer of pyramids was used to interface the hexahedral structured–like layers and the tetrahedra filling the inner part of the domain. The same grid generation procedure was also used to support the low fidelity tool E_1, with different, however, parameters. E_1 used the Spalart–Allmaras turbulence model with wall functions and a coarser hybrid–unstructured grid of about 600.000 nodes. The blade surface was discretized using a 400×85 grid; larger ω and distances of the first layer of nodes off the wall were used. Table 3.1 compares the basic features of the fine (for E_2) and coarse (for E_1) grids.

This optimization was carried out on a cluster of 25 nodes ($2 \times$Quad Core Xeon, 2.0Ghz, with 16 GB RAM each). Each population member was evaluated in parallel on a single node. The wall clock time required for a single E_2–based evaluation was about 6 hours. The same evaluation on E_1 required about 1.2 hours. Below, one *CPU* cost unit is assigned to each evaluation on E_2 and 0.2 units to each on E_1. It should become clear that even if an evaluation failed quite early (during grid generation), this was assigned the full *CPU* cost. These are also summarized in table 3.1.

To compare the modeling accuracy of E_1 and E_2, the reference cascade was firstly analyzed using both tools. The computed radial distributions of the circumferentially mass–averaged total pressure on the (same) exit plane are compared in fig. 3.12, left. Differences between the two models reflect on the PLC_{ave} values which were equal to 0.1498 (based on E_1) and 0.1189 (on E_2).

A $(20,60)$ hierarchical *EA*, with a single population was used. Fitness inheritance and *RBF* networks supported the *IPE* process. During the first generation, all individuals were evaluated on E_1 and only the 6 top of them were re–evaluated on E_2. On the second generation, the fitness inheritance technique was activated. The most promising (between 5 and 10 of them) members in the population were re–evaluated

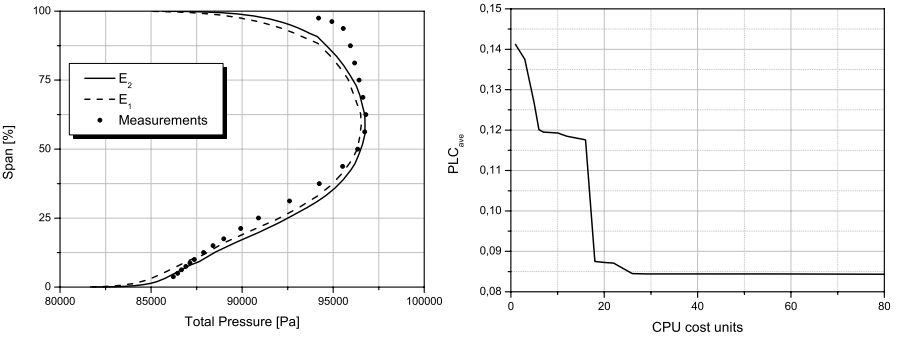


Fig. 3.12 Optimization of an annular cascade: Comparison of E_1, E_2–based predictions and measurements on the reference blade (left). Radial distributions of the circumferentially mass–averaged total pressure on the exit plane; the tip clearance is located at zero span percentage (left). Convergence of the hierarchical *EA* (right)

Table 3.1 Optimization of an annular cascade: Basic features of the E_1 (low–fidelity) and E_2 (high–fidelity) problem–specific evaluation software

	E_1	E_2
Turbulence model	Spalart–Allmaras with wall functions	Spalart–Allmaras, low–Reynolds model
Number of grid nodes	$\backsim 600k$	$\backsim 1.000k$
Number of tetrahedra	$\backsim 225k$	$\backsim 280k$
Number of pyramids	$\backsim 6k$	$\backsim 10k$
Number of prisms	$\backsim 205k$	$\backsim 300k$
Number of hexahedra	$\backsim 475k$	$\backsim 820k$
Average wall distance	$5 \times 10^{-5} m$	$3 \times 10^{-5} m$
Stretching close to wall (ω)	1.20	1.15
Blade surface grid	400×85	400×95
Viscous layers	21	27
Grid generation: Wall clock time	$\backsim 11 min$	$\backsim 18 min$
Flow solution: Wall clock time	$\backsim 1.2h$	$\backsim 6h$
CPU cost units	0.2	1

on E_1 and only the best of them on E_2. Once 100 previously evaluated individuals on E_1 were archived in the database, *RBF* networks were used in place of fitness inheritance. The hierarchical *EA* was stopped at 80 *CPU* cost units.

Fig. 3.12 (right) illustrates the convergence of the optimization algorithm. The mass–averaged *PLC* values of the reference and optimal airfoils were found equal to 0.1189 and 0.0843, respectively. The gain using the hierarchical algorithm is absolutely clear since, by merely using a *MAEA* based on E_2, it would be impossible to locate the optimal solution at the cost of 80 *CPU* cost units.

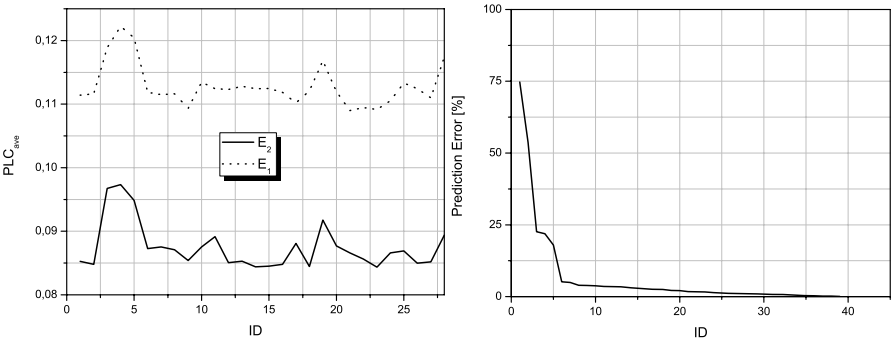


Fig. 3.13 Optimization of an annular cascade: Comparison of objective function values of individuals evaluated on both E_1 and E_2 (left) and *RBF* networks. For the latter, the relative prediction error with respect to E_1 are shown (right)

Table 3.2 Optimization of an annular cascade: Analysis of the overall *CPU* cost

	E_1	E_2
CPU cost units	33	47
Total evaluations	165	47
Failed evaluations	54	19
Infeasible evaluations	35	3

Table 3.2 summarizes statistics of the optimization run. An extremely low number of evaluations on E_2 (only 47) were carried out. Recall that any candidate solution evaluated on E_2 should have previously been evaluated on E_1; the two objective function values these individuals take on by the two models are compared in fig. 3.13. However, 19 of them were assigned a "death penalty" (i.e. an almost infinite objective function value) since E_2 failed to converge. So, fig. 3.13 compares the performance of the 28 remaining individuals, only. The average deviation of PLC_{ave} predictions on E_1 and E_2 (for these 28 individuals) was about 29%. The same figure presents the prediction error of the *RBF* networks with respect to E_1 (rank sorted). The metamodel prediction accuracy seems very satisfactory since, 34 out of the 39 approximate predictions had a relative error less than 5%.

In fig. 3.14, iso–contours of the total pressure loss coefficient $C_{pt} = \frac{\bar{p}_{t,inl} - p_t}{\bar{p}_{t,inl} - \bar{p}_{inl}}$ are plotted on two transversal cross–sections located $0.782C_{ax}$ and $1.145C_{ax}$ downstream of the blade leading edge, for both reference and optimal blades. It is clearly shown that, with the redesigned blade, the losses induced by the tip clearance vortex are much lower and this causes a much lower PLC_{ave} value.

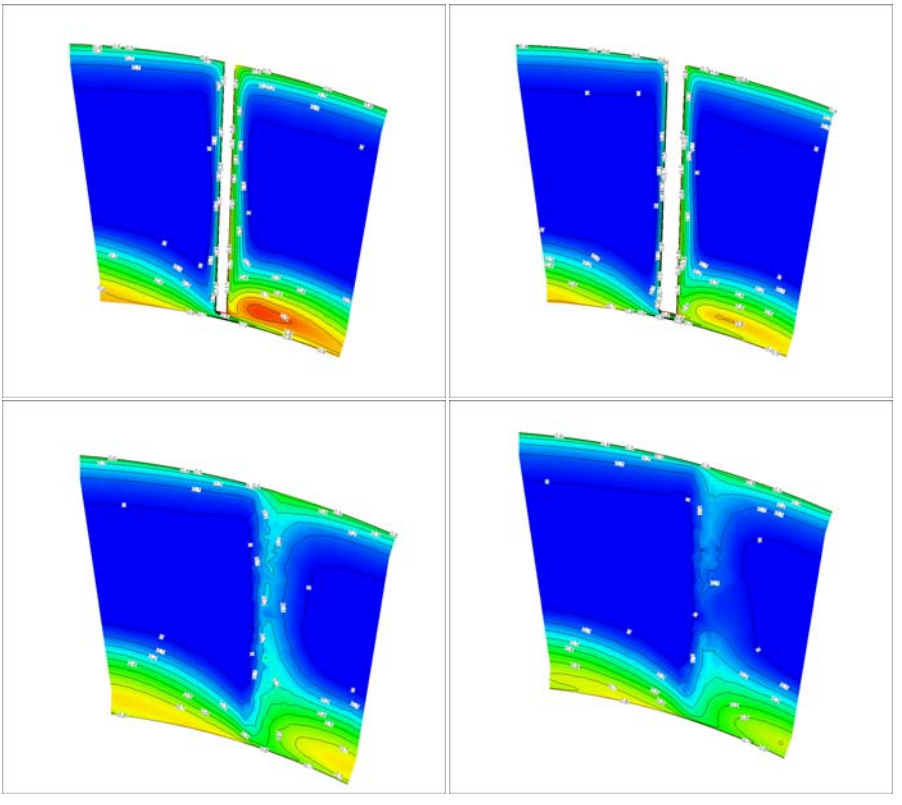


Fig. 3.14 Optimization of an annular cascade: Total pressure loss fields transversal cross–sections located at $0.782C_{ax}$ (top) and $1.145C_{ax}$ (bottom) downstream of the blade leading edge. Reference (left) and optimal (right) blade are included

3.8 Conclusions

Several algorithms capable to reduce the number of evaluations and the wall clock time of *EA*–based optimization were presented. They were based on:

- *Metamodels*, which replace as many calls to the problem–specific evaluation model as possible. *RBF* networks have been used, assisted by fitness inheritance during the first few generations. However, any other artificial neural network, such as a multilayer perceptron, Gaussian processes or even polynomial regression, could have been used instead, [18, 19, 25].
- *Distributed search*, in the form of intercommunicating demes, being advantageous if, particularly, the design is carried out on multiprocessor platforms.
- *Hierarchical schemes*, splitting the search into levels based on evaluation tools of different fidelity and *CPU* cost, different coarse and fine parameterizations or involving gradient–based search for refinement.

Their combination is possible and very efficient indeed. It can be performed in various ways. In this chapter we presented hierarchical distributed (where the levels are independently structured in demes) and distributed hierarchical search (where the hierarchy is implicit to each deme). It was beyond the scope of this chapter to compare the two aforementioned schemes; from several tests, it has been seen that such a conclusion is case dependent. However, it was clearly demonstrated that the combination of the above schemes leads to a considerable economy in the *CPU* cost. Over and above, in this work, an *EA* served as the base search method. The presented hierarchical framework as well as the *IPE* technique is directly extended to any other stochastic search technique, such as evolution strategies with covariant matrix adaptation [3] or particle swarm optimization [51].

References

1. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. IEEE Trans. on Evolutionary Computation 6(5) (2002)
2. Asouti, V., Zymaris, A., Papadimitriou, D., Giannakoglou, K.: Continuous and discrete adjoint approaches for aerodynamic shape optimization with low Mach number preconditioning. Int. J. for Numerical Methods in Fluids 57(10), 1485–1504 (2008)
3. Auger, A., Hansen, N.: A restart cma evolution strategy with increasing population size. In: CEC 2005, UK, vol. 2, pp. 1769–1776 (2005)
4. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Evolution Strategies. Evolutionary Programming, Genetic Algorithms. Oxford University Press, Oxford (1996)
5. Benoudjit, N., Archambeau, C., Lendasse, A., Lee, J., Verleysen, M.: Width optimization of the Gaussian kernels in radial basis function networks. In: ESANN 2002, Bruges, pp. 425–432 (2002)
6. Branke, J., Schmidt, C.: Faster convergence by means of fitness estimation. Soft Computing — A Fusion of Foundations, Methodologies & Applications 9(1), 13–20 (2005)
7. Büche, D., Schraudolph, N., Koumoutsakos, P.: Accelerating evolutionary algorithms with Gaussian process fitness function models. Trans. on Systems, Man & Cybernetics — Part C: Applications & Reviews 35(2), 183–194 (2005)
8. Bull, L.: On model-based evolutionary computation. Soft Computing — A Fusion of Foundations, Methodologies & Applications 3(2), 76–82 (1999)
9. Cantu-Paz, E.: A survey of parallel genetic algorithms. Calculateurs Paralleles, Reseaux et Systemes Repartis 10(2), 141–171 (1998)
10. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
11. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. Trans. on Evolutionary Computation 6(2), 182–197 (2002)
12. Désidéri, J., Janka, A.: Hierarchical parameterization for multilevel evolutionary shape optimization with application to aerodynamics. In: EUROGEN 2003, Barcelona (2003)
13. Doorly, D.J., Peiró, J.: Supervised parallel genetic algorithms in aerodynamic optimisation. AIAA Paper 1997-1852 (1997)
14. Drela, M., Giles, M.: Viscous-inviscid analysis of transonic and low Reynolds number airfoils. AIAA J. 25(10), 1347–1355 (1987)

15. Ducheyne, E., De Baets, B., De Wulf, R.: Fitness inheritance in multiple objective evolutionary algorithms: A test bench and real-world evaluation. Applied Soft Computing 8(1), 337–349 (2008)
16. Duvigneau, R., Chaigne, B., Désidéri, J.: Multi-level parameterization for shape optimization in aerodynamics and electromagnetics using a particle swarm optimization algorithm. Tech. Rep. RR-6003, INRIA, France (2006)
17. Eby, D., Averill, R., Punch III, W., Goodman, E.: Evaluation of injection island GA performance on flywheel design optimization. In: Proceedings of the 3rd Conf. on Adaptive Computing in Design & Manufacturing, pp. 121–136. Springer, Heidelberg (1998)
18. Emmerich, M.T.M., Giotis, A., Özdemir, M., Bäck, T., Giannakoglou, K.: Metamodel-assisted evolution strategies. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 361–370. Springer, Heidelberg (2002)
19. Emmerich, M., Giannakoglou, K., Naujoks, B.: Single- and multi-objective evolutionary optimization assisted by Gaussian random field metamodels. IEEE Trans. on Evolutionary Computation 10(4), 421–439 (2006)
20. Farina, M.: A neural network based generalized response surface multiobjective evolutionary algorithm. In: CEC 2002, Honolulu, HI, vol. 1, pp. 956–961 (2002)
21. Foster, I.: Globus toolkit version 4: Software for service-oriented systems. In: Jin, H., Reed, D., Jiang, W. (eds.) NPC 2005. LNCS, vol. 3779, pp. 2–13. Springer, Heidelberg (2005)
22. Fritzke, B.: Fast learning with incremental RBF networks. Neural Processing Letters, 2–5 (1994)
23. Fritzke, B.: Growing cell structures — A self-organizing network for unsupervised and supervised learning. Neural Networks 7(9), 1441–1460 (1994)
24. Georgopoulou, C., Giannakoglou, K.: A multi-objective metamodel-assisted memetic algorithm with strength-based local refinement. Engineering Optimization Accepted for publication (to appear, 2009)
25. Giannakoglou, K.: Designing turbomachinery blades using evolutionary methods. ASME Paper 99-GT-181 (1999)
26. Giannakoglou, K.: Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. Progress in Aerospace Sciences 38(1), 43–76 (2002)
27. Giannakoglou, K.: The EASY (Evolutionary Algorithms System) software (2008), http://velos0.ltt.mech.ntua.gr/EASY
28. Giannakoglou, K., Georgopoulou, C.: Multiobjective metamodel-assisted memetic algorithms. In: Multiobjective Memetic Algorithms. Studies in Computational Intelligence. Springer, Heidelberg (2009)
29. Giannakoglou, K., Giotis, A., Karakasis, M.: Low-cost genetic optimization based on inexact pre-evaluations and the sensitivity analysis of design parameters. Inverse Problems in Engineering 9, 389–412 (2001)
30. Giannakoglou, K., Papadimitriou, D., Kampolis, I.: Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodels. Computer Methods in Applied Mechanics & Engineering 195, 6312–6329 (2006)
31. Giannakoglou, K., Kampolis, I., Georgopoulou, C.: Metamodel-assisted evolutionary algorithms (MAEAs). In: Introduction to Optimization and Multidisciplinary Design in Aeronautics and Turbomachinery, Lecture Series, von Karman Institute, Rhodes–Saint Genése (2008)

32. Giotis, A., Giannakoglou, K.: Single- and multi-objective airfoil design using genetic algorithms and artificial intelligence. In: EUROGEN 1999, Jyväskylä (1999)
33. Giotis, A., Giannakoglou, K., Périaux, J.: A reduced-cost multi-objective optimization method based on the Pareto front technique, neural networks and PVM. In: ECCOMAS 2000, Barcelona (2000)
34. Goldberg, D.: Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley, Reading (1989)
35. Greenman, R., Roth, K.: Minimizing computational data requirements for multi-element airfoils using neural networks. AIAA Paper 1999-0258 (1999)
36. Haykin, S.: Neural Networks - A Comprehensive Foundation, 2nd edn. Prentice Hall, Englewood Cliffs (1999)
37. Herrera, F., Lozano, M., Moraga, C.: Hierarchical distributed genetic algorithms. Int. J. of Intelligent Systems 14(9), 1099–1121 (1999)
38. Jin, Y., Olhofer, M., Sendhoff, B.: A framework for evolutionary optimization with approximate fitness functions. IEEE Trans. on Evolutionary Computation 6(5), 481–494 (2002)
39. Kampolis, I., Giannakoglou, K.: A multilevel approach to single- and multiobjective aerodynamic optimization. Computer Methods in Applied Mechanics & Engineering 197, 2963–2975 (2008)
40. Kampolis, I., Giannakoglou, K.: Distributed evolutionary algorithms with hierarchical evaluation. Engineering Optimization Accepted for publication (to appear, 2009)
41. Kampolis, I., Karangelos, E., Giannakoglou, K.: Gradient-assisted radial basis function networks: theory and applications. Applied Mathematical Modelling 28(13), 197–209 (2004)
42. Kampolis, I., Papadimitriou, D., Giannakoglou, K.: Evolutionary optimization using a new radial basis function network and the adjoint formulation. Inverse Problems in Science & Engineering 14(4), 397–410 (2006)
43. Kampolis, I., Zymaris, A., Asouti, V., Giannakoglou, K.: Multilevel optimization strategies based on metamodel-assisted evolutionary algorithms, for computationally expensive problems. In: CEC 2007, Singapore (2007)
44. Karakasis, M., Giannakoglou, K.: On the use of metamodel-assisted, multi-objective evolutionary algorithms. Engineering Optimization 38(8), 941–957 (2006)
45. Karakasis, M., Giotis, A., Giannakoglou, K.: Inexact information aided, low-cost, distributed genetic algorithms for aerodynamic shape optimization. Int. J. for Numerical Methods in Fluids 43(10-11), 1149–1166 (2003)
46. Karakasis, M., Koubogiannis, D., Giannakoglou, K.: Hierarchical distributed evolutionary algorithms in shape optimization. Int. J. for Numerical Methods in Fluids 53(3), 455–469 (2007)
47. Karayiannis, N., Mi, G.: Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques. IEEE Trans. on Neural Networks 8(6), 1492–1506 (1997)
48. Keane, A., Nair, P.: Computational Approaches for Aerospace Design – The Pursuit of Excellence. John Wiley & Sons, Ltd., Chichester (2005)
49. Knowles, J., Corne, D.: M-PAES: A memetic algorithm for multiobjective optimization. In: CEC 2000, pp. 325–332. IEEE Press, Los Alamitos (2000)
50. Lambropoulos, N., Koubogiannis, D., Giannakoglou, K.: Acceleration of a Navier-Stokes equation solver for unstructured grids using agglomeration multigrid and parallel processing. Computer Methods in Applied Mechanics & Engineering 193, 781–803 (2004)

51. Langdo, W., Poli, R.: Evolving problems to learn about particle swarm and other optimisers. In: CEC 2005, UK, pp. 81–88 (2005)
52. Liakopoulos, P., Kampolis, I., Giannakoglou, K.: Grid-enabled, hierarchical distributed metamodel-assisted evolutionary algorithms for aerodynamic shape optimization. Future Generation Computer Systems 24, 701–708 (2008)
53. Lim, D., Ong, Y.S., Jin, Y., Sendhoff, B., Lee, B.S.: Efficient hierarchical parallel genetic algorithms using grid computing. Future Generation Computer Systems 23(4), 658–670 (2007)
54. Lin, S.C., Punch, W., Goodman, E.: Coarse-grain parallel genetic algorithms: categorization and new approach. In: 6th IEEE Symposium on Parallel & Distributed Processing, Dallas, pp. 28–37 (1994)
55. Massie, M., Chun, B., Culler, D.: The Ganglia distributed monitoring system: Design, implementation, and experience. Parallel Computing 30(7) (2004)
56. Mathioudakis, K., Papailiou, K., Neris, N., Bonhommet, C., Albrand, G., Wenger, U.: An annular cascade facility for studying tip clearance effects in high speed flows. In: XIII ISABE Conf., Chattanooga, TN (1997)
57. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Heidelberg (1996)
58. Michalewicz, Z., Fogel, D.: How to Solve it: Modern Heuristics, 2nd edn. Springer, Heidelberg (2004)
59. Montero, R., Huedo, E., Llorente, I.: A framework for adaptive execution on grids. J. of Software - Practice & Experience 34, 631–651 (2004)
60. Moody, J., Darken, C.: Fast learning in networks of locally-tuned processing units. Neural Computation 1(2), 281–294 (1989)
61. Muyl, F., Dumas, L., Herbert, V.: Hybrid method for aerodynamic shape optimization in automotive industry. Computers & Fluids 33(5-6), 849–858 (2004)
62. Nakayama, H., Inoue, K., Yoshimori, Y.: Approximate optimization using computational intelligence and its application to reinforcement of cable-stayed bridges. In: ECCOMAS 2004, Jyväskylä (2004)
63. Nocedal, J., Wright, S.: Numerical Optimization. Springer, Heidelberg (1999)
64. Nowostawski, M., Poli, R.: Parallel genetic algorithm taxonomy. In: KES 1999, pp. 88–92 (1999)
65. Ong, Y., Lum, K., Nair, P.: Hybrid evolutionary algorithm with Hermite radial basis function interpolants for computationally expensive adjoint solvers. Computational Optimization & Applications 39(1), 97–119 (2008)
66. Ong, Y.S., Lum, K.Y., Nair, P., Shi, D., Zhang, Z.K.: Global convergence of unconstrained and bound constrained surrogate-assisted evolutionary search in aerodynamic shape design. In: CEC 2003, Canberra, vol. 3, pp. 1856–1863 (2003)
67. Ong, Y.S., Nair, P., Keane, A.: Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA J. 41(4), 687–696 (2003)
68. Ong, Y.S., Lim, M., Zhu, N., Wong, K.: Classification of adaptive memetic algorithms: A comparative study. IEEE Trans. on Systems Man & Cybernetics - Part B 36, 141–152 (2006)
69. Papadimitriou, D., Giannakoglou, K.: A continuous adjoint method with objective function derivatives based on boundary integrals for inviscid and viscous flows. Computers & Fluids 36, 325–341 (2007)
70. Papadimitriou, D., Giannakoglou, K.: Total pressure loss minimization in turbomachinery cascades using a new continuous adjoint formulation. J. of Power & Energy (Part A) 221, 865–872 (2007)

71. Papadrakakis, M., Lagaros, N.D., Tsompanakis, Y.: Structural optimization using evolution strategies and neural networks. Computer Methods in Applied Mechanics & Engineering 156(1-4), 309–333 (1998)
72. Piegl, L., Tiller, W.: The NURBS Book, 2nd edn. Springer, Heidelberg (1997)
73. Pierret, S., Van den Braembussche, R.: Turbomachinery blade design using a Navier-Stokes solver and artificial neural network. ASME J. of Turbomachinery 121(2), 326–332 (1999)
74. Poggio, T., Girosi, F.: Networks for approximation and learning. Proceedings of the IEEE 78(9), 1481–1497 (1990)
75. Politis, E., Giannakoglou, K., Papailiou, K.: High–speed flow in an annular cascade with tip clearance: Numerical investigation. ASME Paper 98-GT-247 (1998)
76. Poloni, C., Giurgevich, A., Onesti, L., Pediroda, V.: Hybridization of a multiobjective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. Computer Methods in Applied Mechanics & Engineering 186(2), 403–420 (2000)
77. Ratle, A.: Optimal sampling strategies for learning a fitness model. In: CEC 1999, Washington, DC, vol. 3, pp. 2078–2085 (1999)
78. Sefrioui, M., Périaux, J.: A hierarchical genetic algorithm using multiple models for optimization. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 879–888. Springer, Heidelberg (2000)
79. Smith, R., Dike, B., Stegmann, S.: Fitness inheritance in genetic algorithms. In: SAC 1995, pp. 345–350. ACM, New York (1995)
80. Spalart, P., Allmaras, S.: A one-equation turbulence model for aerodynamic flows. AIAA Paper 92-0439 (1992)
81. Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: the Condor experience. Concurrency - Practice and Experience 17(2-4), 323–356 (2005)
82. Thévenin, D., Janiga, G.: Optimization and Computational Fluid Dynamics. Springer, Heidelberg (2008)
83. Ulmer, H., Streichert, F., Zell, A.: Evolution strategies assisted by Gaussian processes with improved pre-selection criterion. In: CEC 2003, Canberra, vol. 1, pp. 692–699 (2003)
84. Zhou, Z., Ong, Y.S., Lim, M., Lee, B.: Memetic algorithm using multi–surrogates for computational expensive optimization problems. Soft Computing 11(10), 957–971 (2007)
85. Zitzler, E., Laumans, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm. Tech. Rep. 103, ETH, Computer Engineering & Communication Networks Lab. (TIK), Zurich (2001)
86. Zitzler, E., Laumans, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In: EUROGEN 2001, CIMNE, Barcelona, pp. 19–26 (2001)
87. Zitzler, E., Brockhoff, D., Thiele, L.: The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 862–876. Springer, Heidelberg (2007)

Chapter 4
Knowledge-Based Variable-Fidelity Optimization of Expensive Objective Functions through Space Mapping

Slawomir Koziel and John W. Bandler

Abstract. The growing complexity of engineering modeling and design problems demands effective strategies for optimization of computationally expensive objective functions. To this end, we focus on knowledge-based, variable-fidelity optimization of expensive functions through a tried and tested, yet still rapidly evolving art called space mapping optimization. Fitting into the arena of surrogate-based optimization, space-mapping optimization is a model-driven optimization process where the model is an iteratively updated surrogate derived from a valid, low-fidelity or physics-based coarse model. Space mapping takes several forms. Here, we present and formulate the original input space mapping concept, as well as the more recent implicit and output space mapping concepts. Corresponding surrogate models are presented, classified, and discussed. A proposed optimization flow is explained. Then we illustrate both input space mapping and implicit space mapping through the space mapping optimization of a simple, technology-free wedge-cutting problem. We also present tuning space mapping, a powerful methodology, but one that requires extra engineering knowledge of the problem under investigation. To confirm our work, we select representative examples from the fields of microwave and antenna engineering, including filter and antenna designs.

4.1 Introduction

True of all branches of engineering, the escalating complexity of modeling and design problems drives today's demand for effective strategies for optimization

Slawomir Koziel
Engineering Optimization & Modeling Center, School of Science and Engineering, Reykjavik University, Kringlunni 1, IS-103, Reykjavik, Iceland
e-mail: koziel@ru.is

John W. Bandler
Simulation Optimization Systems Research Laboratory, Department of Electrical and Computer Engineering, McMaster University, 1280 Main Street West, L8S 4K1 Ontario, Canada
e-mail: bandler@mcmaster.ca

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 85–109.
springerlink.com

of computationally expensive objective functions. Challenging as the issue is for medium or large problems involving single disciplines, only the engineer's imagination limits the scope of possibilities for multidisciplinary optimization. It is timely, therefore, to consider methodologies capable of spanning many fields of design optimization endeavour.

This chapter focuses on knowledge-based variable-fidelity optimization of expensive functions through space mapping. The work fits into the arena of surrogate-based optimization (SBO). The work-horse for our specific approach to SBO is a surrogate model derived from a valid, low-fidelity or physics-based coarse model. Occasional and appropriate recourse to the problem's high-fidelity model keeps our surrogate updated. Often the coarse model is a lower fidelity version of the high-fidelity or fine model. In any case, the cost of computing the surrogate is assumed to be little more than that of any underlying coarse model.

In the following paragraphs, we introduce some important contributions to the field from a literature rich with contributions. Our brief review is intended to be representative and influential rather than exhaustive.

Space mapping procedures iteratively update and optimize surrogates based on fast physically based "coarse" models. Early approaches include the original algorithm by Bandler et al. [9], and the Broyden-based aggressive space mapping algorithm by Bandler et al. [10]. Dennis and Torczon [18] exploit trust-region techniques and pattern search methods to manage what they call the interplay between the optimization method and the fidelity of the approximation model. Alexandrov et al. [2] present an approach to managing the use of approximation models of various fidelity. Their approach is based on the idea of trust regions from nonlinear programming. They demonstrate convergence to a solution of the original high-fidelity problem. Their method suggests ways of deciding when the fidelity, hence cost, of the approximations can be increased or decreased during the optimization iterations. More recently, Alexandrov et al. [3] provide a rigorous methodology for solving high-fidelity optimization problems using derivative-based optimization algorithms and any combination of high-fidelity and low-fidelity models. The paper considers both variable-resolution models and variable-fidelity physics models.

A benchmark paper by Booker et al. [14] treats problems for which traditional optimization approaches are not practical. It offers a framework for generating a sequence of approximations to an expensive objective function and managing the use of these approximations as surrogates. The authors' approach does not require or involve derivatives of the objective function. Simpson et al. [45] review the response surface method then present kriging as an alternative approximation method for the design and analysis of computer experiments. They apply both methods to a multidisciplinary design problem involving a computational fluid dynamics model and a finite-element model. Marsden et al. [36] apply shape optimization to time-dependent trailing-edge flow so as to minimize aerodynamic noise. They use the surrogate management framework (SMF), a non-gradient based pattern search method, to explore the design space with an inexpensive surrogate function.

Within the sphere of space mapping, Robinson et al. [44] treat design problems when the coarse and fine models are defined over different design spaces and mappings are required over these spaces.

In a related field, Rayas-Sánchez [41] reviews the state-of-the-art in electromagnetics-based design and optimization using artificial neural networks. He surveys conventional modeling approaches along with typical enhancing and knowledge-based techniques. Rayas-Sánchez reviews strategies for design exploiting knowledge, including neural space-mapping methods.

Space mapping technology emerged in 1994 [9] out of competitive necessity. Full-wave electromagnetic solvers had long been accepted for validating microwave designs obtained through equivalent circuit models. While the idea of employing electromagnetic solvers for direct optimal design attracted microwave engineers, electromagnetic solvers are notoriously CPU-intensive. As originally construed, they also suffered from non-differentiable response evaluation and non-parameterized design variables that were often discrete in the parameter space, etc. Such characteristics are unfriendly to classical gradient optimization algorithms. Thus, state-of-the-art successful interconnection of electromagnetic solvers with powerful optimization techniques still insufficiently addressed the microwave community's ambitions for automated electromagnetics-based design optimization.

The original idea of space mapping [9] was to map designs from optimized circuit models to corresponding electromagnetic models. A "parameter extraction" step calibrated the circuit solver against the electromagnetic simulator in order to minimize observed discrepancies between the two simulations. The circuit model (surrogate) was then updated through extracted parameters and made ready for subsequent classical optimization.

Bandler et al. [11] reviewed the space mapping and the space-mapping-based surrogate modeling concepts and applications in various engineering design optimization problems. They present a mathematical motivation and place space mapping into the context of classical optimization. Recent work in space mapping includes a trust-region approach [5], neural space mapping [6] and implicit space mapping [12]. Parameter extraction is an essential sub-problem used to align the surrogate—an enhanced coarse model—with the fine model. In a 2006 review, Bandler et al. [13] show that all the existing space mapping approaches can be viewed as particular cases of one, generic formulation of space mapping.

Space mapping demonstrably addresses the engineer's need for validated, high-fidelity designs when classical optimization algorithms threaten hundreds of costly simulations, and perhaps days or weeks of CPU time. The methodology exploits underlying fast-to-compute, low-fidelity surrogate models, which are ubiquitous in engineering practice. Space mapping takes the high-fidelity simulator out of the classical optimization loop, instead exploiting the iterative enhancement of the available low-fidelity surrogates. Space mapping optimization algorithms enjoy a desirable feature: they usually provide excellent designs after only a handful of high-fidelity simulations. The methodology follows the traditional experience and intuition of the engineer, yet is amenable to mathematical treatment. It enjoys immediate recognition by the experienced engineering designer.

A key to the success of space mapping, i.e., that it yields satisfactory solutions after a few fine model evaluations, is the recommended physical nature of the coarse model. Other surrogate-model-based methods [14, 18, 24, 39, 46] exploit functional surrogates obtained from direct approximation of the available fine model data and, therefore, cannot compete with space mapping in terms of computational efficiency.

In "implicit" space mapping, preassigned parameters not used in the optimization process can change in the coarse model. In "output" space mapping, we transform the response of the coarse model. Other exciting developments include surrogates that interpolate fine models simulated on a structured grid, frequency mappings; and the recent concept of "tuning" space mapping. The latest review by Koziel et al. [32] places various related concepts contextually into the history of design optimization and modeling of microwave circuits.

Space mapping optimization [21, 29] belongs to the class of surrogate-based optimization methods [14] that generates a sequence of approximations to the objective function and manages the use of these approximations as surrogates for optimization.

Space mapping methodology continues to provide success in diverse areas [4, 17, 20, 22, 25, 27, 28, 33, 42, 43, 49, 50, 51]: electronic, photonic, radio frequency, antenna, microwave, and magnetic systems; civil, mechanical, and aerospace engineering structures, including automotive crashworthiness design [43].

In this chapter, we present and formulate the original input space mapping concept, as well as the more recent implicit and output space mapping concepts. We present, classify and discuss corresponding surrogate models. A proposed optimization flow is explained. Then we illustrate both input space mapping and implicit space mapping through the space mapping optimization of a simple wedge-cutting problem. We also present tuning space mapping, a powerful methodology, but one that requires extra engineering knowledge of the problem under investigation. Throughout, we select representative examples from the fields of microwave and antenna engineering, including filter and antenna designs.

4.2 Space Mapping Optimization

In this section we formulate the space mapping (SM) optimization algorithm, discuss some popular SM approaches, and provide a simple example that explains the operation of space mapping. Some practical issues of SM optimization as well as desirable features of the models involved in the SM process are also indicated.

4.2.1 Formulation of the Space Mapping Algorithm

Let $f : \Omega_f \rightarrow \Re^m$, $\Omega_f \subseteq \Re^n$, denote the high-fidelity (or fine) model of the engineering device. The goal is to solve

$$x_f^* \in \arg\min_{x \in \Omega_f} H(f(x)) \tag{4.1}$$

where $H : \Re^m \to \Re$ is a given merit function, e.g., a norm. $H \circ f$ is the objective function.

We consider the fine model to be expensive to compute and solving (4.1) by direct optimization to be impractical. Instead, we use surrogate models, i.e., models that are not as accurate as the fine model but are computationally cheap, hence suitable for iterative optimization. We consider a general optimization algorithm that generates a sequence of points $x^{(i)} \in \Omega_f$, $i = 1, 2, \ldots$, and a family of surrogate models $s^{(i)} : \Omega_s^{(i)} \to \Re^m$, $i = 0, 1, \ldots$, so that

$$x^{(i+1)} \in \arg \min_{x \in \Omega_f \cap \Omega_s^{(i)}} H(s^{(i)}(x)) \tag{4.2}$$

If the solution to (4.2) is non-unique we may impose regularization.

Space mapping (SM) assumes the existence of a so called coarse model, $c : \Omega_c \to \Re^m$, $\Omega_c \subseteq \Re^n$, that describes the same object as the fine model: less accurate but much faster to evaluate. The family of surrogate models is constructed from the coarse model in such a way that $s^{(i)}$ is a suitable distortion of c, such that the misalignment between the fine and the surrogate models is reduced as much as possible (cf. (4.4)).

Let $\bar{s} : \Omega_s \to \Re^m$ be a generic SM surrogate model which is the coarse model composed with some suitable space mapping transformations, where $\Omega_s \subset \Omega_c \times \Omega_p$, with Ω_p being the parameter space of these transformations. We call Ω_p a SM parameter domain. The surrogate model $s^{(i)}$ is defined as

$$s^{(i)}(x) = \bar{s}(x, p^{(i)}) \tag{4.3}$$

where

$$p^{(i)} \in \arg \min_{p \in \Omega_p^{(i)}} \left(\sum_{k=0}^{i} w_{i.k} ||f(x^{(k)}) - \bar{s}(x^{(k)}, p)|| \right) \tag{4.4}$$

where $\Omega_p^{(i)} = \{p \in \Omega_p : (x^{(k)}, p) \in \Omega_s \quad \text{for} \quad k = 0, 1, \ldots, i\}$ and $w_{i.k}$ are weighting factors. Two typical weight settings are: (i) $w_{i.k} = 1$ for $k = 0, 1, \ldots, i$ (all points $x^{(k)}$, $k = 0, 1, \ldots, i$, have the same contribution to the parameter extraction process) and (ii) $w_{i.k} = 1$ for $i = k$, and $w_{i.k} = 0$ otherwise (only the last iteration point is used in (4.4)). The domain $\Omega_s^{(i)}$ of the surrogate model $s^{(i)}$ is $\Omega_s^{(i)} = \{x \in \Omega_c : (x, p^{(i)}) \in \Omega_s\}$.

The space-mapping optimization algorithm flow can be described as follows:

1. Set $i = 0$; choose the initial solution $x^{(0)}$;
2. Evaluate the fine model to find $f(x^{(i)})$;
3. Obtain the surrogate model $s^{(i)}$ using (4.3) and (4.4);
4. Given $x^{(i)}$ and $s^{(i)}$, obtain $x^{(i+1)}$ using (4.2);
5. If the termination condition is not satisfied go to 2; else terminate the algorithm.

Typically, $x^{(0)} = \arg\min\{x : H(c(x))\}$, i.e., it is the optimal solution of the coarse model, which is the best initial design we normally have at our disposal.

Usually, the algorithm is terminated when it converges (i.e., $||x^{(i)} - x^{(i-1)}||$ and/or $||f(x^{(i)}) - f(x^{(i-1)})||$ are smaller than user-defined values) or when the maximum number of iterations (or fine model evaluations) is exceeded.

4.2.2 Space Mapping Surrogate Models

There is a variety of space mapping surrogate models available [9, 10, 11, 12, 13, 29, 32]. They can be roughly categorized into four groups (here, n is the number of the design variables, m is the number of components of the high- and low-fidelity model response vectors):

- Models based on a (usually linear) distortion of the coarse model parameter space, e.g., input space mapping of the form $\bar{s}(x,p) = \bar{s}(x,B,q) = c(B \cdot x + q)$ [11], where B is an $n \times n$ matrix and q is an $n \times 1$ vector;
- Models based on distortion of the coarse model response, e.g., output space mapping of the form $\bar{s}(x,p) = \bar{s}(x,d) = c(x) + d$ (additive output SM; d is an $m \times 1$ vector) or $\bar{s}(x,p) = \bar{s}(x,a) = a \cdot c(x)$ (multiplicative output SM; a is usually a diagonal $m \times m$ matrix) [11, 29];
- Implicit space mapping, where the parameters used to align the surrogate with the fine model are separate from the design variables, i.e., $\bar{s}(x,p) = \bar{s}(x,x_p) = c_i(x,x_p)$, with c_i being the coarse model dependent on both design variables x and so-called preassigned parameters x_p (e.g., electric permittivity and the height of the dielectric substrate of a microstrip device [12]) that are normally fixed in the fine model but can be freely changed in the coarse model [12, 32];
- Custom models exploiting problem-specific parameters. For example, in microwave engineering, components of the coarse model vector $c(x)$ depend on a certain free parameter, usually the frequency ω of the input signal. In such a case we have $c(x) = [\bar{c}(x,\omega_1) \ \ldots \ \bar{c}(x,\omega_m)]^T$, where $\bar{c}$ is the frequency-dependent model and ω_j, $j = 1,\ldots,m$, are frequency samples at which the model is evaluated. Often, the fine and coarse model responses considered as functions of frequency have a similar "shape" and so-called frequency SM [11, 32] can be useful to reduce their misalignment. The surrogate model has the form of $\bar{s}(x,p) = \bar{s}(x,F) = c_f(x,F) = [\bar{c}(x,f_s(\omega_1,F)) \ \ldots \ \bar{c}(x,f_s(\omega_m,F))]^T$, where the scaling function f_s is usually defined as $f_s(\omega,F) = f_s(\omega,f_1,f_2) = f_1 + f_2\omega$, i.e., $F = [f_1 \ f_2]^T$.

It is a common practice that basic space mapping types are combined together, e.g., the model using both input, output and frequency space mapping would as follows: $\bar{s}(x,p) = \bar{s}(x,B,q,d,F) = c_f(B \cdot x + q, F) + d$. The rationale behind it is that a properly chosen mapping may significantly improve the performance of the space mapping algorithm, however, the optimal selection of the mapping type for a given design problem is not a trivial task [32].

4.2.3 Characterization of Space Mapping

It is critical to the performance of the SM algorithm that the coarse model is physically-based, i.e., it describes the same physical phenomena as the fine model but is not necessarily as accurate as f [32]. On the other hand, the coarse model should be computationally much cheaper than the fine model because both parameter extraction (4.4) and surrogate optimization (4.2) involve multiple evaluations of the coarse model. Having this in mind, SM can be considered as both a variable-fidelity and knowledge-based approach because c is a low-fidelity representation of f and encodes—sometimes substantial—knowledge about the latter. In the microwave engineering area, where space mapping originated, the fine model is usually evaluated using CPU intensive full-wave electromagnetic simulators, while the coarse model is a circuit-equivalent of the microwave structure in question or is based on analytical formulas [11, 32].

If the surrogate model is a sufficiently good representation of the fine model [29], the space-mapping algorithm (4.2) is likely to produce a sequence of vectors $x^{(i)}$ that quickly approach a satisfactory design, however, we cannot expect the final solution to be a local optimum of the fine model in general, unless, for example, first-order consistency conditions between the surrogate and the fine model are ensured (which requires exploiting fine model sensitivity data [29]) and convergence safeguards such as trust region methods are used [16].

Usually, the fine model is only evaluated once per iteration (at every new design $x^{(i+1)}$) for verification purposes and to obtain the data necessary to update the surrogate model. Because of the low computational cost of the surrogate model, its optimization cost (cf. (4.2)) as well as the cost of parameter extraction (4.4) can usually be neglected and the total optimization cost is determined by the evaluation of f. The key point here is that the number of evaluations of the fine model for a well performing surrogate-based algorithm is substantially smaller than for any direct optimization method (e.g., gradient-based) [29]. A typical number of fine model evaluations reported in the literature (e.g., [4, 5, 9, 10, 11, 12, 32]) for space mapping algorithms range between three and ten and is hardly dependent on the problem size n.

It should be noted that a similar idea is shared by other surrogate-based optimization (SBO) methods [2, 3, 14, 18, 36, 45], however, many of them do not use a simplified physically based coarse model: a functional surrogate is created by direct approximation of the available fine model data. There is a large group of functional approximation techniques that can be used to create fast surrogate models, including radial basis functions [15], kriging [48], fuzzy systems [38], and neural networks [19]. In order to achieve reasonable accuracy, all of these methods require, however, a large amount of data. Moreover, the number of data pairs necessary to ensure sufficient accuracy grows exponentially with the number of design variables. Therefore, SBO techniques using functional surrogate models cannot achieve space mapping efficiency in terms of the number of high-fidelity model evaluations required to find satisfactory solutions.

4.2.4 Practical Issues and Open Problems

It is easy to notice that the space mapping algorithm (4.2)-(4.4) has the following two features. First of all, consistency conditions between the fine and surrogate models are not necessarily satisfied. In particular, it is not required that the surrogate model matches the fine model with respect to value and first-order derivative at any of the iteration points. Second, subsequent iterations are accepted regardless of the objective function improvement. As a consequence, convergence of the SM algorithm is not guaranteed in general, and the choice of an optimal space mapping approach for a given problem is not obvious [31].

Several methods for assessing the quality of the coarse/surrogate model have been proposed that are based on information obtained from the fine model at a set of test points [30, 31]. This information is used to estimate certain conditions in the convergence results and allows us to predict whether a given model might be successfully used in space mapping optimization. Using these methods one can also compare the quality of different coarse models, or choose the proper type of space mapping which would suit a given optimization problem [31]. Assessment methods can also be embedded into the space mapping optimization algorithm so that the most suitable surrogate model is selected in each iteration of the algorithm out of a given candidate model set [30].

Convergence properties of the space mapping algorithm can be improved by using the trust region method [16, 32], in which the surrogate model optimization is restricted to a neighborhood of the point $x^{(i)}$ so that we have

$$x^{(i+1)} \in \arg \min_{x \in \Omega_f \cap \Omega_s^{(i)}, ||x-x^{(i)}|| \le \delta^{(i)}} H(s^{(i)}(x)) \tag{4.5}$$

where $\delta^{(i)}$ denotes the trust region radius at iteration i, which is updated at every iteration using classical rules [16].

It should be emphasized that space mapping is not a general-purpose approach. The existence of the computationally cheap and sufficiently accurate coarse model is an important prerequisite of our technique. If such a coarse model does exist, the space mapping method is able to yield a satisfactory design after a few evaluations of the high-fidelity model, which is a dramatic reduction of the computational cost of the optimization compared to other methods. Otherwise, space mapping cannot be used or will not be efficient.

4.2.5 Space Mapping Illustration

In order to illustrate the operation of basic space mapping algorithm we consider the so-called wedge-cutting problem [11] formulated as follows. Given a wedge as shown in Fig. 4.1(a), cut a piece of length x so that the corresponding area, $f(x)$ is equal to $f_0 = 100$. Our fine model is f, which, for the sake of example, is assumed to be given by $f(x) = x(5 + x/16)$. We also consider a simplified representation of the wedge, a rectangle of height H shown in Fig. 4.1(b). The coarse model is the area

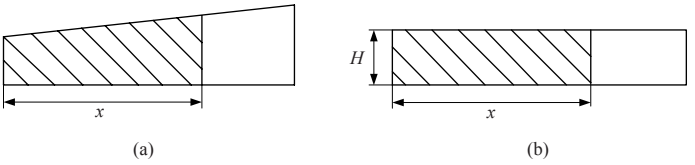


Fig. 4.1 Wedge-cutting problem [11]: (a) the fine model, and (b) the coarse model

of a piece of this rectangle, determined by the length x, so that we have $c(x) = Hx$. Here, we assume that $H = 5$.

The starting point of SM optimization is a coarse model optimal solution $x^{(0)} = 20$. The fine model at $x^{(0)}$ is $f(x^{(0)}) = 125$. For illustration purposes we will solve our problem using the simplest version of the input space mapping and then using implicit space mapping.

4.2.5.1 Wedge Cutting Problem Solved Using Input Space Mapping

We use the following setup for the input space mapping approach. The generic surrogate model is given by $\bar{s}(x,p) = \bar{s}(x,q) = c(x+q)$. The weighting factors in the parameter extraction process (4.4) are given by $w_{i.k} = 1$ for $k = i$ and $w_{i.k} = 0$ otherwise. Thus, the surrogate model can be written in short as:

$$s^{(i)}(x) = c(x+q^{(i)}) \tag{4.6}$$

where

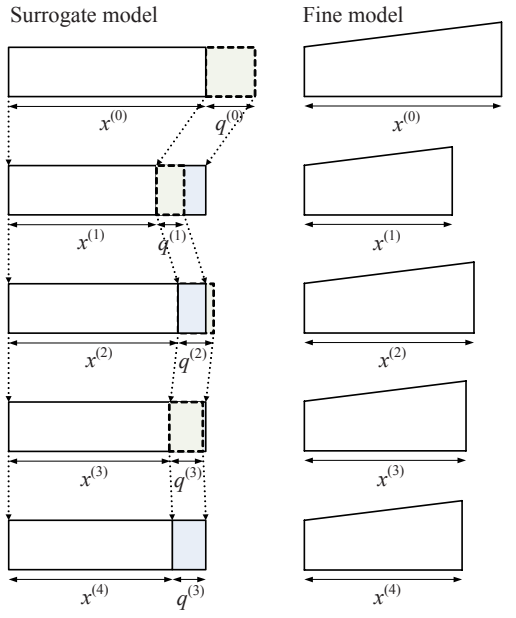
$$q^{(i)} = \arg\min_{q} ||f(x^{(i)}) - c(x^{(i)}+q)|| \tag{4.7}$$

In this simple case, (4.7) has an analytical solution given by $q^{(i)} = f(x^{(i)})/H - x^{(i)}$. Figure 4.2 shows the first four iterations of the SM algorithm solving the wedge cutting problem. This particular input space mapping approach is both simple and direct, yet it converges to an acceptable result (from an engineering point of view) in a remarkably small number of iterations. It is clearly knowledge-based, since the coarse model is a physical approximation to the fine model, and the iteratively updated coarse model attempts to align itself with the fine model. The optimization process mimics a learning process derived from intuition.

4.2.5.2 Wedge Cutting Problem Solved Using Implicit Space Mapping

We use the following setup for the implicit space mapping approach. The generic surrogate model is given by $\bar{s}(x,p) = \bar{s}(x,H) = c_i(x,H)$, where $c_i(x,H) = Hx$. The weighting factors in the parameter extraction process (4.4) are, as before, $w_{i.k} = 1$ for $k = i$ and $w_{i.k} = 0$ otherwise. The surrogate model can be restated as:

$$s^{(i)}(x) = c_i(x,H^{(i)}) = H^{(i)}x \tag{4.8}$$



$x^{(0)} = 20 \Rightarrow f(x^{(0)}) = \mathbf{125}$
$\Rightarrow q^{(0)} = 5$
$\Rightarrow s^{(0)}(x) = c(x+5)$

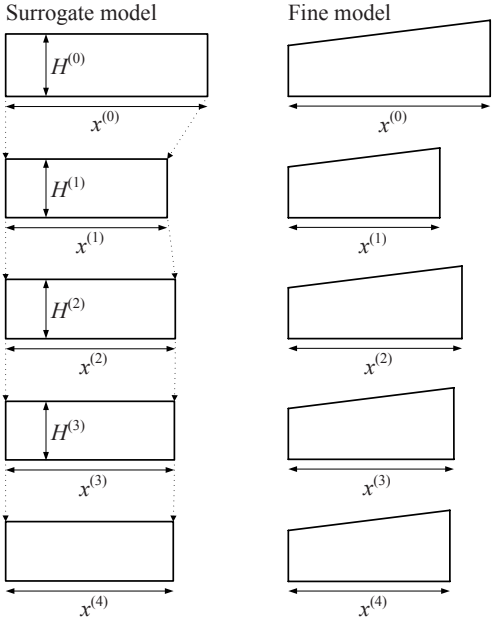
$x^{(1)} = 15 \Rightarrow f(x^{(1)}) = \mathbf{89.06}$
$\Rightarrow q^{(1)} = 2.81$
$\Rightarrow s^{(1)}(x) = c(x+2.81)$

$x^{(2)} = 17.19 \Rightarrow f(x^{(2)}) = \mathbf{104.4}$
$\Rightarrow q^{(2)} = 3.69$
$\Rightarrow s^{(2)}(x) = c(x+3.69)$

$x^{(3)} = 16.31 \Rightarrow f(x^{(3)}) = \mathbf{98.16}$
$\Rightarrow q^{(3)} = 3.32$
$\Rightarrow s^{(3)}(x) = c(x+3.32)$

$x^{(4)} = 16.68 \Rightarrow f(x^{(4)}) = \mathbf{100.76}$

Fig. 4.2 Input space mapping solving the wedge cutting problem [11]



$x^{(0)} = 20 \Rightarrow f(x^{(0)}) = \mathbf{125}$
$\Rightarrow H^{(0)} = 6.25$
$\Rightarrow s^{(0)}(x) = 6.25 \cdot x$

$x^{(1)} = 16 \Rightarrow f(x^{(1)}) = \mathbf{96.0}$
$\Rightarrow H^{(1)} = 6.00$
$\Rightarrow s^{(1)}(x) = 6.00 \cdot x$

$x^{(2)} = 16.67 \Rightarrow f(x^{(2)}) = \mathbf{100.69}$
$\Rightarrow H^{(2)} = 6.04$
$\Rightarrow s^{(2)}(x) = 6.04 \cdot x$

$x^{(3)} = 16.55 \Rightarrow f(x^{(3)}) = \mathbf{99.88}$
$\Rightarrow H^{(3)} = 6.03$
$\Rightarrow s^{(3)}(x) = 6.03 \cdot x$

$x^{(4)} = 16.57 \Rightarrow f(x^{(4)}) = \mathbf{100.02}$

Fig. 4.3 Implicit space mapping solving the wedge cutting problem

where

$$H^{(i)} = \arg\min_{H} ||f(x^{(i)}) - c_i(x^{(i)}, H)|| = \arg\min_{H} ||f(x^{(i)}) - Hx^{(i)}|| \tag{4.9}$$

In this simple case, (4.9) has an analytical solution $H^{(i)} = f(x^{(i)})/x^{(i)}$. Figure 4.3 shows the first four iterations of the SM algorithm solving the wedge cutting problem.

This indirect, implicit space mapping approach is as simple as input space mapping and also converges to an acceptable result (from an engineering point of view) in few iterations. Since our target is area, the H and the x in the Hx of the rectangle are of equal significance as design parameters in the coarse model. The physical approximation remains valid and this optimization process also mimics a learning process derived from intuition. In effect, we are recalibrating the coarse model against measurements of the fine model after each change to the fine model.

4.3 Space Mapping Efficiency

Here, we present two examples to illustrate the efficacy of space mapping technology as applied to microwave design. Specifically, we address filter-like structures operating in the frequency domain. Such problems have a long history of interest to the microwave community. Classically, filters are designed through optimally obtained transfer functions that need to be realized by practical or manufacturable circuit components and structures, whether by lumped elements, waveguide structures, integrated circuits, etc.

The field of filter design is too vast to review in detail here. For our purposes, we assume that useful structures and topologies have already been selected by experts, both the high-fidelity and lower-fidelity models. Typically the high-fidelity models are simulated by full wave electromagnetic simulators that are CPU intensive. By this we mean that a single function evaluation of the high-fidelity model can take minutes, hours, days, or much longer to evaluate. On the other hand, empirical models might be connected together though circuit theory that permits many function evaluations (of the low-fidelity model) in just seconds. Alternatively, the high-fidelity simulator may feature lower resolution simulations that are both sufficient for a coarse model and affordable in CPU time.

4.3.1 Example 1: Microstrip Bandpass Filter

Consider a second-order capacitively-coupled dual-behavior resonator (CCDBR) microstrip filter [35] shown in Fig. 4.4. The filter consists of conductor (metal) (micro)strips put on a dielectric substrate. Properties of the filter, here, the relative amount of the input signal passed to the filter's output expressed using so-called transmission coefficients, depend on the filter geometry. For this example, the design parameters are the lengths of the microstrips as well as the spacing between the lines so that the design variable vector is $x = [L_1\ L_2\ L_3\ S]^T$. The fine model is simulated by the commercial electromagnetic simulator FEKO [23]. The response

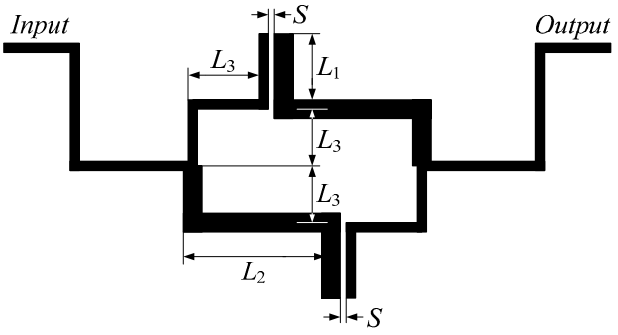


Fig. 4.4 Second-order CCDBR filter: geometry [35]

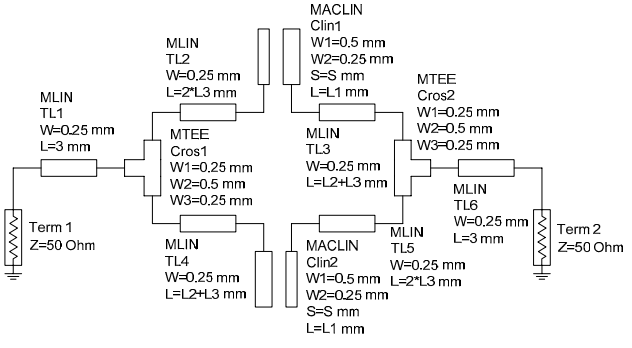


Fig. 4.5 Second-order CCDBR filter: coarse model (Agilent ADS)

$f(x)$ of the fine model is the modulus of the so-called transmission coefficient, $|S_{21}|$, evaluated at 41 frequency points equally spaced over the frequency band 2 GHz to 6 GHz. Evaluation time on a Pentium D 3.4 GHz processor is about 14 minutes.

Empirical modeling is central to engineering theory and practice. Electrical and electronics engineers, in particular, have a strong tradition of developing libraries of fast models already validated by electromagnetic analysis or physical experiment. Thus, users of commercial circuit solvers such as Agilent ADS [1] enjoy a rich and vast library of empirical elements that they can call upon in their quest to formulate suitable coarse models.

Here, a suitable coarse model, Fig. 4.5, is a circuit equivalent of the structure in Fig. 4.4, consisting of circuit-theory-based models of microstrips. The coarse model is implemented in Agilent ADS. Evaluation of the coarse model takes a few milliseconds. It should be noted that both fine and coarse models describe basically the same physical phenomena.

The design specifications are $|S_{21}| \geq -3$ dB for 3.8 GHz $\leq \omega \leq$ 4.2 GHz, and $|S_{21}| \leq -20$ dB for 2.0 GHz $\leq \omega \leq$ 3.2 GHz and 4.8 GHz $\leq \omega \leq$ 6.0 GHz. Thus, the merit function H in this case is a minimax function defined as

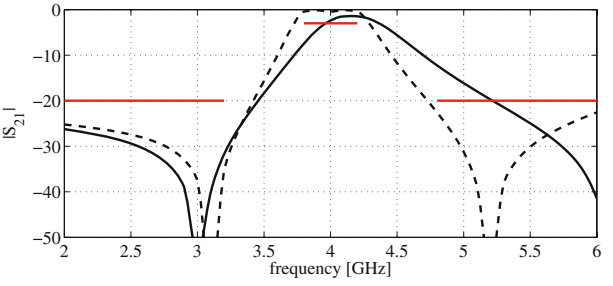


Fig. 4.6 Second-order CCDBR filter: fine model (solid line) and coarse model response (dashed line) at the starting point $x^{(0)}$. Design specifications are marked using horizontal lines

$H = \max\{\max\{3.8\,\text{GHz} \le \omega \le 4.2\,\text{GHz} : -3\,\text{dB} - |S_{21}(\omega)|\}, \max\{2.0\,\text{GHz} \le \omega \le 3.2\,\text{GHz} : |S_{21}(\omega)| + 20\,\text{dB}\}, \max\{4.8\,\text{GHz} \le \omega \le 6.0\,\text{GHz} : |S_{21}(\omega)| + 20\,\text{dB}\}\}$. Note that H is defined as the maximum of the difference between the model response and the specification value over the frequency range of interest, i.e., it describes the maximum violation of the design specifications. In particular, the positive value of H indicates that the specifications are violated at least for one frequency sample. The negative value of H indicates, on the other hand, that the response of the model satisfies the specifications. A graphical interpretation (cf. Fig. 6-9) is that a positive value of H corresponds to the response plot crossing the specifications lines, whereas a negative value of H corresponds to the response plot fitting between the specification lines.

For this problem we used input SM and output SM with the surrogate model defined as $\bar{s}(x,p) = \bar{s}(x,q,d) = c(x+q)+d$. Parameter extraction uses only the current iteration point so that the surrogate model has the form of $s^{(i)}(x) = c(x+q^{(i)})+d^{(i)}$ with $q^{(i)} = \arg\min\{q : ||f(x^{(i)}) - c(x^{(i)}+q)||\}$. Vector $d^{(i)}$ is calculated as $d^{(i)} = f(x^{(i)}) - c(x+q^{(i)})$ after vector $q^{(i)}$ is already known [29]. Due to this, the parameter extraction process is simplified: since vector d contains m components and usually m is much larger than the number of design variables n (for the CCDBR filter we have $m = 41$ and $n = 4$), having both $q^{(i)}$ and $d^{(i)}$ extracted in a nonlinear minimization process (4.4) would be much more complicated and time consuming than just extracting $q^{(i)}$. Also, having $d^{(i)}$ calculated as above allows us to satisfy the zero-order consistency condition between the fine model and the surrogate [3].

The starting point for SM optimization is the optimal solution of the coarse model, $x^{(0)} = [2.415 \;\; 6.093 \;\; 1.167 \;\; 0.082]^T$ mm. Figure 4.6 shows the responses of the fine and coarse models at $x^{(0)}$. The fine model specification error (i.e., minimax objective function value) at initial design is +7.8 dB. Figure 4.7 shows the fine model response and the response of the surrogate model $c(x+q^{(0)})$, i.e., the coarse model with the input SM vector q realizing $\arg\min q : ||f(x^{(0)}) - c(x^{(0)}+q)||$ (the response of the surrogate model $s^{(0)}(x^{(0)})$ is identical to $f(x^{(0)})$). Note that the match between the input SM surrogate and the fine model is very good. What is

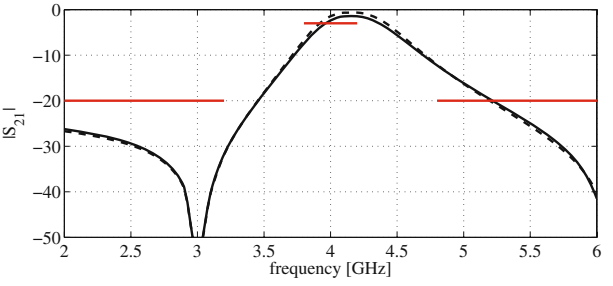


Fig. 4.7 Second-order CCDBR filter: fine model response (solid line) and the response of the surrogate model $c(x+q^{(0)})$ (dashed line) at the starting point $x^{(0)}$

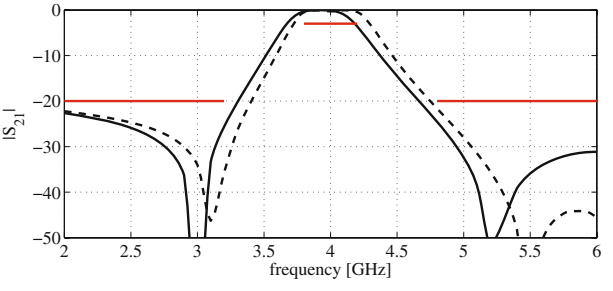


Fig. 4.8 Second-order CCDBR filter: fine model response (solid line) and the response of the surrogate model $c(x+q^{(0)})$ (dashed line) at $x^{(1)}$, the optimum of the surrogate model $s^{(0)}$

even more important, this good match is maintained away from $x^{(0)}$, as illustrated in Fig. 4.8, which shows the fine model response and the response of the surrogate model $c(x+q^{(0)})$ at $x^{(1)}$, the optimum of the surrogate model $s^{(0)}$. This means that the space mapping surrogate not only exhibits good approximation capability but also has excellent generalization (prediction) capability. It should be emphasized that the space mapping surrogate is established using fine model data at just a single point (design). This is only possible because the coarse model encodes substantial knowledge about the physical phenomena described by the fine model.

SM optimization is accomplished after five iterations. Fig. 4.9 shows the fine model response at the final solution, $x^{(5)} = [3.344\ \ 4.820\ \ 1.092\ \ 0.052]^T$ mm; the corresponding minimax objective function value is -1.4 dB. Table 4.1 compares the computational efficiency of the SM algorithm and direct optimization using Matlab's *fminimax* routine [37]. SM optimization is about 16 times faster than direct optimization. Note that the SM optimization time is slightly larger than the total fine model evaluation time ($6 \cdot 14$ min = 84 min), which is because of some overhead related to multiple evaluations of the surrogate model (cf. (4.2) and (4.4)).

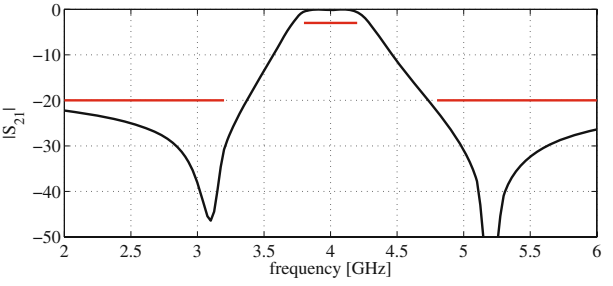


Fig. 4.9 Second-order CCDBR filter: optimized fine model response. Design specifications are marked using horizontal lines

Table 4.1 Second-order CCDBR filter: SM optimization versus direct optimization

Optimization Procedure	Final Minimax Objective Function Value	Number of Fine Model Evaluations	Total Optimization Time
Direct Optimization	−1.3 dB	106	24 h 45 min
SM Optimization	−1.4 dB	6	1 h 31 min

4.3.2 Example 2: Ring Antenna [51]

As a second example [51], consider the stacked probe-fed printed annular ring antenna shown in Fig. 4.10 [26]. The antenna is printed on a printed circuit board (PCB) with electrical permittivity $\varepsilon_{r1} = 2.2$, and height $d_1 = 6.096$ mm for the lower substrate, and $\varepsilon_{r2} = 1.07$, $d_2 = 8.0$ mm for the upper substrate. The radius of the feed pin is $r_0 = 0.325$ mm. The design parameters are the outer and inner radius of each ring and the feed position, namely $x = [a_1 \;\; a_2 \;\; b_1 \;\; b_2 \;\; \rho_1]^T$.

The fine model is evaluated using the electromagnetic simulator FEKO [23]. Response $f(x)$ of the fine model is the modulus of the transmission coefficient, $|S_{11}|$, evaluated at 9 frequency points equally spaced over the frequency band 1.75 GHz to 2.15 GHz. The coarse model is also simulated in FEKO. The difference between the fine and coarse model is in the simulation mesh density. The number of triangular meshes for the fine model is 2661, whereas the coarse model has only 83 meshes. The simulation time for the fine and coarse model is 1 hour 18 minutes and 8.7 seconds, respectively. The design specification is $|S_{11}| \leq -10$ dB for 1.75 GHz $\leq \omega \leq 2.15$ GHz.

This problem has been solved using implicit and output SM. The relative permittivities of the two layers, ε_{r1} and ε_{r2}, are used as preassigned parameters (cf. Section 4.2.2). The generic surrogate model takes the form of $\bar{s}(x,p) = \bar{s}(x,[\varepsilon_{r1} \;\; \varepsilon_{r2}]^T,d) = c(x,[\varepsilon_{r1},\varepsilon_{r2}]^T) + d$. As in the previous example, parameter extraction uses only the current iteration point so that the surrogate model has the form of $s^{(i)}(x) = c(x,[\varepsilon_{r1}^{(i)} \;\; \varepsilon_{r2}^{(i)}]^T) + d^{(i)}$ with $[\varepsilon_{r1}^{(i)} \;\; \varepsilon_{r2}^{(i)}]^T = \arg\min\{[\varepsilon_{r1} \;\; \varepsilon_{r2}]^T : ||f(x^{(i)}) - c(x^{(i)},$

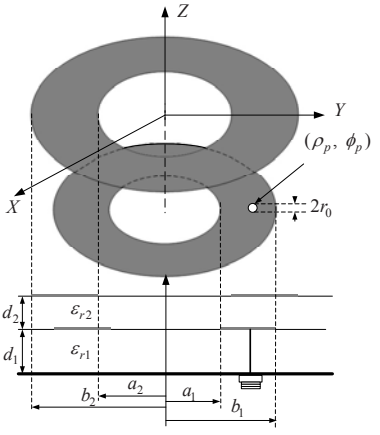


Fig. 4.10 Geometry of a stacked probe-fed printed double annular ring antenna [26]

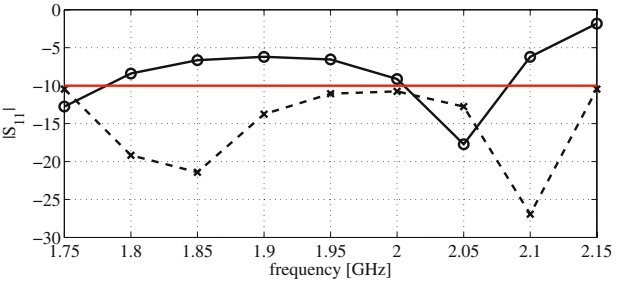


Fig. 4.11 Double annular ring antenna: fine model (solid line) and coarse model response (dashed line) at the starting point $x^{(0)}$. Design specifications are marked using a horizontal line

$[\varepsilon_{r1} \ \ \varepsilon_{r2}]^T)||\}$. Vector $d^{(i)}$ is calculated as $d^{(i)} = f(x^{(i)}) - c(x^{(i)}, [\varepsilon_{r1}^{(i)} \ \ \varepsilon_{r2}^{(i)}]^T)$ after vector $[\varepsilon_{r1}^{(i)} \ \ \varepsilon_{r2}^{(i)}]^T$ is known.

The starting point for SM optimization is the optimal solution of the coarse model, $x^{((0)} = [9.228 \ \ 8.722 \ \ 30.723 \ \ 34.127 \ \ 18.211]^T$ mm. Figure 4.11 shows the responses of the fine and coarse models at $x^{(0)}$. The fine model minimax objective function value at the initial design is +8.2 dB.

SM optimization is accomplished after three iterations (four fine model evaluations). Fig. 4.12 shows the fine model response at the final solution, $x^{(3)} = [10.674$ $7.809 \ \ 28.462 \ \ 32.504 \ \ 19.682]^T$ mm as well as the response of the surrogate model $s^{(2)}(x^{(3)})$; the fine model minimax objective function value is -0.2 dB. Note that the fine model response in Fig. 4.12 corresponds to an almost optimum design in a minimax sense. The SM optimization time is 5 hours 58 minutes and is larger than the total fine model evaluation time ($4 \cdot 78$ minutes = 5 hours 12 minutes), which is because of the overhead related to multiple evaluations of the surrogate model.

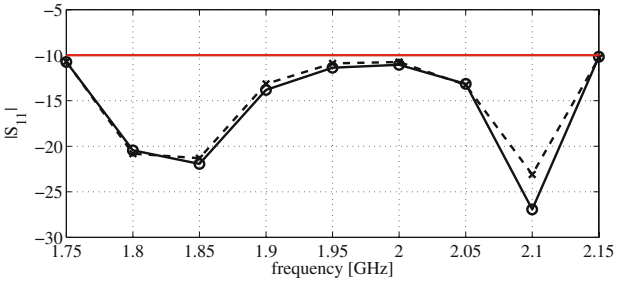


Fig. 4.12 Double annular ring antenna: fine model (solid line) and surrogate model response (dashed line) at the final design $x^{(3)}$. Design specifications are marked using a horizontal line

Direct optimization of the fine model in this example was not attempted. With a simulation time of 1 hour and 18 minutes per system analysis, direct optimization would require about a week, which is not acceptable.

4.3.3 Discussion

In this section we have studied two representative examples taken from electromagnetics-based microwave engineering design. We see that a major obstacle in executing the optimizations include the high computational cost of full wave electromagnetic simulation by commercial solvers. Space mapping effectively replaces the direct optimization of the high-fidelity model by iterative re-optimization and updating of the faster surrogate based on the problem-specific knowledge embedded in the underlying coarse model. Thus, space mapping optimization shifts the CPU burden from the slower simulator to the faster simulator.

4.4 Exploiting Extra Knowledge: Tuning Space Mapping

Tuning is ubiquitous in engineering practice. It is usually associated with the process of manipulating free or tunable parameters of a device or system after that device or system has been manufactured. The traditional purpose of permitting tunable elements is (1) to facilitate user-flexibility in achieving a desired response or behavior from a manufactured outcome during its operation, or (2) to correct inevitable postproduction manufacturing defects, small due perhaps to tolerances, or large due perhaps to faults in the manufacturing process [7, 8]. Tuning of an engineering design can be seen, in essence, as a user- or robot-directed optimization process.

The tuning space mapping approach is an iterative optimization procedure that assumes the existence of a so-called tuning model which is less accurate but computationally much cheaper than the fine model. The model incorporates relevant data from the fine model (typically fine model responses, in a manner of a device under

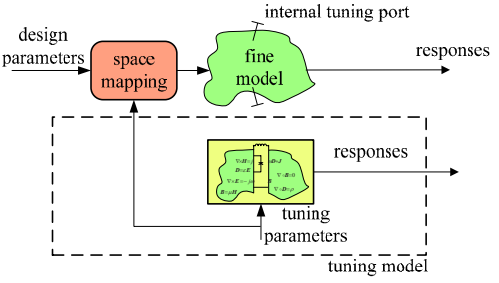


Fig. 4.13 The concept of the tuning model

test) at the current iteration point, and tuning parameters (typically implemented through circuit elements inserted into tuning ports). The tunable parameters are adjusted so that the model satisfies the original design specifications. The conceptual illustration of the tuning model is shown in Fig. 4.13. The procedure is invasive in the sense that the structure may need to be cut. The fine model simulator must allow such cuts and allow tuning elements to be inserted.

A certain relation (not necessarily analytical) between the parameters of the tuning model and the design variables is assumed, so that the new design is obtained by translating the adjusted parameters into the corresponding design variable values using this very relation.

4.4.1 Tuning Space Mapping Formulation

The TSM algorithm produces a sequence of points (design variable vectors) $x^{(i)}$, $i = 0, 1, \ldots$. The iteration of the algorithm consists of two steps: optimization of the tuning model and a calibration procedure. First, the current tuning model $t^{(i)}$ is built using fine model data at point $x^{(i)}$. In general, because the fine model has undergone a disturbance, the tuning model response may not agree with the response of the fine model at $x^{(i)}$ even if the values of the tuning parameters x_t are zero, so that these values must be adjusted to, say, $x_{t.0}^{(i)}$ in order to obtain alignment:

$$x_{t.0}^{(i)} = \arg\min_{x_t} ||f(x^{(i)}) - t^{(i)}(x_t)|| \tag{4.10}$$

In the next step, we optimize $t^{(i)}$ to have it meet the design specifications. We obtain the optimal values of the tuning parameters $x_t^{(i)}$ as follows:

$$x_{t.1}^{(i)} = \arg\min_{x_t} H(t^{(i)}(x_t)) \tag{4.11}$$

Having $x_{t.1}^{(i)}$ we perform the calibration procedure to determine changes in the design variables that yield the same change in the calibration model response as that caused

by $x_{t.1}^{(i)} - x_{t.0}^{(i)}$, where $x_{t.0}$ are initial values of the tuning parameters (normally zero). We first adjust the SM parameters $p^{(i)}$ of the calibration model c to obtain a match with the fine model response at $x^{(i)}$

$$p^{(i)} = \arg\min_{p} ||f(x^{(i)}) - c(x^{(i)}, p, x_{t.0}^{(i)})|| \tag{4.12}$$

The calibration model is then optimized with respect to the design variables in order to obtain the next iteration point $x^{(i+1)}$

$$x^{(i+1)} = \arg\min_{x} ||t^{(i)}(x_{t.1}^{(i)}) - c(x, p^{(i)}, x_{t.0}^{(i)})|| \tag{4.13}$$

Note that we use $x_{t.0}^{(i)}$ in (4.12), which corresponds to the state of the tuning model after performing the alignment procedure (4.10), and $x_{t.1}^{(i)}$ in (4.13), which corresponds to the optimized tuning model (cf. (4.11)). Thus, (4.12) and (4.13) allow us to find the change of design variable values $x^{(i+1)} - x^{(i)}$ necessary to compensate the effect of changing the tuning parameters from $x_{t.0}^{(i)}$ to $x_{t.1}^{(i)}$.

TSM exploits the standard SM optimization, classical circuit and electromagnetic (EM) theory, as well as the engineer's expertise. For example, in a physics-based simulation according to classical EM theory, design parameters such as physical length and width of a microstrip line can be mapped to a "tuning component" such as a capacitor. The calibration process then transfers the tuning parameters to physical design parameters, which can be achieved by taking advantage of classical theory and engineering experience. Still, the TSM algorithm can be seen as a specialized case of a standard SM. On the other hand, TSM allows greater flexibility in terms of the surrogate model which may, in general, involve any relation between the tuning parameters and design variables.

4.4.2 TSM Optimization of Chebyshev Bandpass Filter

Consider the box-section Chebyshev microstrip bandpass filter [34] shown in Fig. 4.14. The design parameters are $x = [L_1\ L_2\ L_3\ L_4\ L_5\ S_1\ S_2]^T$. The fine model is evaluated using the full-wave electromagnetic simulator Sonnet ***em*** [47]. The width parameters are $W = 40$ mil (1 mil = 0.001 inch) and $W_1 = 150$ mil. Substrate parameters are: relative permittivity $\varepsilon_r = 3.63$, and height $H = 20$ mil. The design specifications are $|S_{21}| \leq -20$ dB for 1.8 GHz $\leq \omega \leq$ 2.15 GHz and 2.65 GHz $\leq \omega \leq$ 3.0 GHz, and $|S_{21}| \leq -3$ dB for 2.4 GHz $\leq \omega \leq$ 2.5 GHz.

The idea of inserting tuning elements between so-called "co-calibrated" ports [40] has opened the door to successful commercial exploitation of tuning space mapping. The Sonnet ***em*** system [47] has been enhanced to permit tuning elements to be easily inserted into infinitesimal gaps between designated tuning ports. In effect, Sonnet ***em*** can represent the cut fine model by response data placed into a so-called multi-port S-parameter matrix which may be loaded into the Agilent ADS

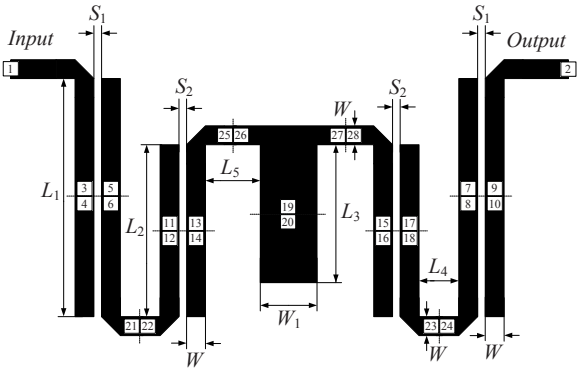


Fig. 4.14 Box-section Chebyshev bandpass filter: geometry [34], and places for inserting the tuning ports (denoted as white rectangles; the numbers correspond to the terminals of S-parameter component S28P of the tuning model shown in Fig. 4.15)

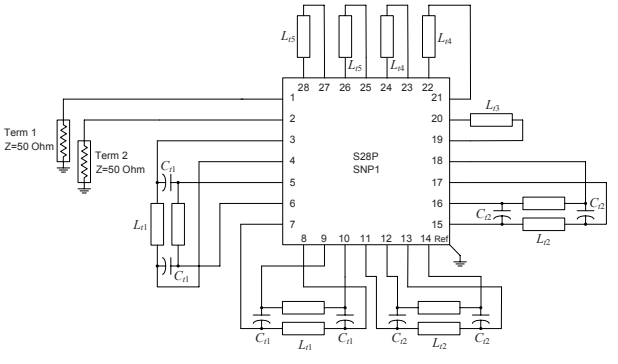


Fig. 4.15 Box-section Chebyshev bandpass filter: tuning model (Agilent ADS)

circuit simulator [1]. Tuning elements connected to the appropriate ports complete the tuning model within the ADS simulator.

Specifically in our example, the tuning model is constructed by dividing the polygons corresponding to the length parameters L_1 to L_5 in the middle and inserting the tuning ports at the new cut edges. Its S28P data file (a 28-port S-parameter matrix) is then loaded into the S-parameter component in Agilent ADS [1]. The circuit-theory coupled-line components and capacitor components are designated as tuning elements and are inserted into each pair of tuning ports (Fig. 4.15). The lengths of the imposed coupled-lines and the capacitances of the capacitors are assigned to be the tuning parameters, so that we have $x_t = [L_{t1}\ \ L_{t2}\ \ L_{t3}\ \ L_{t4}\ \ L_{t5}\ \ C_{t1}\ \ C_{t2}]^T$ (L_{tk} are in mil, C_{tk} in pF).

The calibration model is a circuit equivalent model implemented in ADS and shown in Fig. 4.16. It contains the same tuning elements as the tuning model. It basically mimics the division of the coupled-lines performed while preparing t. The

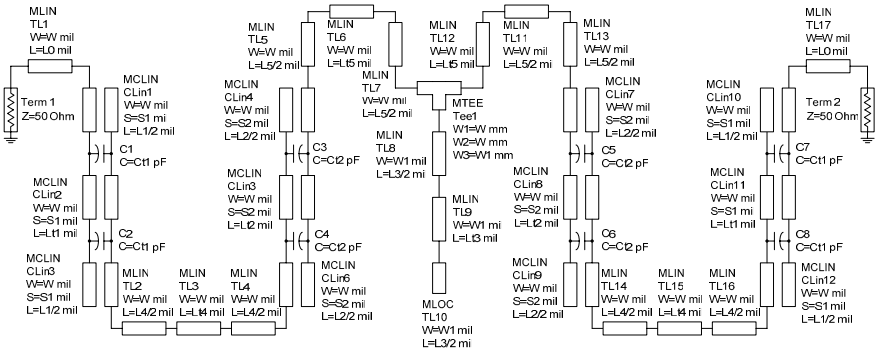


Fig. 4.16 Box-section Chebyshev bandpass filter: calibration model (Agilent ADS)

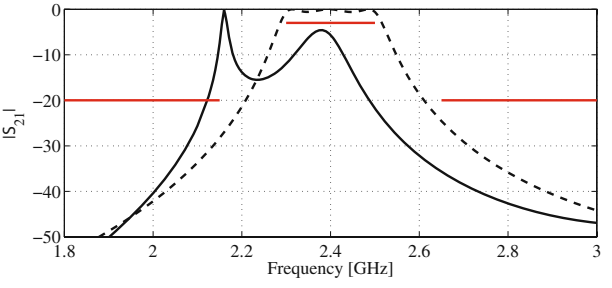


Fig. 4.17 Box-section Chebyshev bandpass filter: the coarse (dashed line) and fine (solid line) model response at the initial design. Design specifications are marked using horizontal lines

calibration model also contains six (implicit) SM parameters that will be used as parameters p in the calibration process (4.12), (4.13). These parameters are $p = [\varepsilon_{r1}\ \ \varepsilon_{r2}\ \ \varepsilon_{r3}\ \ \varepsilon_{r4}\ \ \varepsilon_{r5}\ \ H]^T$, where ε_{rk} is the dielectric constant of the microstrip line segment of length L_k (cf. Fig. 4.14), and H is the substrate height of the filter. Initial values of these parameters are $[3.63\ \ 3.63\ \ 3.63\ \ 3.63\ \ 3.63\ \ 20]^T$.

The misalignment between the fine and tuning model responses with the tuning elements set to zero is negligible so that $x_{t.0}^{(0)} = [0\ 0\ 0\ 0\ 0\ 0]^T$ was used throughout. The values of the tuning parameters at the optimal design of the tuning model are $x_{t.1}^{(0)} = [-85.2\ \ 132.5\ \ 5.24\ \ 1.13\ \ -15.24\ \ 0.169\ \ -0.290]^T$. Note that some of the parameters take negative values, which is permitted in ADS. The values of the preassigned parameters obtained in the first calibration phase (4.12) are $p^{(0)} = [3.10\ \ 6.98\ \ 4.29\ \ 7.00\ \ 6.05\ \ 17.41]^T$.

Figure 4.17 shows the coarse and fine model responses at the initial design, whereas Fig. 4.18 shows the fine model response after just one TSM iteration with $x^{(1)} = [1022\ \ 398\ \ 46\ \ 56\ \ 235\ \ 4\ \ 10]^T$ mil (the corresponding minimax objective function value is -1.8 dB).

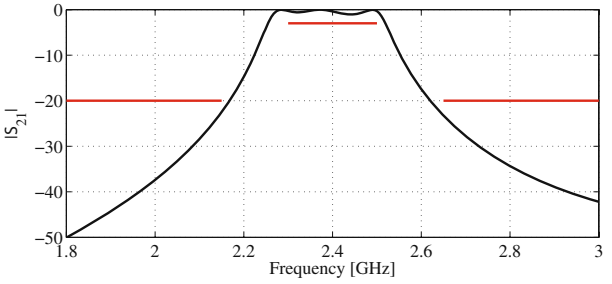


Fig. 4.18 Box-section Chebyshev bandpass filter: fine model response at the design found after one iteration of the TSM algorithm. Design specifications are marked using horizontal lines

4.4.3 Summary

We have considered simulation-based tuning within the scope of space mapping. In our TSM approach, which is significantly more knowledge-intensive than regular space mapping, we construct a tuning model directly by cutting into the fine model and connecting tuning elements to the resulting internal ports of the structure. Relevant parameters or preassigned parameters of these auxiliary elements are chosen to be tunable and are varied to match the tuning model to the fine model. This process takes little CPU effort as the tuning model is typically implemented within a circuit simulator. An updated tuning model is then available for design prediction. The prediction is fed back to fine model simulator after simple calibration. The process is repeated until the fine model response is sufficiently close to the design target.

4.5 Conclusions

Ultimately, every engineer seeks high-fidelity, but cheap, solutions to computationally expensive design problems. So much the better if, without loss of fidelity or sacrifice of optimality, the design process can be cast as a simple nonlinear optimization of black-box functions. However, to ensure low cost, a tight limit on the number of high-fidelity simulation runs to evaluate expensive objective functions and constraints is mandatory, otherwise such optimization problems can become computationally intractable. In our situation, pure classical methods of optimization are likely to perform poorly or fail since we limit the number of high-fidelity function simulations to only a handful. However, space mapping heavily exploits classical methods in the optimization of the underlying surrogates.

Input space mapping requires expert knowledge, usually deals with relatively few free optimization variables, but the parameter extraction step can be a difficult nonlinear optimization problem to solve. Expertise is helpful in implicit space mapping because of the many possibly available pre-assigned parameters. In output space mapping, engineering expertise may by somewhat less necessary, but the

process can involve a large number of variables. However, the parameter extraction step might not require coarse model re-simulation. The new tuning space mapping approach is an effective simulator-based approach but requires significantly more expertise to execute.

Essential to overall success, we believe, is a suitable combination of (1) classical optimization algorithms, (2) computational intelligence, (3) fast physics-based surrogates, and (4) the designer's engineering expertise. Our contribution to space mapping exploits these necessary ingredients.

References

1. Agilent ADS 2008: Agilent Technologies, Santa Rosa, CA, USA (2008)
2. Alexandrov, N.M., Dennis, J.E., Lewis, R.M., Torczon, V.: A trust region framework for managing the use of approximation models in optimization. Structural Optimization 15, 16–23 (1998)
3. Alexandrov, N.M., Lewis, R.M.: An overview of first-order model management for engineering optimization. Optimization Eng. 2, 413–430 (2001)
4. Amari, S., LeDrew, C., Menzel, W.: Space-mapping optimization of planar coupled-resonator microwave filters. IEEE Trans. Microwave Theory Tech. 54, 2153–2159 (2006)
5. Bakr, M.H., Bandler, J.W., Biernacki, R.M., Chen, S.H., Madsen, K.: A trust region aggressive space mapping algorithm for EM optimization. IEEE Trans. Microwave Theory Tech. 46, 2412–2425 (1998)
6. Bakr, M.H., Bandler, J.W., Ismail, M.A., Rayas-Sánchez, J.E., Zhang, Q.J.: Neural space-mapping optimization for EM-based design. IEEE Trans. Microwave Theory Tech. 48, 2307–2315 (2000)
7. Bandler, J.W., Liu, P.C., Tromp, H.: A nonlinear programming approach to optimal design centering, tolerancing and tuning. IEEE Trans. Circuits and Systems 23, 155–165 (1976)
8. Bandler, J.W., Salama, A.E.: Functional approach to microwave postproduction tuning. IEEE Trans. Microwave Theory Tech. 33, 302–310 (1985)
9. Bandler, J.W., Biernacki, R.M., Chen, S.H., Grobelny, P.A., Hemmers, R.H.: Space mapping technique for electromagnetic optimization. IEEE Trans. Microwave Theory Tech. 42, 536–544 (1994)
10. Bandler, J.W., Biernacki, R.M., Chen, S.H., Hemmers, R.H., Madsen, K.: Electromagnetic optimization exploiting aggressive space mapping. IEEE Trans. Microwave Theory Tech. 43, 2874–2882 (1995)
11. Bandler, J.W., Cheng, Q.S., Dakroury, S.A., Mohamed, A.S., Bakr, M.H., Madsen, K., Søndergaard, J.: Space mapping: the state of the art. IEEE Trans. Microwave Theory Tech. 52, 337–361 (2004)
12. Bandler, J.W., Cheng, Q.S., Nikolova, N.K., Ismail, M.A.: Implicit space mapping optimization exploiting preassigned parameters. IEEE Trans. Microwave Theory Tech. 52, 378–385 (2004)
13. Bandler, J.W., Koziel, S., Madsen, K.: Space mapping for engineering optimization. SIAG/Optimization Views-and-News Special Issue on Surrogate/Derivative-free Optimization 17, 19–26 (2006)
14. Booker, A.J., Dennis, J.E., Frank, P.D., Serafini, D.B., Torczon, V., Trosset, M.W.: A rigorous framework for optimization of expensive functions by surrogates. Structural Optimization 17, 1–13 (1999)

15. Buhmann, M.D., Ablowitz, M.J.: Radial Basis Functions: Theory and Implementations. Cambridge University Press, Cambridge (2003)
16. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust Region Methods. MPS-SIAM Series on Optimization (2000)
17. Crevecoeur, G., Dupre, L., Van de Walle, R.: Space mapping optimization of the magnetic circuit of electrical machines including local material degradation. IEEE Trans. Magn. 43, 2609–2611 (2007)
18. Dennis, J.E., Torczon, V.: Managing approximation models in optimization. In: Alexandrov, N.M., Hussaini, M.Y. (eds.) Multidisciplinary Design Optimization: State of the Art, pp. 330–347. SIAM, Philadelphia (1997)
19. Ding, X., Devabhaktuni, V.K., Chattaraj, B., Yagoub, M.C.E., Doe, M., Xu, J.J., Zhang, Q.J.: Neural network approaches to electromagnetic based modeling of passive components and their applications to high-frequency and high-speed nonlinear circuit optimization. IEEE Trans. Microwave Theory Tech. 52, 436–449 (2004)
20. Dorica, M., Giannacopoulos, D.D.: Response surface space mapping for electromagnetic optimization. IEEE Trans. Magn. 42, 1123–1126 (2006)
21. Echeverria, D., Hemker, P.W.: Space mapping and defect correction. CMAM The International Mathematical Journal Computational Methods in Applied Mathematics 5, 107–136 (2005)
22. Encica, L., Makarovic, J., Lomonova, E.A., Vandenput, A.J.A.: Space mapping optimization of a cylindrical voice coil actuator. IEEE Trans. Industry Applications 42, 1437–1444 (2006)
23. FEKO User's Manual, Suite 5.4, EM Software and Systems-S.A (Pty) Ltd., Stellenbosch, South Africa (2008)
24. Gano, S.E., Renaud, J.E., Sanders, B.: Variable fidelity optimization using a kriging based scaling function. In: Proc. 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conf., Albany, New York, USA (2004)
25. Ismail, M.A., Smith, D., Panariello, A., Wang, Y., Yu, M.: EM-based design of large-scale dielectric-resonator filters and multiplexers by space mapping. IEEE Trans. Microwave Theory Tech. 52, 386–392 (2004)
26. Kokotoff, D.M., Aberle, J.T., Waterhouse, R.B.: Rigorous analysis of probe-fed printed annular ring antennas. IEEE Trans. Antennas and Propagation 47, 384–388 (1999)
27. Koziel, S., Bandler, J.W., Mohamed, A.S., Madsen, K.: Enhanced surrogate models for statistical design exploiting space mapping technology. In: IEEE MTT-S Int. Microwave Symp. Dig., Long Beach, CA, pp. 1609–1612 (2005)
28. Koziel, S., Bandler, J.W., Madsen, K.: Space-mapping based interpolation for engineering optimization. IEEE Trans. Microwave Theory Tech. 54, 2410–2421 (2006)
29. Koziel, S., Bandler, J.W., Madsen, K.: A space mapping framework for engineering optimization: theory and implementation. IEEE Trans. Microwave Theory Tech. 54, 3721–3730 (2006)
30. Koziel, S., Bandler, J.W.: Space-mapping optimization with adaptive surrogate model. IEEE Trans. Microwave Theory Tech. 55, 541–547 (2007)
31. Koziel, S., Bandler, J.W., Madsen, K.: Quality assessment of coarse models and surrogates for space mapping optimization. Optimization Eng. 9, 375–391 (2008)
32. Koziel, S., Cheng, Q.S., Bandler, J.W.: Space mapping. IEEE Microwave Magazine 9(6), 105–122 (2008)
33. Leary, S.J., Bhaskar, A., Keane, A.J.: A constraint mapping approach to the structural optimization of an expensive model using surrogates. Optimization Eng. 2, 385–398 (2001)

34. Liao, C.K., Chi, P.L., Chang, C.Y.: Microstrip realization of generalized Chebyshev filters with box-like coupling schemes. IEEE Trans. Microwave Theory Tech. 55, 147–153 (2007)
35. Manchec, A., Quendo, C., Favennec, J.F., Rius, E., Person, C.: Synthesis of capacitive-coupled dual-behavior resonator (CCDBR) filters. IEEE Trans. Microwave Theory Tech. 54, 2346–2355 (2006)
36. Marsden, A.L., Wang, M., Dennis, J.E., Moin, P.: Optimal aeroacoustic shape design using the surrogate management framework. Optimization Eng. 5, 235–262 (2004)
37. Matlab, ver. 7.14, The MathWorks, Inc., Natick, MA, USA (2008)
38. Miraftab, V., Mansour, R.R.: A robust fuzzy-logic technique for computer-aided diagnosis of microwave filters. IEEE Trans. Microwave Theory Tech. 52, 450–456 (2004)
39. Queipo, N.V., Haftka, R.T., Shyy, W., Goel, T., Vaidynathan, R., Tucker, P.K.: Surrogate-based analysis and optimization. Progress in Aerospace Sciences 41, 1–28 (2005)
40. Rautio, J.C.: RF design closure—companion modeling and tuning methods. In: IEEE MTT IMS Workshop: Microwave component design using space mapping technology, San Francisco, CA (2006)
41. Rayas-Sánchez, J.E.: EM-based optimization of microwave circuits using artificial neural networks: the state of the art. IEEE Trans. Microwave Theory Tech. 52, 420–435 (2004)
42. Rayas-Sánchez, J.E., Lara-Rojo, F., Martínez-Guerrero, E.: A linear inverse space mapping (LISM) algorithm to design linear and nonlinear RF and microwave circuits. IEEE Trans. Microwave Theory Tech. 53, 960–968 (2005)
43. Redhe, M., Nilsson, L.: Using space mapping and surrogate models to optimize vehicle crashworthiness design. In: 9th AIAA/ISSMO Multidisciplinary Analysis and Optimization Symp., Atlanta, GA, Paper AIAA-2002-5536 (2002)
44. Robinson, T.D., Eldred, M.S., Willcox, K.E., Haimes, R.: Surrogate-based optimization using multifidelity models with variable parameterization and corrected space mapping. AIAA Journal 46, 2814–2822 (2008)
45. Simpson, T.W., Maurey, T.M., Korte, J.J., Mistree, F.: Kriging models for global approximation in simulation-based multidisciplinary design optimization. AIAA Journal 39, 2233–2241 (2001)
46. Simpson, T.W., Peplinski, J., Koch, P.N., Allen, J.K.: Metamodels for computer-based engineering design: survey and recommendations. Engineering with Computers 17, 129–150 (2001)
47. Sonnet *em* Version 11.54, Sonnet Software, Inc., North Syracuse, NY, USA (2008)
48. Van Beers, W.C.M., Kleijnen, J.P.C.: Kriging interpolation in simulation: survey. In: Proc. 2004 Winter Simulation Conf., pp. 113–121 (2004)
49. Wu, K.L., Zhao, Y.J., Wang, J., Cheng, M.K.K.: An effective dynamic coarse model for optimization design of LTCC RF circuits with aggressive space mapping. IEEE Trans. Microwave Theory Tech. 52, 393–402 (2004)
50. Zhang, L., Xu, J., Yagoub, M.C.E., Ding, R., Zhang, Q.J.: Efficient analytical formulation and sensitivity analysis of neuro-space mapping for nonlinear microwave device modeling. IEEE Trans. Microwave Theory Tech. 53, 2752–2767 (2005)
51. Zhu, J., Bandler, J.W., Nikolova, N.K., Koziel, S.: Antenna optimization through space mapping. IEEE Trans. Antennas and Propagation 55, 651–658 (2007)

Chapter 5
Reducing Function Evaluations Using Adaptively Controlled Differential Evolution with Rough Approximation Model

Tetsuyuki Takahama and Setsuko Sakai

Abstract. In this chapter, a rough approximation model, which is an approximation model with low accuracy and without learning process, is presented in order to reduce the number of function evaluations effectively. Although the approximation errors between the true function values and the approximation values are not small, the rough model can estimate the order relation of solutions with fair accuracy. By utilizing this nature of the rough model, we have proposed estimated comparison method, in which function evaluations are omitted when the order relation of solutions can be judged by approximation values. In the method, a parameter for error margin is introduced to avoid incorrect judgment. Also, a parameter for utilizing congestion of solutions is introduced to avoid omitting promising solutions. In order to improve the stability and efficiency of the method, we propose adaptive control of the margin parameter and the congestion parameter according to the success rate of the judgment. The advantage of these improvements is shown by comparing the results obtained by Differential Evolution (DE), DE with the estimated comparison method, adaptively controlled DE with the estimated comparison method and particle swarm optimization in various types of benchmark functions.

5.1 Introduction

Evolutionary computation has been successfully applied to various fields of science and engineering. Evolutionary algorithms (EAs) have been proved to be powerful function optimization algorithms. However, EAs need a large number of function

Tetsuyuki Takahama
Hiroshima City University, 3-4-1 Ozukahigashi, Asaminami-ku, Hiroshima, Japan
e-mail: takahama@info.hiroshima-cu.ac.jp

Setsuko Sakai
Hiroshima Shudo University, 1-1-1 Ozukahigashi, Asaminami-ku, Hiroshima, Japan
e-mail: setuko@shudo-u.ac.jp

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 111–129.
springerlink.com

evaluations before a well acceptable solution can be found. Recently, the size of optimization problems tends to become large, and the cost of function evaluation becomes high. It is necessary to develop more efficient optimization algorithms in order to reduce the number of function evaluations.

An effective method for reducing function evaluations is to build an approximation model for the objective function and to solve the optimization problems using the approximation values [6]. However, building the high-quality approximation model is very difficult and time-consuming. Also, a proper approximation model depends on the problems to be optimized. It is difficult to design a general-purpose approximation model with high accuracy.

In order to solve these difficulties, we have proposed *estimated comparison method* [17, 20]. In the method, an approximation model with low accuracy and without learning process, or a rough approximation model, is utilized to reduce the number of function evaluations effectively. The approximation errors between the true function values and the approximation values estimated by the rough approximation model will not be small. However, the rough model can estimate whether the function value of a solution is smaller than that of the other solution or not with fair accuracy, and can be used to compare solutions. Thus, the estimated comparison, which compares solutions using the rough approximation values, can be defined.

In the estimated comparison, the approximation values are compared first. When a value is judged to be worse enough than the other value, the estimated comparison returns the estimated result without evaluating the objective function. When it is difficult to judge the result of comparison from the approximation values, true values are obtained by evaluating the objective function and the estimated comparison returns the true result based on the true values. By using the estimated comparison, the evaluation of the objective function is sometimes omitted and the number of function evaluations can be reduced.

Two parameters, an error margin parameter and a congestion parameter, are introduced in the estimated comparison. The *error margin parameter* is used to allow approximation error and avoid incorrect judgment. If the error margin is too large, cautious judgment is made and the objective function is often evaluated to obtain true values. As the result, the efficiency of optimization could not be improved much. If the margin parameter is too small, the evaluation of the objective function is often omitted, but the judgment would often be incorrect. As the result, the optimization process might be led to a wrong direction. In this study, we propose to control the margin parameter adaptively based on success rate of the comparison. The *congestion parameter* is used not to block the search for new direction. If the congestion of a solution is low, the solution exists in new or not-yet-visited area and it is difficult to estimate the true function value of the solution. It is important to evaluate the function and confirm whether the solution is good or not. If the congestion parameter is too large, the solution in new area will always be evaluated. As the result, the efficiency of optimization could not be improved much. If the congestion parameter is too small, the search for new area would often be blocked. As the result, the speed of optimization process might be slow down. In this study, we propose to control the congestion parameter adaptively based on the success rate.

In this chapter, *potential model* is used as a rough approximation model. The potential model can estimate a function value of a point based on some other points without learning process and can be used as a general-purpose rough approximation model. Differential Evolution (DE) [2, 13, 14, 15, 19] is used as an optimization algorithm and the estimated comparison is introduced in the survivor selection phase of DE. The advantage of these improvements is shown by comparing the results obtained by DE, DE with the estimated comparison method, adaptively controlled DE with the estimated comparison method and particle swarm optimization in various types of benchmark functions.

The rest of this chapter is organized as follows: Section 2 describes evolutionary algorithms using approximation models briefly. Section 3 describes the potential model as a rough approximation model. The estimated comparison is defined using the potential model and the adaptive control of parameters is proposed. The adaptive DE with the estimated comparison method is proposed in Section 4. Section 5 presents experimental results on various benchmark problems. Section 6 describes a comparative study between the adaptive method and particle swarm optimization. Finally, Section 7 concludes with a brief summary of this chapter and a few remarks.

5.2 Optimization and Approximation Models

5.2.1 Optimization Problems

In this study, the following optimization problem (P) with upper bound constraints and lower bound constraints will be discussed.

$$\begin{aligned} \text{(P)} \quad & \text{minimize } f(\boldsymbol{x}) \\ & \text{subject to } l_i \leq x_i \leq u_i,\ i = 1,\ldots,n, \end{aligned} \tag{5.1}$$

where $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ is an n dimensional vector, $f(\boldsymbol{x})$ is an objective function. Values u_i and l_i are the upper bound and the lower bound of x_i, respectively. Also, let the search space in which every point satisfies the upper and lower bound constraints be denoted by $\mathcal{S}$.

The objective function $f(\boldsymbol{x})$ will be approximated using a rough approximation model.

5.2.2 Evolutionary Algorithms Using Approximation Models

In this section, evolutionary algorithms using approximation models are briefly reviewed.

Various approximation models are utilized to approximate the objective function. For example, quadratic model is used as a simple case of polynomial models [4]. Kriging models [4, 12] approximate the function by a global model and a localized deviation. Also, multi-layered neural networks in neural network models [11] and

Radial Basis Function (RBF) network models [5, 7, 8, 9] are often used. In most approximation models, model parameters are learned by least square method, gradient method, maximum likelihood method and so on. In general, learning model parameters is time-consuming process, especially in order to obtain a model with higher accuracy or a model of a large function such as a function with large dimensions.

Evolutionary algorithms with approximation models can be classified into some types:

- All individuals have only approximation values. Very high quality approximation model is built and the objective function is optimized using approximation values only. It is possible to reduce function evaluations greatly. However, these methods can be applied to well-informed objective function and cannot be applied to general problems.
- Some individuals have approximation values and others have true values. Methods in this type are called evolution control approaches and can be classified into individual-based and generation-based control [6]. The individual-based control means that good individuals (or randomly selected individuals) use true values and others use approximation values in each generation [7, 8]. The generation-based control means that all individuals use true values once in a fixed number of generations and use approximation values in other generations [8, 9]. In the approaches, the approximation model should be accurate because approximation values are compared with true values. Also, it is known that approximation models with high accuracy sometimes generate a false optimum or hide a true optimum. Individuals may converge into a false optimum while they are optimized using the approximation models in some generations. Thus, these approaches are much affected by the quality of approximation models. It is difficult to utilize rough approximation models.
- All individuals have true values. Some methods in this type are called surrogate approaches. In the surrogate approaches, an estimated optimum is searched using an approximation model that is usually a local model. The estimated optimum is evaluated to obtain the true value and also to improve the approximation model [1, 5, 10]. If the true value is good, the value is included as an individual. In the approaches, rough approximation models might be used because approximation values are compared with other approximation values. These approaches are less affected by the approximation model than the evolution control approaches. However, they have the process of optimization using the approximation model only. If the process is repeated many times, they are much affected by the quality of approximation models.

5.2.3 Estimated Comparison Method

The estimated comparison method is classified into the category where all individuals have true values. However, the method is different from the surrogate approaches. It uses a global approximation model of current individuals using a rough approximation model. It does not search for an estimated optimum, but it judges

whether a new individual is worth evaluating its true value or not. Also, it can specify the error margin parameter for allowing approximation error and the congestion parameter for accepting the promising solutions in a new direction when the comparison is carried out. Thus, it is not affected by the approximation model much.

The reduction of function evaluations by the estimated comparison method is not larger than that by other optimization methods using approximation models with high accuracy. However, the estimated comparison method does not need the learning process of the approximation model which is often time-consuming and needs much effort to tune the learning parameters. The estimated comparison method is fast and easy-to-use approach and can be applied to wide range of problems including from low or medium computation cost to high computation cost problems. It is thought that the estimated comparison method is a more general-purpose method than other methods with high-quality approximation models.

5.3 Rough Approximation Model

In this study, potential model is used as a rough approximation model.

5.3.1 Potential Model

Potential energy is stored energy that depends on the relative position of various parts of a system. The gravity potential energy is an example of potential energy. If there is an object of which mass is m, there exists gravity potential energy E_g around the object. If there is another object of which mass is m' at a distance r from the object, there exists the attractive force F_g between two objects.

$$E_g = -G\frac{m}{r}, \quad F_g = G\frac{mm'}{r^2} \tag{5.2}$$

where G is a proportional constant or the gravitational constant.

In this study, it is supposed that when a solution $\boldsymbol{x}$ exists, there are potential for objective U_o and potential for congestion U_c at a distance r from the solution as follows:

$$U_o = \frac{f(\boldsymbol{x})}{r^p} \tag{5.3}$$

$$U_c = \frac{1}{r^p} \tag{5.4}$$

where p is a positive number and usually 1 or 2. The proportional constant is 1 for simplicity.

When a set of solutions $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$ are given and the objective values $f(\boldsymbol{x}_i)$, $i = 1, 2, \cdots, N$ are known, two potential functions at a point $\boldsymbol{y}$ can be defined as follows:

$$U_o(\boldsymbol{y}) = \sum_i \frac{f(\boldsymbol{x}_i)}{d(\boldsymbol{x}_i,\boldsymbol{y})^p} \tag{5.5}$$

$$U_c(\boldsymbol{y}) = \sum_i \frac{1}{d(\boldsymbol{x}_i,\boldsymbol{y})^p} \tag{5.6}$$

where $d(\boldsymbol{x},\boldsymbol{y})$ is a distance between points $\boldsymbol{x}$ and $\boldsymbol{y}$.

It is obvious that U_o shows a measure of the function value at $\boldsymbol{y}$ and U_c shows the congestion of the point $\boldsymbol{y}$. If U_o is large, the function value tends to be large. If U_c is large, there are many points near the point.

The approximation value $\hat{f}(\boldsymbol{y})$ at the point $\boldsymbol{y}$ can be defined as follows:

$$\hat{f}(\boldsymbol{y}) = U_o(\boldsymbol{y})/U_c(\boldsymbol{y}) \tag{5.7}$$

For example, if $\boldsymbol{y}$ is same as $\boldsymbol{x}_i$, $\hat{f}(\boldsymbol{y})$ becomes $f(\boldsymbol{x}_i)$.

5.3.2 Estimated Comparison

When the true function values of all points in $X = \{\boldsymbol{x}_i | i = 1,2,\cdots,N\}$ are known and a new child point $\boldsymbol{x}_i'$ is generated from a parent point $\boldsymbol{x}_i$, the approximation values at points $\boldsymbol{x}_i'$ and $\boldsymbol{x}_i$ are given as follows:

$$U_o(\boldsymbol{x}_i^{(\prime)}) = \sum_{j\neq i} \frac{f(\boldsymbol{x}_j)}{d(\boldsymbol{x}_j,\boldsymbol{x}_i^{(\prime)})^p} \tag{5.8}$$

$$U_c(\boldsymbol{x}_i^{(\prime)}) = \sum_{j\neq i} \frac{1}{d(\boldsymbol{x}_j,\boldsymbol{x}_i^{(\prime)})^p} \tag{5.9}$$

$$\hat{f}(\boldsymbol{x}_i^{(\prime)}) = U_o(\boldsymbol{x}_i^{(\prime)})/U_c(\boldsymbol{x}_i^{(\prime)}) \tag{5.10}$$

It should be noted that the parent point $\boldsymbol{x}_j (j = i)$ is omitted in the right side of U_o and U_c. If the parent point is not omitted, the approximation value of the parent point becomes the true value. As the result, the difference between the precision of approximation at the parent point and that at the child point becomes large, and it is difficult to compare the approximation values.

The estimated comparison judges whether the child point is better than the parent point. In the comparison, a reference value z for indicating accuracy level of the approximation model, the error margin parameter $\delta(\delta \geq 0)$ and the congestion parameter $\lambda(0 \leq \lambda \leq 1)$ are introduced. The estimated comparison can be defined as follows:

```
better(x_i', x_i, z) {
  if(U_c(x_i') ≤ λU_c(x_i) || f̂(x_i') ≤ f̂(x_i) + δz) {
     Evaluate x_i';
     if(f(x_i') < f(x_i)) return yes;
  }
  return no;
}
```

When λ is 0, the congestion is not considered because U_c is positive. Otherwise, if the congestion of child point is much less than that of parent point, the child point is evaluated. The recommended value of λ is $[\frac{1}{5}, \frac{1}{2}]$.

When δ is 0, the estimated comparison can reject many children and omit a large number of function evaluations. However, the possibility of rejecting good child becomes high and a true optimum sometimes might be skipped. When δ is large, the possibility of rejecting good child becomes low. However, the estimated comparison can reject fewer children and omit a small number of function evaluations. Thus, δ should have a proper value.

In this study, the reference value z is given by the standard deviation of approximation values, σ:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{f}(\boldsymbol{x}_i) - \bar{f}\right)^2} \tag{5.11}$$

$$\bar{f} = \frac{1}{N}\sum_{i=1}^{N}\hat{f}(\boldsymbol{x}_i) \tag{5.12}$$

In this case, the recommended value of the margin parameter δ is $[0.05, 0.5]$.

5.3.3 Adaptive Control

In this study, a simple adaptive control schema of δ and λ, which is derived from Rechenberg's 1/5 success rule, is adopted as shown in Table 5.1. The accuracy of the approximation model is classified into 5 categories, Very Bad, Bad, Normal, Good, and Very Good, according to the success rate of the comparison between the child and the parent. The success rate is given by the ratio between the number of successful evaluations, or the number of cases when the child is better than the parent and the parent is replaced by the child, and the total number of evaluations in one generation.

If the success rate is large, the accuracy of the approximation model is high and the parameters δ and λ can be decreased. Thus, when the success rate is Good, δ and λ are decreased by a factor of 0.9. When the rate is Very Good, δ and λ have the

Table 5.1 Rules of adaptive control

Category	Success Rate	$\delta(t+1)$	$\lambda(t+1)$
Very Bad	$[0, 0.1]$	$\delta_{\max}$	$\lambda_{\max}$
Bad	$(0.1, 0.2]$	$1.5\delta(t)$	$1.5\lambda(t)$
Normal	$(0.2, 0.3]$	$\delta(t)$	$\lambda(t)$
Good	$(0.3, 0.4]$	$0.9\delta(t)$	$0.9\lambda(t)$
Very Good	$(0.4, \infty)$	$\delta_{\min}$	$\lambda_{\min}$

where $\delta(t) \in [\delta_{\min}, \delta_{\max}] = [10^{-5}, 0.2]$,
$\lambda(t) \in [\lambda_{\min}, \lambda_{\max}] = [10^{-3}, 0.4]$.

minimum value δ_{min} and λ_{min}, respectively. If the success rate is small, the accuracy is low and the margin parameter and the congestion parameter should be increased. Thus, when the rate is Bad, δ and λ are increased by a factor of 1.5. When the rate is Very Bad, δ and λ have the maximum value δ_{max} and λ_{max}, respectively.

5.4 Differential Evolution with the Estimated Comparison Method

In this section, Differential Evolution and DE with the estimated comparison method are described.

5.4.1 Differential Evolution

Differential evolution is a variant of ES proposed by Storn and Price [13, 14]. DE is a stochastic direct search method using population or multiple search points. DE has been successfully applied to the optimization problems including non-linear, non-differentiable, non-convex and multi-modal functions. It has been shown that DE is fast and robust to these functions [2].

There are some variants of DE that have been proposed, such as DE/best/1/bin and DE/rand/1/exp. The variants are classified using the notation DE/*base*/*num*/*cross*. "*base*" indicates the method of selecting a parent that will form the base vector. For example, DE/rand/*num*/*cross* selects the parent for the base vector at random from the population. DE/best/*num*/*cross* selects the best individual in the population. "*num*" indicates the number of difference vectors used to perturb the base vector. "*cross*" indicates the crossover mechanism used to create a child. For example, DE/*base*/*num*/bin shows that crossover is controlled by binomial crossover using constant crossover rate. DE/*base*/*num*/exp shows that crossover is controlled by a two-point crossover with exponentially decreasing crossover rate.

In DE, initial individuals are randomly generated within the search space and form an initial population. Each individual contains n genes as decision variables or a decision vector. At each generation or iteration, all individuals are selected as parents. Each parent is processed as follows: The mutation process begins by choosing $1 + 2$ *num* individuals from all individuals except for the parent in the processing. The first individual is a base vector. All subsequent individuals are paired to create *num* difference vectors. The difference vectors are scaled by a scaling factor F and added to the base vector. The resulting vector is then recombined with the parent. The probability of recombination at an element is controlled by the crossover rate CR. This crossover process produces a trial vector. Finally, for survivor selection, the trial vector is accepted for the next generation if the trial vector is better than the parent.

In this study, DE/rand/1/exp variant, where the number of difference vector is 1 or $num = 1$, is used.

5.4.2 Adaptive DE with the Estimated Comparison Method

The algorithm of the adaptive DE with the estimated comparison method based on DE/rand/1/exp variant, which is used in this study, is as follows:

Step 0: Initialization. Initial N individuals $\boldsymbol{x}_i$ are generated randomly in the search space $\mathcal{S}$ and form an initial population $P = \{\boldsymbol{x}_i | i = 1, 2, \cdots, N\}$. The parameters of the estimated comparison method are initialized as $\delta = \delta_{\max}$ and $\lambda = \lambda_{\max}$.

Step 1: Termination condition. If predefined condition, such that the number of generations (iterations) exceeds the maximum generation (iteration) $T_{\max}$, is satisfied, the algorithm is terminated.

Step 2: Mutation. For each individual $\boldsymbol{x}_i$, three individuals $\boldsymbol{x}_{p1}$, $\boldsymbol{x}_{p2}$ and $\boldsymbol{x}_{p3}$ are chosen from the population without overlapping $\boldsymbol{x}_i$ and each other. A new vector $\boldsymbol{x}'$ is generated by the base vector $\boldsymbol{x}_{p1}$ and the difference vector $\boldsymbol{x}_{p2} - \boldsymbol{x}_{p3}$ as follows:

$$\boldsymbol{x}' = \boldsymbol{x}_{p1} + F(\boldsymbol{x}_{p2} - \boldsymbol{x}_{p3}) \tag{5.13}$$

where F is a scaling factor.

Step 3: Crossover. The vector $\boldsymbol{x}'$ is recombined with the parent $\boldsymbol{x}_i$. A crossover point j is chosen randomly from all dimensions $[1, n]$. The element at the j-th dimension of the trial vector $\boldsymbol{x}^{\text{new}}$ is inherited from the j-th element of the vector $\boldsymbol{x}'$. The elements of subsequent dimensions are inherited from $\boldsymbol{x}'$ with exponentially decreasing probability defined by a crossover rate CR. Otherwise, the elements are inherited from the parent $\boldsymbol{x}_i$. In real processing, Step2 and Step3 are integrated as one operation.

Step 4: Survivor selection. The estimated comparison is used for comparing the trial vector and the parent. The trial vector $\boldsymbol{x}^{\text{new}}$ is accepted for the next generation if the trial vector is better than the parent $\boldsymbol{x}_i$ by using the estimated comparison.

Step 5: Adaptive Control of the parameters. The success rate is calculated and the parameters δ and λ are updated according to Table 5.1.

Step 6: Go back to Step1.

The pseudo-code of the adaptive DE/rand/1/exp with the estimated comparison method is as follows:

```
Adaptive_DE/rand/1/exp_with_estimated_comparison()
{
  P=Generate N individuals {x_i} randomly;
  Evaluate x_i,  i = 1,2,···,N;
  Initialize δ = δ_max and λ = λmax;
  for(t=1; termination condition is not satisfied; t++) {
    σ=the standard deviation of approximation values in P;
    for(i=1; i ≤ N; i++) {
      (p_1,p_2,p_3)=select randomly in [1,N]\{i}
                s.t. p_j ≠ p_k(j,k = 1,2,3, j ≠ k);
      x_i^new=x_i ∈ P;
      j=select randomly from [1,n];
```

```
      k=1;
      do {
        x_ij^new=x_{p1,j}+F(x_{p2,j}−x_{p3,j});
        j=(j+1)%n;
        k++;
      } while(k≤n && u(0,1)<CR);
// estimated comparison
      if(better(x^new, x_i, σ)) x_i=x^new;
    }
    Update δ and λ adaptively;
  }
}
```

where $u(0,1)$ is a uniform random variable generator between $[0,1]$.

In this study, current population P is used as the set of solutions which have known objective values. As the search process progresses, the area where individuals exist may become elliptical. In order to handle such a case, the normalized distance is introduced, in which the distance is normalized by the width of each dimension in the current population P.

$$d(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_j \left(\frac{x_j - y_j}{\max_{\boldsymbol{x}_i \in P} x_{ij} - \min_{\boldsymbol{x}_i \in P} x_{ij}} \right)^2} \tag{5.14}$$

5.5 Numerical Experiments

5.5.1 Test Problems

In this section, the estimated comparison method is applied to sphere function, Rosenbrock function, Rastrigin function, Ackley function and Griewank function. These functions have various surfaces such as unimodal, multimodal, smooth, bumpy, or steep surfaces. Table 5.2 shows features of the functions.

The function definitions and their search spaces, where n is the dimension of the decision vector, are as follows:

- f_1: Sphere function

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2, \ -5.12 \le x_i \le 5.12 \tag{5.15}$$

  This function is a unimodal function and has the minimum value 0 at $(0,0,\cdots,0)$.
- f_2: Generalized Rosenbrock (Star type) function

$$f(\boldsymbol{x}) = \sum_{i=2}^{n} \{100(x_1 - x_i^2)^2 + (x_i - 1)^2\}, \ -2.048 \le x_i \le 2.048$$

This function is a unimodal function with a steep surface and has the minimum value 0 at $(1,1,\cdots,1)$.

- f_3: Ill-scaled generalized Rosenbrock (Star type) function

$$f(\boldsymbol{x}) = \sum_{i=2}^{n} \{100(x_1 - (ix_i)^2)^2 + (ix_i - 1)^2\},\ -2.048/i \le x_i \le 2.048/i$$

This function is a unimodal and ill-scaled function with a steep surface and has the minimum value 0 at $(1, \frac{1}{2}, \cdots, \frac{1}{n})$.

- f_4: Generalized Rastrigin function

$$f(\boldsymbol{x}) = 10n + \sum_{i=1}^{n} \{x_i^2 - 10\cos(2\pi x_i)\},\ -5.12 \le x_i \le 5.12$$

This function is a multimodal function with a very bumpy surface and has the minimum value 0 at $(0,0,\cdots,0)$.

- f_5: Ackley function

$$f(\boldsymbol{x}) = 20 + \exp(1) - 20\exp\left(-0.2\frac{1}{n}\sum_{i=1}^{n} x_i^2\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right),$$
$$-32 \le x_i \le 32$$

This function is a multimodal function with a bumpy surface and has the minimum value 0 at $(0,0,\cdots,0)$.

- f_6: Griewank function

$$f(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \Pi_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1,\ -600 \le x_i \le 600$$

This function is a multimodal function with a less bumpy surface and has the minimum value 0 at $(0,0,\cdots,0)$.

Figure 5.1 shows the graphs of functions f_2, f_4, f_5 and f_6 in case of $n = 2$.

Table 5.2 Features of test functions

Function	modality	surface	dependency of variables	ill-scale
f_1	unimodal	smooth	—	—
f_2	unimodal	steep	strong	—
f_3	unimodal	steep	strong	strong
f_4	multimodal	bumpy (large bumps)	—	—
f_5	multimodal	bumpy	—	—
f_6	multimodal	bumpy (small bumps)	—	—

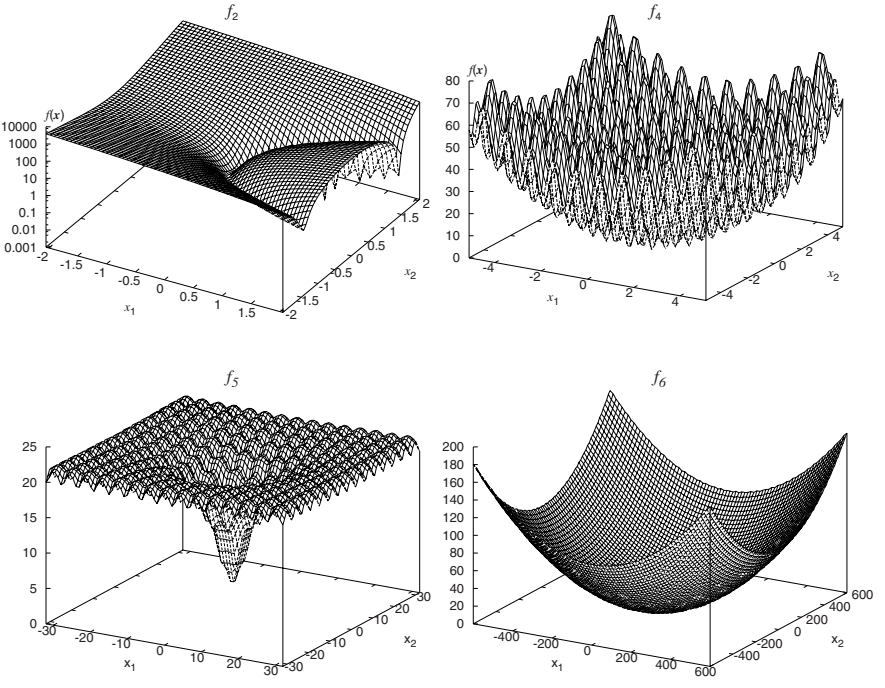


Fig. 5.1 Graphs of f_2, f_4, f_5 and f_6

5.5.2 Conditions of Experiments

All functions are optimized with a fairly large number of decision variables $n = 50$. Experimental conditions for DE, potential model and the estimated comparison are as follows: Parameters for DE are population size $N = 80$, scaling factor $F = 0.6$ and crossover rate $CR = 0.95$. DE/rand/1/exp is adopted in DE with the estimation comparison method. The parameter p for U_o and U_c in Eqs. (5.5) and (5.6) is 2. In the estimated comparison method, the initial value of the margin parameter $\delta(0) = \delta_{\max} = 0.4$ and the initial value of the congestion parameter $\lambda(0) = \lambda_{\max} = 0.2$, and the parameters are adaptively controlled according to Table 5.1. The fixed values of the margin parameter and the congestion parameter, $(0.2, 0.4)$ and $(0.1, 0.2)$, are also examined for comparison.

In this study, 25 independent runs are performed. In each run, the optimization is terminated when a near optimal solution of which function value is less than or equal to 1×10^{-3} is found.

5.5.3 Experimental Results

Table 5.3 shows the results of optimization. The column labeled "Func." shows the function name optimized, and "Method" shows optimization method where

Table 5.3 Comparison among adaptive DE with the estimated comparison method, DE with the estimated comparison method, and DE

Func.	Method	eval	success	fail	rate(%)	reduce(%)
f_1	adaptive	**36,256.68**	13,224.08	22,950.88	**36.56**	**57.67**
	(0.2, 0.4)	46,324.32	14,649.48	31,592.96	31.68	45.92
	(0.1, 0.2)	41,344.04	14,157.88	27,104.32	34.31	51.73
	DE	85,656.16	15,357.96	70,216.20	17.95	0
f_2	adaptive	586,018.92	65,552.36	520,384.56	11.19	25.85
	(0.2, 0.4)	**578,136.68**	58,460.00	519,594.72	10.11	**26.84**
	(0.1, 0.2)	617,305.88	87,315.84	529,908.20	**14.15**	21.89
	DE	790,263.36	45,467.08	744,714.28	5.75	0
f_3	adaptive	586,919.64	66,579.04	520,258.72	11.35	28.23
	(0.2, 0.4)	**582,880.12**	59,880.64	522,917.60	10.27	**28.73**
	(0.1, 0.2)	620,157.24	86,223.28	533,852.04	**13.91**	24.17
	DE	817,802.48	45,482.80	772,237.68	5.56	0
f_4	adaptive	304,002.68	22,342.08	281,578.76	7.35	45.42
	(0.2, 0.4)	313,046.68	23,243.68	289,721.12	7.43	43.79
	(0.1, 0.2)	**295,009.04**	22,895.68	272,031.48	**7.76**	**47.03**
	DE	556,960.80	24,504.44	532,374.36	4.40	0
f_5	adaptive	**69,175.84**	23,521.36	45,572.80	**34.04**	**55.61**
	(0.2, 0.4)	83,019.20	25,245.48	57,691.84	30.44	46.72
	(0.1, 0.2)	75,401.76	24,669.00	50,651.08	32.75	51.61
	DE	155,823.64	26,636.68	129,104.96	17.10	0
f_6	adaptive	**56,982.00**	19,982.08	36,918.08	**35.12**	**56.05**
	(0.2, 0.4)	71,336.48	21,822.72	49,431.88	30.63	44.98
	(0.1, 0.2)	63,453.12	21,104.56	42,266.72	33.30	51.06
	DE	129,658.04	22,888.72	106,687.32	17.66	0

"adaptive" means the adaptive DE with the estimated comparison method using potential model, "DE" means original DE/rand/1/exp, and others mean DE with the estimated comparison method using fixed parameter values (δ, λ) specified in the table. The columns labeled "eval", "success", "fail" and "rate" show the total number of evaluation until a near optimal solution is found, the number of successful evaluations where the child solution is better than the parent solution, the number of failure evaluations and the success rate on average, respectively. The column "reduce" shows the ratio of how many times function evaluations is reduced compared with DE.

The function f_1 is a unimodal and smooth function. It is easy to approximate the function. The adaptive DE with the estimated comparison method achieved the best result and reduced 57.67% of function evaluations compared with DE.

The functions f_2 and f_3 are unimodal but steep functions. It is difficult to approximate the functions. The DE with the estimated comparison method using $(\delta, \lambda) = (0.2, 0.4)$ reduced 26.84% (f_2) and 28.73% (f_3) of function evaluations compared with DE and achieved the best result. The adaptive DE with estimated comparison method reduced 25.85% (f_2) and 28.23% (f_3) of function evaluations.

It achieved the second best result and the reduction rate is almost same as the best result.

The function f_4 is a multimodal and very bumpy function. It is difficult to approximate the function. The DE with the estimated comparison method using $(\delta, \lambda) = (0.1, 0.2)$ reduced 47.03% of function evaluations compared with DE and achieved the best result. The adaptive DE with the estimated comparison method reduced 45.42% of function evaluations. It achieved the second best result and the reduction rate is almost same as the best result.

The function f_5 is a multimodal and bumpy function. It is not so difficult to approximate the function. The adaptive DE with the estimated comparison method reduced 55.61% of function evaluations compared with DE and achieved the best result.

The function f_6 is a multimodal and less bumpy function. It is fairly easy to approximate the function. The adaptive DE with the estimated comparison method reduced 56.05% of function evaluations compared with DE and achieved the best result.

It is shown that the adaptive and original DE with the estimated comparison method are far better than original DE in all problems. Also, it is shown that the adaptive DE with the estimated comparison method could reduce from 25% to 57% function evaluations without hand-tuning the parameters δ and λ and achieved better or almost same results compared with the estimated comparison method with hand-tuning the parameters.

Figures 5.2, 5.3, 5.4, 5.5, 5.6 and 5.7 show single logarithmic plots of the best function values (left figures) and adaptively controlled parameter values of δ and λ (right figures) over the number of function evaluations for function f_1, f_2, f_3, f_4, f_5 and f_6, respectively. Note that the ends of graphs are violated because some runs are terminated earlier than some other runs when the near optimal solution is found. In the figures of the best values, thick solid lines, thin solid lines and dotted lines show optimization process by adaptive, $(0.2, 0.4)$ and $(0.1, 0.2)$ DE with the estimated comparison method, and chain lines show that by DE. In the figures of the parameter values, solid lines and dotted lines show the values of δ and λ controlled by the adaptive DE with the estimated comparison method.

It is clear that the adaptive and original DE with the estimated comparison method can find better solutions faster than DE. Also, it is clear that the adaptive control can adjust the parameter values properly and dynamically. The function f_1 can be approximated easily. As shown in the right of Fig. 5.2, the parameter values of δ and λ are small and are often decreased to the minimum values. The approximation of the functions f_2 and f_3 is very difficult. As shown in the right of Figs. 5.3 and 5.4, the parameter values are large and are often increased to the maximum values. The approximation of f_4 is difficult because the function is multimodal. However, after the valley of the function surface including the minimum value has found, f_4 becomes unimodal and the approximation becomes easy. As shown in the right of Fig. 5.5, the parameter values are large in the early stage of the search process and become small in the last stage. The approximation of f_5 and f_6 is not so difficult. As shown in the right of Figs. 5.6 and 5.7, the parameter values are almost in the

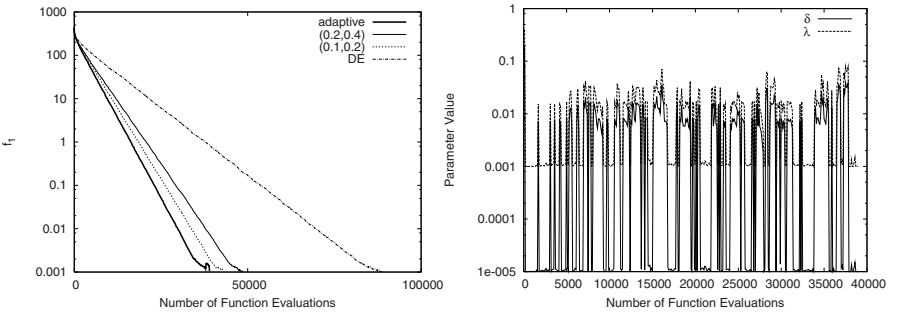


Fig. 5.2 Optimization of f_1

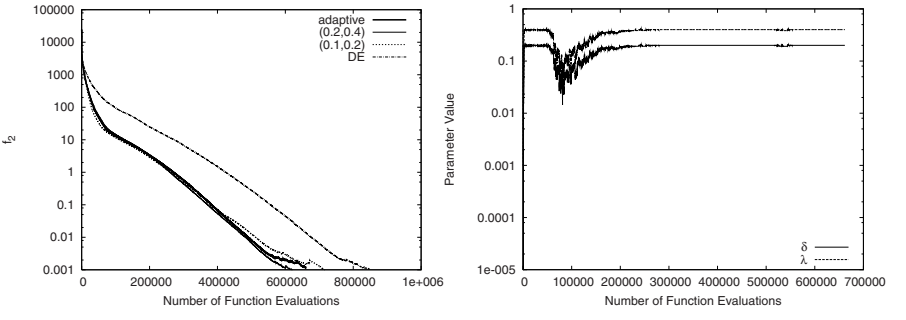


Fig. 5.3 Optimization of f_2

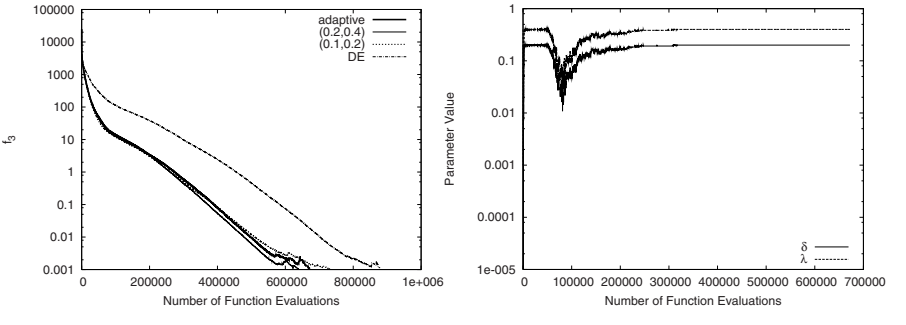


Fig. 5.4 Optimization of f_3

lower range of $[0.01, 0.1]$ than the fixed case of $(\delta, \lambda) = (0.2, 0.1)$ and more number of function evaluations can be skipped than the fixed case. The parameter values in f_6 are often lower than those in f_5, because f_6 is less bumpy than f_5. Thus, it is thought that the parameter values are properly controlled according to the difficulty of approximation for the objective functions.

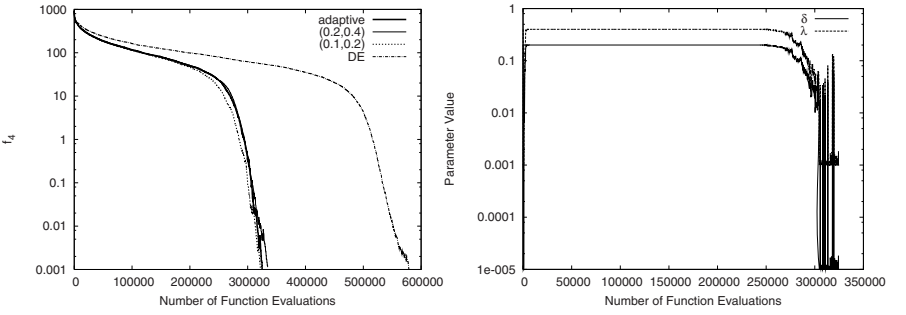


Fig. 5.5 Optimization of f_4

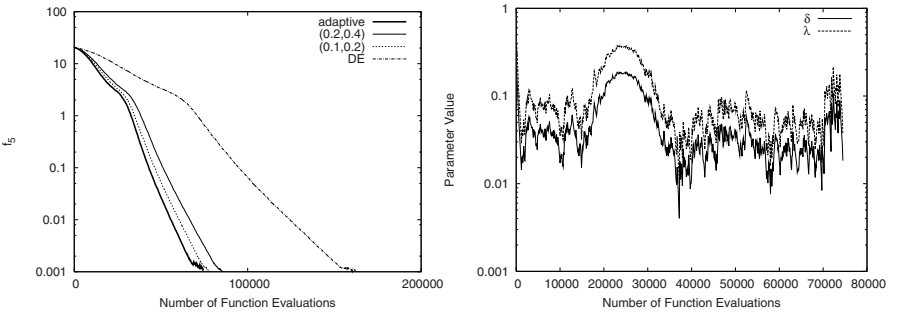


Fig. 5.6 Optimization of f_5

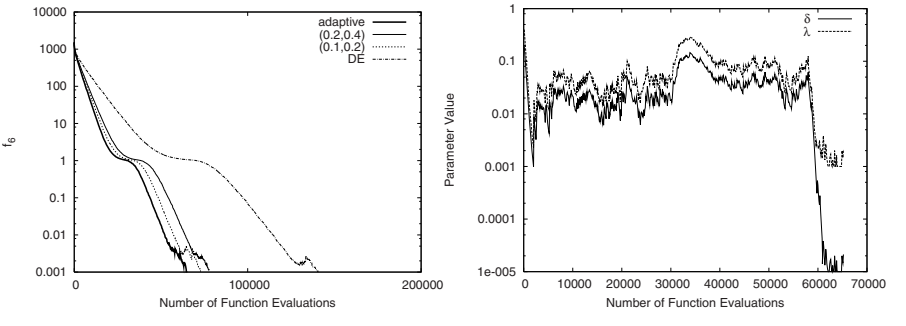


Fig. 5.7 Optimization of f_6

5.6 Discussion

In this section, the adaptive DE with the estimated comparison is compared with particle swarm optimization (PSO), because PSO is known as a fast and efficient optimization algorithm. In this study, the standard Particle Swarm Optimization 2007

(SPSO-07) [3], which is a recent standard PSO acknowledged by PSO community, is used for the comparison.

Searching procedures by PSO can be described as follows: A group of particles optimizes the objective function. At any time t, each particle i knows its position $\boldsymbol{x}_i(t)$ and the velocity $\boldsymbol{v}_i(t)$. It remembers its best previous position $\boldsymbol{pb}_i(t)$. It also knows the best position of the best particle among its neighbors, $\boldsymbol{nb}_i(t)$. The neighbors are defined by the topology of the particles.

Initial positions are generated randomly inside the search space. In SPSO-07, initial velocity is defined as the half-difference of two random positions. The particles are updated as follows:

$$v_{ij}(t+1) = wv_{ij}(t) + u(0,c_1)(pb_{ij}(t) - x_{ij}(t)) + u(0,c_2)(nb_{ij}(t) - x_{ij}(t)), \quad (5.16)$$

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \quad (5.17)$$

where w is an inertia weight, c_1 is a cognitive parameter, c_2 is a social parameter, and $u(0,c)$ returns a random value in $[0,c]$ according to the uniform distribution. In SPSO-07, random topology is employed: Each particle chooses a few informants (K informants) at random, selects the best one from them as the best particle. If it finds no particle better than itself, it becomes the best particle.

In the experiment, 25 independent runs are performed. In each run, the optimization is terminated when a near optimal solution of which function value is less than or equal to 1×10^{-3} is found. Also, the run is terminated when the number of function evaluations exceeds 1,000,000. The default parameter settings for SPSO-07 are as follows: The swarm size that is the number of particles is $\lfloor 10 + 2\sqrt{n} \rfloor = 24$, $K = 3$, $w = 1/(2\ln 2) \approx 0.721348$ and $c_1 = c_2 = 0.5 + \ln 2 \approx 1.193147$. The swarm size is changed to 100, because the swarm size 100 attained most stable result among the swarm sizes 24, 50, 80 and 100.

Table 5.4 shows the experimental results of the adaptive DE and SPSO-07. The column labeled "success" shows the ratio of successful runs when the near optimal value was found over 25 runs, "eval" shows the average number of function evaluations in the successful runs, and "best" shows the average of the best objective values over 25 runs.

As for the functions f_2, f_3 and f_4, SPSO-07 can not find the near optimal solution within 1,000,000 function evaluations. On the contrary, the adaptive DE can find the near optimal solutions in all runs. As for the functions f_1 and f_5, the both methods can find the near optimal solutions stably. The adaptive DE can find the solutions with fewer number of function evaluations than SPSO-07. As for the function f_6, SPSO-07 failed to find the near optimal solution in two runs. On the contrary, the adaptive DE can find the near optimal solutions in all runs. Also, the adaptive DE can find the solution with fewer number of function evaluations than SPSO-07. Thus, it is thought that the adaptive DE can find the near optimal solutions more stably and faster than SPSO-07.

Table 5.4 Comparison between adaptive DE with the estimated comparison method and SPSO-07

Func.	Method	success	eval	best
f_1	adaptive	**100%**	**36,256.68**	**0.001**
	SPSO-07	**100%**	38,956.00	**0.001**
f_2	adaptive	100%	586,018.92	**0.001**
	SPSO-07	0%	—	2.528
f_3	adaptive	100%	**586,919.64**	**0.001**
	SPSO-07	0%	—	2.534
f_4	adaptive	100%	304,002.68	**0.001**
	SPSO-07	0%	—	67.059
f_5	adaptive	**100%**	**69,175.84**	**0.001**
	SPSO-07	**100%**	72,016.00	**0.001**
f_6	adaptive	**100%**	**56,982.00**	**0.001**
	SPSO-07	92%	61,721.74	0.002

5.7 Conclusions

We proposed to utilize a rough approximation model, which is an approximation model with low accuracy and without learning process, in order to reduce the number of function evaluations in wide range of problems including from low or medium computation cost to high computation cost problems. We proposed the estimated comparison method, in which the function evaluation of a solution is skipped when the goodness of the solution can be judged from the approximation value of it. Also, we proposed to control the margin parameter and the congestion parameter adaptively in the estimated comparison method. Through the optimization of various types of test problems, it is shown that the estimated comparison method is very effective to reduce function evaluations. Also, it is shown that without tuning the parameters the adaptive DE with the estimated comparison method can improve the optimization process and reduce about from 25% to 57% function evaluations compared with DE.

In the future, we will apply the estimated comparison method into constrained optimization problems using the ε constrained Differential Evolution (εDE) [16, 19]. We have shown some results of constrained optmization using the estimated comparison method and εDE in [18]. We plan to apply the estimated comparison method into other evolutionary algorithms such as particle swarm optimization. Also, we will apply the estimated comparison method to real world problems, and test the performance of the method.

References

1. Büche, D., Schraudolph, N.N., Koumoutsakos, P.: Accelerating evolutionary algorithms with gaussian process fitness function models. IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews 35(2), 183–194 (2005)
2. Chakraborty, U.K. (ed.): Advances in Differential Evolution. Springer, Heidelberg (2008)
3. Clerc, M.: Standard_pso_2007.c (2007), http://www.particleswarm.info/

4. Giunta, A., Watson, L.: A comparison of approximation modeling techniques: Polynomial versus interpolating models. Tech. Rep. 98-4755, AIAA (1998)
5. Guimarães, F.G., Wanner, E.F., Campelo, F., Takahashi, R.H., Igarashi, H., Lowther, D.A., Ramírez, J.A.: Local learning and search in memetic algorithms. In: Proc. of the 2006 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, pp. 9841–9848 (2006)
6. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing 9, 3–12 (2005)
7. Jin, Y., Sendhoff, B.: Reducing fitness evaluations using clustering techniques and neural network ensembles. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 688–699. Springer, Heidelberg (2004)
8. Jin, Y., Olhofer, M., Sendhoff, B.: On evolutionary optimization with approximate fitness functions. In: Proc. of the Genetic and Evolutionary Computation Conference, pp. 786–792. Morgan Kaufmann, San Francisco (2000)
9. Jin, Y., Olhofer, M., Sendhoff, B.: A framework for evolutionary optimization with approximate fitness functions. IEEE Trans. on Evolutionary Computation 6(5), 481–494 (2002)
10. Ong, Y.S., Zhou, Z., Lim, D.: Curse and blessing of uncertainty in evolutionary algorithm using approximation. In: Proc. of the 2006 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, pp. 9833–9840 (2006)
11. Shyy, W., Tucker, P.K., Vaidyanathan, R.: Response surface and neural network techniques for rocket engine injector optimization. Tech. Rep. 99-2455, AIAA (1999)
12. Simpson, T.W., Mauery, T.M., Korte, J.J., Mistree, F.: Comparison of response surface and kriging models in the multidisciplinary design of an aerospike nozzle. Tech. Rep. 98-4758, AIAA (1998)
13. Storn, R., Price, K.: Minimizing the real functions of the ICEC 1996 contest by differential evolution. In: Proc. of the International Conference on Evolutionary Computation, pp. 842–844 (1996)
14. Storn, R., Price, K.: Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)
15. Takahama, T., Sakai, S.: Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites. In: Proc. of the 2006 IEEE Congress on Evolutionary Computation, pp. 308–315 (2006)
16. Takahama, T., Sakai, S.: Constrained optimization by the ε constrained differential evolution with dynamic ε-level control. In: Chakraborty, U. (ed.) Advances in Differential Evolution. Springer, Heidelberg (2008)
17. Takahama, T., Sakai, S.: Reducing function evaluations in differential evolution using rough approximation-based comparison. In: Proc. of the 2008 IEEE Congress on Evolutionary Computation, pp. 2307–2314 (2008)
18. Takahama, T., Sakai, S.: Efficient constrained optimization by the ε constrained differential evolution using an approximation model with low accuracy. Transactions of the Japanese Society for Artificial Intelligence 24(1), 34–45 (2009) (in Japanese)
19. Takahama, T., Sakai, S., Iwane, N.: Solving nonlinear constrained optimization problems by the ε constrained differential evolution. In: Proc. of the 2006 IEEE Conference on Systems, Man, and Cybernetics, pp. 2322–2327 (2006)
20. Takahama, T., Sakai, S., Hara, A.: Reducing the number of function evaluations in differential evolution by estimated comparison method using an approximation model with low accuracy. IEICE Trans. on Information and Systems J91-D(5), 1275–1285 (2008) (in Japanese)

Chapter 6
Kriging Is Well-Suited to Parallelize Optimization

David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro

Abstract. The optimization of expensive-to-evaluate functions generally relies on metamodel-based exploration strategies. Many deterministic global optimization algorithms used in the field of computer experiments are based on Kriging (Gaussian process regression). Starting with a spatial predictor including a measure of uncertainty, they proceed by iteratively choosing the point maximizing a criterion which is a compromise between predicted performance and uncertainty. Distributing the evaluation of such numerically expensive objective functions on many processors is an appealing idea. Here we investigate a multi-points optimization criterion, the *multipoints expected improvement* (q-$\mathbb{E}I$), aimed at choosing several points at the same time. An analytical expression of the q-$\mathbb{E}I$ is given when $q = 2$, and a consistent statistical estimate is given for the general case. We then propose two classes of heuristic strategies meant to approximately optimize the q-$\mathbb{E}I$, and apply them to the classical Branin-Hoo test-case function. It is finally demonstrated within the covered example that the latter strategies perform as good as the best Latin Hypercubes and Uniform Designs ever found by simulation (2000 designs drawn at random for every $q \in [1, 10]$).

David Ginsbourger
Département 3MI, Ecole Nationale Supérieure des Mines, 158 cours Fauriel, Saint-Etienne, France
e-mail: ginsbourger@emse.fr

Rodolphe Le Riche
CNRS (UMR 5146) and Département 3MI, Ecole Nationale Supérieure des Mines, 158 cours Fauriel, Saint-Etienne, France
e-mail: leriche@emse.fr

Laurent Carraro
Département 3MI, Ecole Nationale Supérieure des Mines, 158 cours Fauriel, Saint-Etienne, France
e-mail: carraro@emse.fr

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 131–162.
springerlink.com

6.1 Introduction

6.1.1 Motivations: Efficient Optimization Algorithms for Expensive Computer Experiments

Beyond both estalished frameworks of derivative-based descent and stochastic search algorithms, the rise of expensive optimization problems creates the need for new specific approaches and procedures. The word "expensive" —which refers to price and/or time issues— implies severely restricted budgets in terms of objective function evaluations. Such limitations contrast with the computational burden typically associated with stochastic search techniques, like genetic algorithms. Furthermore, the latter evaluations provide no differential information in a majority of expensive optimization problems, whether the objective function originate from physical or from simulated experiments. Hence there exists a strong motivation for developing derivative-free algorithms, with a particular focus on their optimization performances in a drastically limited number of evaluations. Investigating and implementing adequate strategies constitute a contemporary challenge at the interface between Applied Mathematics and Computational Intelligence, especially when it comes to reducing optimization durations by efficiently taking advantage of parallel computation facilities.

The primary aim of this chapter is to address parallelization issues for the optimization of expensive-to-evaluate simulators, such as increasingly encountered in engineering applications like car crash tests, nuclear safety, or reservoir forecasting. More specifically, the work presented here takes place in the frame of metamodel-based design of computer experiments, in the sense of [42]. Even though the results and discussions might be extended to a more general scope, we restrict ourself here for clarity to single-objective optimization problems for deterministic codes. The simulator is seen as black-box function y with d-dimensional vector of inputs and scalar output, the latter being often obtained as combination of several responses. *Metamodels*, also called *surrogate models*, are simplified representations of y. They can be used for predicting values of y outside the initial design, or visualizing the influence of each variable on y [27, 43]. They may also guide further sampling decisions for various purposes, such as refining the exploration of the input space in preferential zones or optimizing the function y [22]. Classical surrogates include radial basis functions [37], interpolation splines [52], neural nets [8] (*deterministic* metamodels), or linear and non-linear regression [2], and Kriging [7] (*probabilistic* metamodels). We concentrate here on the advantages of probabilistic metamodels for parallel exploration and optimization, with a particular focus on the virtues of Kriging.

6.1.2 Where Computational Intelligence and Kriging Meet

Computational intelligence (CI) methods share, in various proportions, four features:

An History Going from Experiments to Theory: CI methods very often originate from empirical computing experiments, in particular from experiments that mimick natural processes (e.g., neural networks [4], ant colony optimization [5], simulated annealing [25]). Later on, as researchers use and analyze them, theory develops and their mathematical content grows. A good example is provided by the evolutionary algorithms [9] which have progressively mixed the genetic metaphor and stochastic optimization theory.

An Indirect Problem Representation: In standard evolutionary optimization methods, knowledge about the cost function takes the indirect form of a set of well-performing points, known as "current population". Such set of points is an implicit, partial, representation of a function. In fuzzy methods, the probability density functions of the uncertain variables are averaged out. Such indirect representations enable to work with few mathematical assumptions and have broadened the range of applicability of CI methods.

Parallelized Decision Process: Most CI approaches are inherently parallel. For example, the evolutionary or particle swarm optimization [24] methods process sets of points in parallel. Neural networks have an internal parallel structure. Today, parallelism is crucial for taking advantage of the increasingly distributed computing capacity. The parallel decision making possibilities are related to the indirect problem representations (through set of points, distributions) and to the use of randomness in the decision process.

Heuristics: Implicit problem representations and the empirical genesis of the CI methods rarely allow mathematical proofs of the methods properties. Most CI methods are thus *heuristics.*

Kriging has recently gained popularity among several research communities related to CI, ranging from *Data Mining* [16] and *Bayesian Statistics* [34, 48] to *Machine Learning* [39], where it is linked to *Gaussian Process Regression* [53] and *Kernel Methods* [12]. Recent works [17, 30, 31] illustrate the practical relevance of Kriging to approximate computer codes in application areas such as aerospace engineering or materials science. Indeed, probabilistic metamodels like Kriging seem to be particularly adapted for the optimization of black-box functions, as analyzed and illustrated in the excellent article [20]. The current Chapter is devoted to the optimization of black-box functions using a Kriging metamodel [14, 22, 49, 51]. Let us now stress some essential relationships between Kriging and CI by revisiting the above list of features.

A History from Field Studies to Mathematical Statistics: Kriging comes from the earth sciences [29, 33], and has been progressively developed since the 1950's along with the discipline called *geostatistics* [23, 32]. Originally aimed at estimating natural ressources in mining applications, it has later been adapted to address very general interpolation and approximation problems [42, 43]. The word "Kriging" comes from the name of a mining engineer, Prof. Daniel G. Krige, who was a pioneer in the application of mathematical statistics to the study of new gold mines using a limited number of boreholes [29].

An Indirect Representation of the Problem: As will be detailed later in the text, the Kriging metamodel has a powerful interpretation in terms of stochastic process conditioned by observed data points. The optimized functions are thus indirectly represented by stochastic processes.

Parallelized Decision Process: The central contribution of this chapter is to propose tools enabling parallelized versions of state-of-the art Kriging-based optimization algorithms.

Heuristics: Although the methods discussed here are mathematically founded on the multipoints expected improvement, the maximization of this criterion is not mathematically tractable beyond a few dimensions. In the last part of the chapter, it is replaced by the "Kriging believer" and the "constant liar" heuristics.

Through their indirect problem representation, their parallelism and their heuristical nature, the Kriging-based optimization methods presented hereafter are Computational Intelligence methods.

6.1.3 Towards Kriging-Based Parallel Optimization: Summary of Obtained Results and Outline of the Chapter

This chapter is a follow-up to [14]. It proposes metamodel-based optimization criteria and related algorithms that are well-suited to parallelization since they yield several points at each iteration. The simulations associated with these points can be distributed on different processors, which helps performing the optimization when the simulations are calculation intensive. The algorithms are derived from a multi-points optimization criterion, the *multi-points* or *q-points expected improvement* (q-EI). In particular, an analytic expression is derived for the 2-EI, and consistent statistical estimates relying on Monte-Carlo methods are provided for the general case. All calculations are performed in the framework of Gaussian processes (**GP**). Two classes of heuristic strategies, the *Kriging Believer* (**KB**) and *Constant Liar* (**CL**), are subsequently introduced to obtain approximately q-EI-optimal designs. The latter strategies are tested and compared on a classical test case, where the *Constant Liar* appears to constitute a legitimate heuristic optimizer of the q-EI criterion. Without too much loss of generality, the probabilistic metamodel considered is Ordinary Kriging (**OK**, see eqs. 6.1,6.2,6.35), like in the founder work [22] introducing the now famous **EGO** algorithm. In order to make this document self-contained, non-specialist readers may find an overview of existing criteria for Kriging-based sequential optimization in the next pages, as well as a short but dense introduction to GP and OK in the body of the chapter, with complements in appendix. The outline of the chapter is as follows:

- Section 6.2 (*Background in Kriging for Sequential Optimization*) recalls the OK equations, with a focus on the joint conditional distributions associated with this probabilistic metamodel. A progressive introduction to Kriging-based criteria for sequential optimization is then proposed, culminating with the presentation of the EGO algorithm and its obvious limitations in a context of distributed computing.

- Section 6.3 (*The Multi-points Expected Improvement*) consists in the presentation of the q-EI criterion —continuing the work initiated in [47]—, its explicit calculation when $q = 2$, and the derivation of estimates of the latter criterion in the general case, relying on Monte-Carlo simulations of gaussian vectors.
- Section 6.4 (*Approximated q-EI maximization*) introduces two heuristic strategies, KB and CL, to circumvent the computational complexity of a direct q-EI maximization. These strategies are tested on a classical test-case, and CL is found to be a very promizing competitor for approximated q-EI maximization
- Section 6.5 (*Towards Kriging-based Parallel Optimization: Conclusion and Perspectives*) gives a summary of obtained results as well as some related practical recommendations, and finally suggests what the authors think are perspectives of research to address the most urgently in order to extend this work.
- The appendix 6.6 is a short but dense introduction to GP for machine learning, with an emphasis on the foundations of both Simple Kriging and Ordinary Kriging by GP conditioning.

Some Notations: $y : \mathbf{x} \in D \subset \mathbb{R}^d \rightarrow y(\mathbf{x}) \in \mathbb{R}$ refers to the objective function, where $d \in \mathbb{N}\backslash\{0\}$ is the number of input variables and D is the set in which the inputs vary, most of the time assumed to be a compact and connex[1] subset of $\mathbb{R}^d$. At first, y is known at a *Design of Experiments* $\mathbf{X} = \{\mathbf{x}^1, ..., \mathbf{x}^n\}$, where $n \in \mathbb{N}$ is the number of initial runs or experiments, and each $\mathbf{x}^i$ $(1 \leq i \leq n)$ is hence a d-dimensional vector $(\mathbf{x}^i_1, \ldots, \mathbf{x}^i_d)$. We denote by $\mathbf{Y} = \{y(\mathbf{x}^1), ..., y(\mathbf{x}^n)\}$ the set of observations made by evaluating y at the points of $\mathbf{X}$. The data $(\mathbf{X}, \mathbf{Y})$ provides information on which is initially based the metamodeling of y, with an accuracy that depends on n, the geometry of $\mathbf{X}$, and the regularity of y. The OK mean predictor and prediction variance are denoted by the functions $m_{OK}(.)$ and $s^2_{OK}(.)$. The random process implicitely underlying OK is denoted by $Y(.)$, in accordance with the notations of eq. (6.35) presented in appendix. The symbol "|" is used for conditioning, together with the classical symbols for probability and expectation, respectively $\mathbb{P}$ and $\mathbb{E}$.

6.2 Background in Kriging for Sequential Optimization

6.2.1 The Ordinary Kriging Metamodel and Its Gaussian Process Interpretation

OK is the most popular Kriging metamodel, simultaneously due to its great versatility and applicability. It provides a mean predictor of spatial phenomena, with a quantification of the expected prediction accuracy at each site. A full derivation of the OK mean predictor and variance in a GP setting is proposed in the appendix. The corresponding OK mean and variance functions are given by the following formulae:

[1] Connexity is sometimes untenable in practical applications, see e.g. [46] for a treatment of disconnected feasible regions.

$$m_{OK}(\mathbf{x}) = \left[\mathbf{c}(\mathbf{x}) + \left(\frac{1-\mathbf{c}(\mathbf{x})^T\Sigma^{-1}\mathbf{1}_n}{\mathbf{1}_n^T\Sigma^{-1}\mathbf{1}_n}\right)\mathbf{1}_n\right]^T \Sigma^{-1}\mathbf{Y}, \tag{6.1}$$

$$s^2_{OK}(\mathbf{x}) = \sigma^2 - \mathbf{c}(\mathbf{x})^T\Sigma^{-1}\mathbf{c}(\mathbf{x}) + \frac{(1-\mathbf{1}_n^T\Sigma^{-1}\mathbf{c}(\mathbf{x}))^2}{\mathbf{1}_n^T\Sigma^{-1}\mathbf{1}_n}, \tag{6.2}$$

where $\mathbf{c}(x) := \left(c(Y(\mathbf{x}),Y(\mathbf{x}^1)),...,c(Y(\mathbf{x}),Y(\mathbf{x}^n))\right)^T$, and Σ and σ^2 are defined following the assumptions[2] and notations given in appendix 6.6. Classical properties of OK include that $\forall i \in [1,n]\ m_{OK}(\mathbf{x}^i) = y(\mathbf{x}^i)$ and $s^2_{OK}(\mathbf{x}^i) = 0$, therefore $[Y(\mathbf{x})|Y(\mathbf{X}) = \mathbf{Y}]$ is interpolating. Note that $[Y(\mathbf{x}^a)|Y(\mathbf{X}) = \mathbf{Y}]$ and $[Y(\mathbf{x}^b)|Y(\mathbf{X}) = \mathbf{Y}]$ are dependent random variables, where $\mathbf{x}^a$ and $\mathbf{x}^b$ are arbitrary points of D, as we will develop later.

The OK metamodel of the Branin-Hoo function (Cf. eq. 6.25) is plotted on fig. 6.2.1. The OK interpolation (upper middle) is based only on 9 observations. Even if the shape is reasonably respected (lower middle), the contour of the mean shows an artificial optimal zone (upper middle, around the point $(6,2)$). In other respects, the variance is not depending on the observations[3] (eq. 6.2). Note its particular shape, due to the anisotropy of the covariance kernel estimated by likelihood maximization. In modern interpretations [39], deriving OK equations is often based on the assumption that y is a realization of a random process Y with unknown constant mean and known covariance (see [1] or [12] for a review of classical covariance kernels). Here we follow the derivation of 6.6.4, which has the advantage of delivering a gaussian posterior distribution:

$$[Y(\mathbf{x})|Y(\mathbf{X}) = \mathbf{Y}] \sim \mathcal{N}(m_{OK}(\mathbf{x}), s^2_{OK}(\mathbf{x})) \tag{6.3}$$

Note that both a structure selection and a parametric estimation are made in practice: one often chooses a generalized exponential kernel with plugged-in maximum likelihood covariance hyperparameters, i.e. without taking the estimation variance into account [22]. This issue is sometimes addressed using a full bayesian treatment, as can be found in [43], or more recently in [15, 34, 39]. Rephrasing eq. 6.3, under the latter GP assumptions, the random variable $Y(\mathbf{x})$ knowing the values of $\{y(\mathbf{x^1}),...,y(\mathbf{x^n})\}$ follows a gaussian distribution which mean and variance are respectively $\mathbb{E}[Y(\mathbf{x})|Y(\mathbf{X}) = \mathbf{Y}] = m_{OK}(\mathbf{x})$ and $Var[Y(\mathbf{x})|Y(\mathbf{X}) = \mathbf{Y}] = s^2_{OK}(\mathbf{x})$. In fact, as shown in appendix (Cf. eq. 6.38), one can even get much more than these marginal conditional distributions; $Y(.)|Y(\mathbf{X}) = \mathbf{Y}$ constitutes a random process

[2] An extension of the Kriging equations to the framework of covariance non-stationary processes [35] is straightforward but beyond the scope of the present work.

[3] Phenomenon known as homoskedasticity of the Kriging variance with respect to the observations [7].

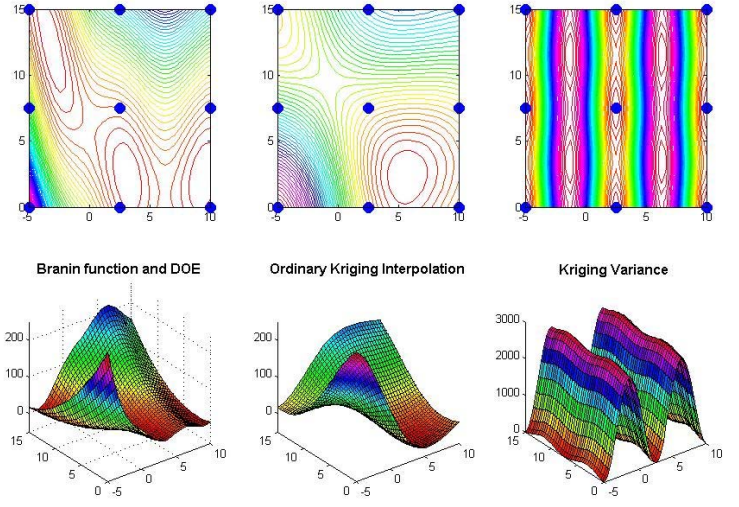


Fig. 6.1 Ordinary Kriging of the Branin-Hoo function (function, Kriging mean value and variance, from left to right). The design of experiments is a 3×3 factorial design. The covariance is an anisotropic squared exponential with parameters estimated by gaussian likelihood maximization [7]

which is itself gaussian, and as such completely characterized by its conditional mean, m_{OK}, and conditional covariance kernel c_{OK} explicited herunder:

$$[Y(\ )|Y(\mathbf{X}) = \mathbf{Y}] \sim GP(m_{OK}(\ )\ c_{OK}(\quad)) \tag{6.4}$$

$$\text{where } c_{OK}(\mathbf{x}\ \mathbf{x}') = c(\mathbf{x}-\mathbf{x}') - \mathbf{c}(\mathbf{x})^T\Sigma^{-1}\mathbf{c}(\mathbf{x}') + \sigma^2\left[\frac{(1-\mathbf{1}_n^T\Sigma^{-1}\mathbf{c}(\mathbf{x}))(1-\mathbf{1}_n^T\Sigma^{-1}\mathbf{c}(\mathbf{x}'))}{\mathbf{1}_n^T\Sigma^{-1}\mathbf{1}_n}\right] \tag{6.5}$$

This new kernel c_{OK} is not stationary, even if c is. In other respects, the knowledge of m_{OK} and c_{OK} is the first step to performing conditional simulations of Y knowing the observations $Y(\mathbf{X}) = \mathbf{Y}$, which is easily feasible at any new finite design of experiments, whatever the dimension of inputs. This will enable the computation of any multi-points sampling criterion, such as proposed in the forthcoming section about parallelization.

6.2.2 *Kriging-Based Optimization Criteria*

GP metamodels [39, 53] such as OK has been used for optimization (minimization, by default). There is a detailed review of optimization methods relying on a

metamodel in [44, 45] or [20]. The latter analyzes why directly optimizing a deterministic metamodel (like a spline, a polynomial, or the Kriging mean) is dangerous, and does not even necessarily lead to a local optimum. Kriging-based sequential optimization strategies (as developed in [22], and commented in [20]) address the issue of converging to non (locally) optimal points, by taking the Kriging variance term into account (hence encouraging the algorithms to explore outside the already visited zones). Such algorithms produce one point at each iteration that maximizes a figure of merit based upon $[Y(\mathbf{x})|Y(\mathbf{X}) = \mathbf{Y}]$. In essence, the criteria balance Kriging mean prediction and uncertainty.

6.2.2.1 Visiting the Point with Most Promizing Mean: Minizing m_{OK}

When approximating y by m_{OK}, it might seem natural to hope that minimizing m_{OK} instead of y brings satisfying results. However, a function and its approximation (m_{OK} or other) can show substantial differences in terms of optimal values and optimizers. More specifically, depending on the kind of covariance kernel used in OK, the minimizer of m_{OK} is susceptible to lie at (or near to) the design point with minimal y value. Taking the geometry of the design of experiments and space-filling considerations into account within exploration criteria then makes sense. The Kriging variance can be of providential help for this purpose.

6.2.2.2 Visiting the Point with Highest Uncertainty: Maximizing s_{OK}

A fundamental mistake of minimizing m_{OK} is that no account is done of the uncertainty associated with it. At the extreme inverse, it is possible to define the next optimization iterate as the least known point in D,

$$\mathbf{x}' = argmax_{\mathbf{x}\in D}\, s_{OK}(\mathbf{x}) \tag{6.6}$$

This procedure defines a series of $\mathbf{x}'$s which will fill the space D and hence ultimately locate a global optimum. Yet, since no use is made of previously obtained $\mathbf{Y}$ information —look at formula 6.2 for s^2_{OK}—, there is no bias in favor of high performance regions. Maximizing the uncertainty is inefficient in practice.

6.2.2.3 Compromizing between m_{OK} and s_{OK}

The most general formulation for compromizing between the exploitation of previous simulations brought by m_{OK} and the exploration based on s_{OK} is the multicriteria problem

$$\begin{cases} \min_{\mathbf{x}\in D} m_{OK}(\mathbf{x}) \\ \max_{\mathbf{x}\in D} s_{OK}(\mathbf{x}) \end{cases} \tag{6.7}$$

Let $\mathscr{P}$ denote the Pareto set of solutions[4]. Finding one (or many) elements in $\mathscr{P}$ remains a difficult problem since $\mathscr{P}$ typically contains an infinite number of points. A comparable approach called *direct*, although not based on OK, is described in

[4] Definition of the Pareto front of $(s_{OK}, -m_{OK})$: $\forall x \in \mathscr{P}, \nexists\ z \in D : (m_{OK}(z) < m_{OK}(x)$ and $s_{OK}(z) \geq s_{OK}(x))$ or $(m_{OK}(z) \leq m_{OK}(x)$ and $s_{OK}(z) > s_{OK}(x))$.

[21]: the metamodel is piecewise linear and the uncertainty measure is a distance to already known points. The space D is discretized and the Pareto optimal set defines areas where discretization is refined. The method becomes computationally expensive as the number of iterations and dimensions increase. Note that [3] proposes several parallelized versions of *direct*.

6.2.2.4 Maximizing the Probability of Improvement

Among the numerous criteria presented in [20], the probability of getting an improvement of the function with respect to the past evaluations seems to be one of the most fundamental. This function is defined for every $\mathbf{x} \in D$ as the probability for the random variable $Y(\mathbf{x})$ to be below the currently known minimum $min(\mathbf{Y}) = \min\{y(\mathbf{x}^1), ..., y(\mathbf{x}^n)\}$ conditional on the observations at the design of experiments:

$$PI(\mathbf{x}) := P(Y(\mathbf{x}) \leq \min(Y(\mathbf{X})) | Y(\mathbf{X}) = \mathbf{Y}) \tag{6.8}$$

$$= \mathbb{E}\left[\mathbf{1}_{Y(\mathbf{x}) \leq \min(Y(\mathbf{X}))} | Y(\mathbf{X}) = \mathbf{Y}\right] = \Phi\left(\frac{\min(\mathbf{Y}) - m_{OK}(\mathbf{x})}{s_{OK}(\mathbf{x})}\right), \tag{6.9}$$

where Φ is the gaussian cumulative distribution function, and the last equality follows eq. 6.3. The threshold $\min(\mathbf{Y})$ is sometimes replaced by some arbitrary target $T \in \mathbb{R}$, as evokated in [38]. PI is known to provide a very local search whenever the value of T is equal or close to $min(\mathbf{Y})$. Taking several T's is a remedy proposed by [20] to force global exploration. Of course, this new degree of freedom is also one more parameter to fit. In other respects, PI has also been succesfully used as pre-selection criterion in GP-assisted evolution strategies [49], where it was pointed out that PI is performant but has a tendency to sample in unexplored areas. We argue that the chosen covariance structure plays a capital role in such matters, depending whether the Kriging mean is overshooting the observations or not. The next presented criterion, the *expected improvement*, is less sensitive to such issues since it explicitly integrates both Kriging mean and variance.

6.2.2.5 Maximizing the Expected Improvement

An alternative solution is to maximize the *expected improvement* (EI),

$$EI(\mathbf{x}) := \mathbb{E}[(\min(Y(\mathbf{X}) - Y(\mathbf{x}))^+ | Y(\mathbf{X}) = \mathbf{Y}] = \mathbb{E}[\max\{0, \min(Y(\mathbf{X})) - Y(\mathbf{x})\} | Y(\mathbf{X}) = \mathbf{Y}], \tag{6.10}$$

that additionally takes into account the magnitude of the improvements. EI measures how much improvement is expected by sampling at $\mathbf{x}$. *In fine*, the improvement will be 0 if $y(\mathbf{x})$ is above $min(\mathbf{Y})$ and $min(\mathbf{Y}) - y(\mathbf{x})$ else. Knowing the conditional distribution of $Y(\mathbf{x})$, it is possible to calculate EI in closed form:

$$EI(\mathbf{x}) = (min(\mathbf{Y}) - m_{OK}(\mathbf{x}))\Phi\left(\frac{min(\mathbf{Y}) - m_{OK}(\mathbf{x})}{s_{OK}(\mathbf{x})}\right) + s_{OK}(\mathbf{x})\phi\left(\frac{min(\mathbf{Y}) - m_{OK}(\mathbf{x})}{s_{OK}(\mathbf{x})}\right), \tag{6.11}$$

where ϕ stands for the probability density function of the standard normal law $\mathcal{N}(0,1)$.

Proof of 6.11: $EI(\mathbf{x}) = \mathbb{E}[(min(\mathbf{Y}) - Y(\mathbf{x}))\mathbb{1}_{Y(\mathbf{x}) \leq min(\mathbf{Y})} | Y(\mathbf{X}) = \mathbf{Y}]$

$$= \int_{-\infty}^{min(\mathbf{Y})} (min(\mathbf{Y}) - t) f_{\mathcal{N}(m_{KO}(\mathbf{x}), s^2_{KO}(\mathbf{x}))}(t)dt = \int_{-\infty}^{\frac{min(\mathbf{Y}) - m_{KO}(\mathbf{x})}{s_{KO}(\mathbf{x})}} (min(\mathbf{Y}) - m_{KO}(\mathbf{x}) - s_{KO}(\mathbf{x}) \times u) f_{\mathcal{N}(0,1)}(u)du$$

$$= (min(\mathbf{Y}) - m_{KO}(\mathbf{x})) \int_{-\infty}^{\frac{min(\mathbf{Y}) - m_{KO}(\mathbf{x})}{s_{KO}(\mathbf{x})}} f_{\mathcal{N}(0,1)}(u)du - s_{KO}(\mathbf{x}) \int_{-\infty}^{\frac{min(\mathbf{Y}) - m_{KO}(\mathbf{x})}{s_{KO}(\mathbf{x})}} u \times f_{\mathcal{N}(0,1)}(u)du$$

$$= (min(\mathbf{Y}) - m_{KO}(\mathbf{x}))\,\Phi\left(\frac{min(\mathbf{Y}) - m_{KO}(\mathbf{x})}{s_{KO}(\mathbf{x})}\right) + s_{KO}(\mathbf{x})\phi\left(\frac{min(\mathbf{Y}) - m_{KO}(\mathbf{x})}{s_{KO}(\mathbf{x})}\right)$$

EI represents a trade-off between promising and uncertain zones. This criterion has important properties for sequential exploration: it is null at the already visited sites, and positive everywhere else with a magnitude that is increasing with the Kriging variance and with the decreasing Kriging mean (EI maximizers are indeed part of the Pareto front of $(s_{OK}, -m_{OK})$). Such features are usually demanded from global optimization procedures (see [21] for instance). EI and the probability of improvement are compared in fig. (2).

6.2.2.6 The *Stepwise Uncertainty Reduction* (SUR) Strategy

SUR has been introduced in [11] and extended to global optimization in [50, 51]. By modeling y using the process Y's conditional law $Y(\mathbf{x})|\mathbf{Y}$, it is possible to define $\mathbf{x}^*|\mathbf{Y}$, the conditional law of Y's global minimizer $\mathbf{x}^*$, and its density $p_{\mathbf{x}^*|\mathbf{Y}}(\mathbf{x})$. The uncertainty about the location of $\mathbf{x}^*$ is measured as the entropy of $p_{\mathbf{x}^*|\mathbf{Y}}(\mathbf{x})$, $H(\mathbf{x}^*|\mathbf{Y})$. $H(\mathbf{x}^*|\mathbf{Y})$ diminishes as the distribution of $\mathbf{x}^*|\mathbf{Y}$ gets more peaked. Conceptually, the SUR strategy for global optimization chooses as next iterate the point that specifies the most the location of the optimum,

$$\mathbf{x}' = argmin_{\mathbf{x} \in D} H(\mathbf{x}^*|\mathbf{Y}, Y(\mathbf{x})) \tag{6.12}$$

In practice, $p_{\mathbf{x}^*|\mathbf{Y}}(\mathbf{x})$ is estimated by Monte-Carlo sampling of $Y(\mathbf{x})|\mathbf{Y}$ at a finite number of locations in D, which may become a problem in high dimensional D's as the number of locations must geometrically increase with d to properly fill the space. The SUR criterion is different in nature from the criteria presented so far in that it does not maximize an immediate (i.e. at the next iteration) payoff but rather lays the foundation of a delayed payoff by gaining a more global knowledge on Y (reduce the entropy of its optima). The multi-points EI criterion we are focusing on in the present chapter also uses a delayed payoff measure.

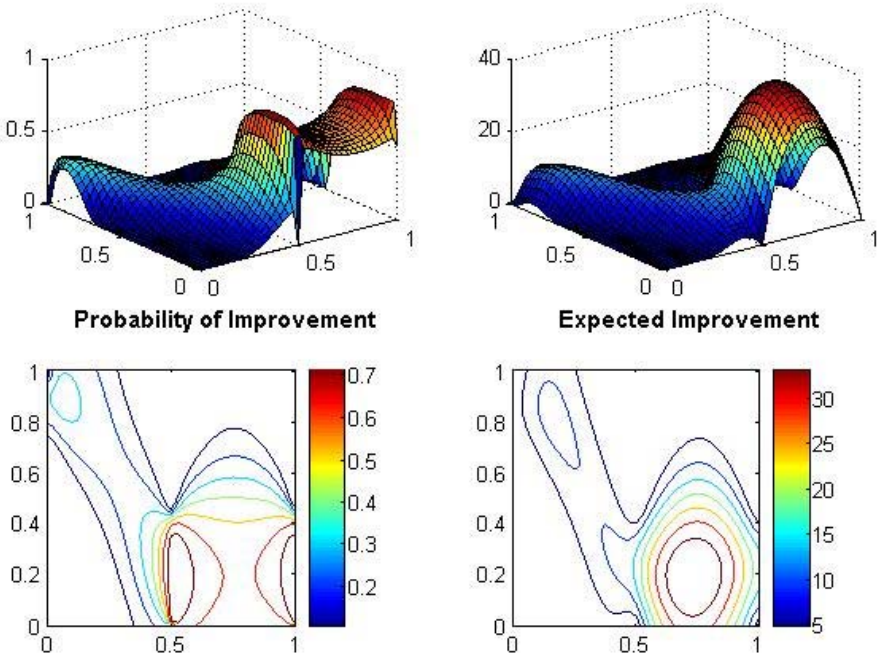


Fig. 6.2 PI and EI surfaces of the Branin-Hoo function (same design of experiments, Kriging model, and covariance parameters as in fig. 6.2.1). Maximizing PI leads to sample near the good points (associated with low observations) whereas maximizing EI leads here to sample between the good points. By construction, both criteria are null at the design of experiments, but the probability of improvement is very close to $\frac{1}{2}$ in a neighborhood of the point(s) where the function takes its current minimum

6.2.2.7 The *Efficient Global Optimization* (EGO) Algorithm

EGO [22] relies on the EI criterion. Starting with an initial Design $\mathbf{X}$ (typically a Latin Hypercube), EGO sequentially visits the current global maximizer of EI (say the first visited one if there is more than one global maximizer) and updates the OK metamodel at each iteration, including hyperparameters re-estimation:

1. Evaluate y at $\mathbf{X}$, set $\mathbf{Y} = y(\mathbf{X})$ and estimate covariance parameters of Y by MLE (Maximum Likelihood Estimation)
2. While stopping criterion not met
   a. Compute $\mathbf{x}' = argmax_{\mathbf{x} \in D} EI(\mathbf{x})$, set $\mathbf{X} = \mathbf{X} \cup \{\mathbf{x}'\}$ and $\mathbf{Y} = \mathbf{Y} \cup \{y(\mathbf{x}')\}$
   b. Re-estimate covariance parameters by MLE

After having been developed in [22, 47], EGO has inspired contemporary works in optimization of expensive-to-evaluate functions. For instance, [19] exposes some EGO-based methods for the optimization of noisy black-box functions like stochastic simulators. [18] focuses on multiple numerical simulators with different levels of fidelity, and introduces the so-called *augmented EI* criterion, integrating possible heterogeneity in the simulation times. Moreover, [26] proposes an adaptation to multi-objective optimization, [17] proposes an original multi-objective adaptation of EGO for physical experiments, and [28] focuses on robust criteria for multiobjective constrained optimization with applications to laminating processes.

In all, one major drawback of the EGO-like algorithms discussed so far is that they do not allow parallel evaluations of y, which is desirable for costly simulators (e.g. a crash-test simulation run typically lasts 24 hours). This was already pointed out in [47], where the multi-points EI was defined but not further developed. Here we continue this work by expliciting the latter multi-points EI (q-EI), and by proposing two classes of heuristics strategies meant to approximatly optimize the q-EI, and hence (almost) simultaneously deliver an arbitrary number of points without intermediate evaluations of y. In particular, we analytically derive the 2-EI, and explain in detail how to take advantage of statistical interpretations of Kriging to consistently compute q-EI by simulation when $q > 2$, which happens to provide quite a general template for desiging Kriging-based parallel evaluation strategies dedicated to optimization or other purposes.

6.3 The Multi-points Expected Improvement (q-EI) Criterion

The main objective of the present work is to analyze and then approximately optimize a global optimization criterion, the q-EI, that yields q points. Since q-EI is an extension of EI, all derivations are performed within the framework of OK. Such criterion is the first step towards a parallelized version of the EGO algorithm [22]. It also departs, like the SUR criterion, from other criteria that look for an immediate payoff. We now propose a progressive construction of the q-EI, by coming back to the random variable *improvement*.

Both criteria of PI and EI that we have previously recalled share indeed the feature of being conditional expectations of quantities involving the *improvement*. The *improvement* brought by sampling at some $\mathbf{x} \in D$ is indeed defined by $I(\mathbf{x}) := (min(Y(\mathbf{X})) - Y(\mathbf{x}))^+$, and is positive whenever the value sampled at $\mathbf{x}$, $Y(\mathbf{x})$, is below the current minimum $min(Y(\mathbf{X}))$. Now, if we sample Y at q new locations $\mathbf{x}^{n+1}, \ldots, \mathbf{x}^{n+q} \in D$ simultaneously, it seems quite natural to define the joint —or *multipoints*— improvement as follows:

$$\begin{aligned} \forall \mathbf{x}^{n+1} \quad \mathbf{x}^{n+q} \in D \quad I(\mathbf{x}^{n+1} \quad \mathbf{x}^{n+q}) &:= \max\left(I(\mathbf{x}^{n+1}) \quad I(\mathbf{x}^{n+q})\right) \\ &= \max\left((min(Y(\mathbf{X})) - Y(\mathbf{x}^{n+1}))^+ \quad (min(Y(\mathbf{X})) - Y(\mathbf{x}^{n+q}))^+\right) \\ &= \left(\min(Y(\mathbf{X})) - \min(Y(\mathbf{x}^{n+1}) \quad Y(\mathbf{x}^{n+q}))\right)^+ \end{aligned} \tag{6.13}$$

where we used the fact that $\forall a,b,c \in \mathbb{R},\ \max((a-b)^+,(a-c)^+) = (a-b)^+$ if $b \leq c$ and $(a-c)^+$ else. The way of unifying the q criteria of (1-point) improvements used in eq. 6.13 deserves to be called *elitist*: one juges the quality of the set of q-points as a function only of the one that performs the best. This is to be compared for instance to the weighted sums of criteria encountered in many political science applications.

The q-points EI criterion (as already defined but not developed in [47] under the name "q-step EI") is then straightforwardly defined as conditional expectation of the improvement brought by the q considered points:

$$\begin{aligned} EI(\mathbf{x}^{n+1} \quad \mathbf{x}^{n+q}) &:= \mathbb{E}[\max\{(min(Y(\mathbf{X})) - Y(\mathbf{x}^{n+1}))^+ \quad (min(\mathbf{Y}) - Y(\mathbf{x}^{n+q}))^+\}|Y(\mathbf{X}) = \mathbf{Y}] \\ &= \mathbb{E}\left[(\min(Y(\mathbf{X})) - \min\left(Y(\mathbf{x}^{n+1}) \quad Y(\mathbf{x}^{n+q})\right))^+|Y(\mathbf{X}) = \mathbf{Y}\right] \\ &= \mathbb{E}\left[(\min(\mathbf{Y}) - \min\left(Y(\mathbf{x}^{n+1}) \quad Y(\mathbf{x}^{n+q})\right))^+|Y(\mathbf{X}) = \mathbf{Y}\right] \end{aligned} \tag{6.14}$$

Hence, the q-EI may be seen as the regular EI applied to the random variable $\min(Y(\mathbf{x}^{n+1}),...,Y(\mathbf{x}^{n+q}))$. We thus have to deal with a minimum of dependent random variables. Fortunately, eq. 6.4 provides us with the exact joint distribution of the q unknown responses conditional on the observations:

$$[(Y(\mathbf{x}^{n+1}),...,Y(\mathbf{x}^{n+q}))|Y(\mathbf{X}) = \mathbf{Y}] \sim \mathcal{N}((m_{OK}(\mathbf{x}^{n+1}),...,m_{OK}(\mathbf{x}^{n+q})), S_q) \tag{6.15}$$

where the elements of the conditional covariance matrix S_q are $(S_q)_{i,j} = c_{OK}(\mathbf{x}^{n+i}, \mathbf{x}^{n+j})$ (See eq. 6.5). We now propose two different ways to evaluate the criterion eq. 6.14, depending whether $q = 2$ or $q \geq 3$.

6.3.1 Analytical Calculation of 2-EI

We first focus on the calculation of the 2-EI associated with two arbitrary points $\mathbf{x}^{n+1}, \mathbf{x}^{n+2} \in D$, defined as

$$EI(\mathbf{x}^{n+1}, \mathbf{x}^{n+2}) := \mathbb{E}[(\min(Y(\mathbf{X})) - min(Y(\mathbf{x}^{n+1}), Y(\mathbf{x}^{n+2})))^+|Y(\mathbf{X}) = \mathbf{Y}],$$

Let us remark that in reformulating the positive part function, the expression above can also be written:

$$EI(\mathbf{x}^{n+1}, \mathbf{x}^{n+2}) = \mathbb{E}[(\min(Y(\mathbf{X})) - min(Y(\mathbf{x}^{n+1}), Y(\mathbf{x}^{n+2})))\mathbf{1}_{min(Y(\mathbf{x}^{n+1}),Y(\mathbf{x}^{n+2})) \leq \min(\mathbf{Y})}|Y(\mathbf{X}) = \mathbf{Y}].$$

We will now show that the 2-EI can be developed as a sum of two 1-EI's, plus a correction term involving 1- and 2-dimensional gaussian cumulative distributions.

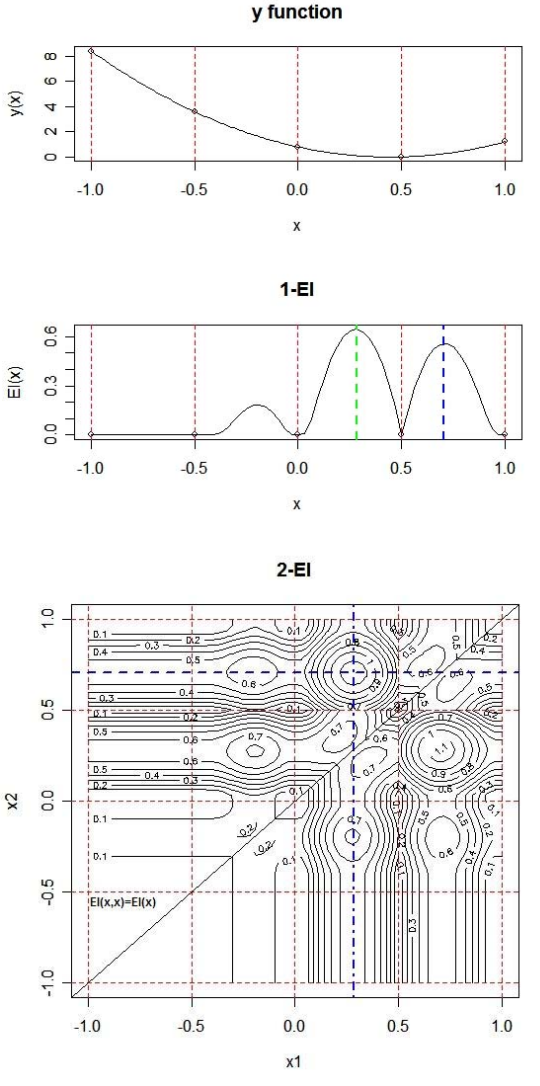


Fig. 6.3 1-EI (lower left) and 2-EI (right) functions associated with a monodimensional quadratic function $(y(x) = 4 \times (x - 0.45)^2$ known at $\mathbf{X} = \{-1, -0.5, 0, 0.5, 1\}$. The OK metamodel has here a cubic covariance with parameters $\sigma^2 = 10,$ scale $= 0.9)$

Before all, some classical results of conditional calculus allow us to precise the dependence between $Y(\mathbf{x}^{n+1})$ and $Y(\mathbf{x}^{n+2})$ conditional on $Y(\mathbf{X}) = \mathbf{Y}$, and to fix some additional notations. $\forall i, j \in \{1,2\}$ $(i \neq j)$, we note:

$$\begin{cases} m_i := m_{KO}(\mathbf{x}^i) = \mathbb{E}[Y(\mathbf{x}^{n+i})|Y(\mathbf{X}) = \mathbf{Y}], \\ \sigma_i := s_{KO}(\mathbf{x}^{n+i}) = \sqrt{Var[Y(\mathbf{x}^{n+i})|Y(\mathbf{X}) = \mathbf{Y}]}, \\ c_{1,2} := \rho_{1,2}\sigma_1\sigma_2 := cov[Y(\mathbf{x}^{n+1}), Y(\mathbf{x}^{n+2})|Y(\mathbf{X}) = \mathbf{Y}], \\ m_{i|j} = \mathbb{E}[Y(\mathbf{x}^{n+i})|Y(\mathbf{X}) = \mathbf{Y}, Y(\mathbf{x}^{n+j}))] = m_i + c_{1,2}\sigma_i^{-2}(Y(\mathbf{x}^{n+j}) - m_j), \\ \sigma_{i|j}^2 = \sigma_i^2 - c_{1,2}^2\sigma_j^{-2} = \sigma_i^2(1-\rho_{12}^2). \end{cases} \tag{6.16}$$

At this stage we are in position to compute $EI(\mathbf{x}^{n+1}, \mathbf{x}^{n+2})$ in four steps. From now on, we replace the complete notation $Y(\mathbf{x}^{n+i})$ by Y_i and forget the conditioning on $Y(\mathbf{X}) = \mathbf{Y}$ for the sake of clarity.

Step 1

$$\begin{aligned} EI(\mathbf{x}^{n+1}, \mathbf{x}^{n+2}) &= \mathbb{E}[(min(\mathbf{Y}) - min(Y_1, Y_2))\mathbf{1}_{min(Y_1,Y_2)\leq min(\mathbf{Y})}] \\ &= \mathbb{E}[(min(\mathbf{Y}) - min(Y_1, Y_2))\mathbf{1}_{min(Y_1,Y_2)\leq min(\mathbf{Y})}(\mathbf{1}_{Y_1\leq Y_2} + \mathbf{1}_{Y_2\leq Y_1})] \\ &= \mathbb{E}[(min(\mathbf{Y}) - Y_1)\mathbf{1}_{Y_1\leq min(\mathbf{Y})}\mathbf{1}_{Y_1\leq Y_2}] + E[(min(\mathbf{Y}) - Y_2)\mathbf{1}_{Y_2\leq min(\mathbf{Y})}\mathbf{1}_{Y_2\leq Y_1}] \end{aligned}$$

Since both terms of the last sum are similar (up to a permutation between $\mathbf{x}^{n+1}$ and $\mathbf{x}^{n+2}$), we will restrict our attention to the first one. Using $\mathbf{1}_{Y_1\leq Y_2} = 1 - \mathbf{1}_{Y_2\leq Y_1}$[5], we get:

$$\begin{aligned} \mathbb{E}[(min(\mathbf{Y}) - Y_1)\mathbf{1}_{Y_1\leq min(\mathbf{Y})}\mathbf{1}_{Y_1\leq Y_2}] &= \mathbb{E}[(min(\mathbf{Y}) - Y_1)\mathbf{1}_{Y_1\leq min(\mathbf{Y})}(1 - \mathbf{1}_{Y_2\leq Y_1})] \\ &= EI(\mathbf{x}^{n+1}) - \mathbb{E}[(min(\mathbf{Y}) - Y_1)\mathbf{1}_{Y_1\leq min(\mathbf{Y})}\mathbf{1}_{Y_2\leq Y_1}] \\ &= EI(\mathbf{x}^{n+1}) + B(\mathbf{x}^{n+1}, \mathbf{x}^{n+2}) \end{aligned}$$

where $B(\mathbf{x}^{n+1}, \mathbf{x}^{n+2}) = E[(Y_1 - min(\mathbf{Y}))\mathbf{1}_{Y_1\leq min(\mathbf{Y})}\mathbf{1}_{Y_2\leq Y_1}]$. Informally, $B(\mathbf{x}^{n+1}, \mathbf{x}^{n+2})$ is the opposite of the improvement brought by Y_1 when $Y_2 \leq Y_1$ and hence that doesn't contribute to the 2-points improvement. Our aim in the next steps will be to give an explicit expression for $B(\mathbf{x}^{n+1}, \mathbf{x}^{n+2})$.

Step 2

$$B(\mathbf{x}^{n+1}, \mathbf{x}^{n+2}) = \mathbb{E}[Y_1\mathbf{1}_{Y_1\leq min(\mathbf{Y})}\mathbf{1}_{Y_2\leq Y_1}] - min(\mathbf{Y})\mathbb{E}[\mathbf{1}_{Y_1\leq min(\mathbf{Y})}\mathbf{1}_{Y_2\leq Y_1}]$$

At this point, it is worth noticing that $Y_1 \overset{\mathscr{L}}{=} m_1 + \sigma_1 N_1$ (always conditional on $Y(\mathbf{X}) = \mathbf{Y}$) with $N_1 \sim \mathcal{N}(0,1)$. Substituing this decomposition in the last expression of $B(\mathbf{x}^{n+1}, \mathbf{x}^{n+2})$ delivers:

$$B(\mathbf{x}^{n+1}, \mathbf{x}^{n+2}) = \sigma_1\mathbb{E}[N_1\mathbf{1}_{Y_1\leq min(\mathbf{Y})}\mathbf{1}_{Y_2\leq Y_1}] + (m_1 - min(\mathbf{Y}))\mathbb{E}[\mathbf{1}_{Y_1\leq min(\mathbf{Y})}\mathbf{1}_{Y_2\leq Y_1}]$$

[5] This expression should be noted $1 - \mathbf{1}_{Y_2<Y_1}$, but since we work with continous random variables, it sufficies that their correlation is $\neq 1$ for the expression to be exact ($\{Y_1 = Y_2\}$ is then neglectable). We implicitely do this assumption in the following.

The two terms of this sum require some attention. We compute them in detail in the two next steps.

Step 3. Using a key property of conditional calculus[6], we obtain

$$\mathbb{E}[N_1 \mathbf{1}_{Y_1 \leq min(\mathbf{Y})} \mathbf{1}_{Y_2 \leq Y_1}] = \mathbb{E}[N_1 \mathbf{1}_{Y_1 \leq min(\mathbf{Y})} \mathbb{E}[\mathbf{1}_{Y_2 \leq Y_1} | Y_1]],$$

and the fact that $Y_2|Y_1 \sim \mathcal{N}(m_{2|1}(Y_1), s^2_{2|1}(Y_1))$ (all conditional on the observations) leads to the following:

$$\mathbb{E}[\mathbf{1}_{Y_2 \leq Y_1} | Y_1] = \Phi\left(\frac{Y_1 - m_{2|1}}{s_{2|1}}\right) = \Phi\left(\frac{Y_1 - m_2 - \frac{c_{1,2}}{\sigma_1^2}(Y_1 - m_1)}{\sigma_2\sqrt{1-\rho_{12}^2}}\right)$$

Back to the main term and using again the normal decomposition of Y_1, we get:

$$\mathbb{E}\left[N_1 \mathbf{1}_{Y_1 \leq min(\mathbf{Y})} 1_{Y_2 \leq Y_1}\right] = \left[N_1 1_{N_1 \leq \frac{min(\mathbf{Y}) - m_1}{\sigma_1}} \Phi\left(\frac{m_1 - m_2 + (\sigma_1 - \rho_{12}\sigma_2)N_1}{\sigma_2\sqrt{1-\rho_{12}^2}}\right)\right] = \mathbb{E}\left[N_1 \mathbf{1}_{N_1 \leq \gamma_1} \Phi(\alpha_1 N_1 + \beta_1)\right]$$

$$\text{where } \gamma_1 = \frac{min(\mathbf{Y}) - m_1}{\sigma_1},\ \beta_1 = \frac{m_1 - m_2}{\sigma_2\sqrt{1-\rho_{12}^2}} \text{ and } \alpha_1 = \frac{\sigma_1 - \rho_{12}\sigma_2}{\sigma_2\sqrt{1-\rho_{12}^2}} \tag{6.17}$$

$\mathbb{E}[N_1 \mathbf{1}_{N_1 \leq \gamma_1} \Phi(\alpha_1 N_1 + \beta_1)]$ can be computed applying an integration by parts:

$$\int_{-\infty}^{\gamma_1} u\phi(u)\Phi(\alpha_1 u + \beta_1)du = -\phi(\gamma_1)\Phi(\alpha_1\gamma_1 + \beta_1) + \frac{\alpha_1}{2\pi}\int_{-\infty}^{\gamma_1} e^{\frac{-u^2-(\alpha_1 u+\beta_1)^2}{2}}du$$

And since $u^2 + (\alpha_1 u + \beta_1)^2 = \left(\sqrt{(1+\alpha_1^2)}u + \frac{\alpha_1\beta_1}{\sqrt{1+\alpha_1^2}}\right)^2 + \frac{\beta_1^2}{1+\alpha_1^2}$, the last integral reduces to:

$$\sqrt{2\pi}\phi\left(\sqrt{\frac{\beta_1^2}{1+\alpha_1^2}}\right)\int_{-\infty}^{\gamma_1} e^{\frac{-\left(\sqrt{(1+\alpha_1^2)}u + \frac{\alpha_1\beta_1}{\sqrt{1+\alpha_1^2}}\right)^2}{2}}du = \frac{2\pi\phi\left(\sqrt{\frac{\beta_1^2}{1+\alpha_1^2}}\right)}{\sqrt{(1+\alpha_1^2)}}\int_{-\infty}^{\sqrt{(1+\alpha_1^2)}\gamma_1 + \frac{\alpha_1\beta_1}{\sqrt{1+\alpha_1^2}}} \frac{e^{\frac{-v^2}{2}}}{\sqrt{2\pi}}dv$$

We conclude in using the definition of the cumulative distribution function:

$$\mathbb{E}[N_1 \mathbf{1}_{Y_1 \leq min(\mathbf{Y})} 1_{Y_2 \leq Y_1}] = -\phi(\gamma_1)\Phi(\alpha_1\gamma_1 + \beta_1) + \frac{\alpha_1\phi\left(\sqrt{\frac{\beta_1^2}{1+\alpha_1^2}}\right)}{\sqrt{(1+\alpha_1^2)}}\Phi\left(\sqrt{(1+\alpha_1^2)}\gamma_1 + \frac{\alpha_1\beta_1}{\sqrt{1+\alpha_1^2}}\right)$$

[6] For all function ϕ in $\mathcal{L}^2(\mathbb{R},\mathbb{R})$, $E[X\phi(Y)] = E[E[X|Y]\phi(Y)]$.

Step 4. We then compute the term $\mathbb{E}[\mathbf{1}_{Y_1 \leq min(\mathbf{Y})}\mathbf{1}_{Y_2 \leq Y_1}] = E[\mathbf{1}_{X \leq min(\mathbf{Y})}\mathbf{1}_{Z \leq 0}]$, where $(X,Z) := (Y_1, Y_2 - Y_1)$ follows a 2-dimensional gaussian distribution with expectation $M = (m_1, m_2 - m_1)$, and covariance matrix $\Gamma := \begin{pmatrix} \sigma_1^2 & c_{1,2} - \sigma_1^2 \\ c_{1,2} - \sigma_1^2 & \sigma_2^2 + \sigma_1^2 - 2c_{1,2} \end{pmatrix}$. The final results rely on the fact that: $\mathbb{E}[\mathbf{1}_{X \leq min(\mathbf{Y})}\mathbf{1}_{Z \leq 0}] = CDF(M,\Gamma)(min(\mathbf{Y}),0)$, where CDF stands for the bi-gaussian cumulative distribution function:

$$EI(\mathbf{x}^1,\mathbf{x}^2) = EI(\mathbf{x}^1) + EI(\mathbf{x}^2) + B(\mathbf{x}^1,\mathbf{x}^2) + B(\mathbf{x}^2,\mathbf{x}^1), \qquad (6.18)$$

$$\text{where } \begin{cases} B(\mathbf{x}^1,\mathbf{x}^2) = (m_{OK}(\mathbf{x}^1) - min(\mathbf{Y}))\delta(\mathbf{x}^1,\mathbf{x}^2) + \sigma_{OK}(\mathbf{x}^1)\varepsilon(\mathbf{x}^1,\mathbf{x}^2) \\ \varepsilon(\mathbf{x}^1,\mathbf{x}^2) = \alpha_1\phi\left(\frac{|\beta_1|}{\sqrt{(1+\alpha_1^2)}}\right)\Phi\left((1+\alpha_1^2)^{\frac{1}{2}}\left(\gamma + \frac{\alpha_1\beta_1}{1+\alpha_1^2}\right)\right) - \phi(\gamma)\Phi(\alpha_1\gamma+\beta_1) \\ \delta(\mathbf{x}^1,\mathbf{x}^2) = CDF(\Gamma)\begin{pmatrix} min(\mathbf{Y}) - m_1 \\ m_1 - m_2 \end{pmatrix} \end{cases}$$

Figure 6.3.1 represents the 1-EI and the 2-EI contour plots associated with a deterministic polynomial function known at 5 points. 1-EI advises here to sample between the "good points" of $\mathbf{X}$. The 2-EI contour illustrates some general properties: 2-EI is symmetric and its diagonal equals 1-EI, what can be easily seen by coming back to the definitions. Roughly said, 2-EI is high whenever the 2 points have high 1-EI and are reasonably distant from another (precisely, in the sense of the metric used in OK). Additionally, maximizing 2-EI selects here the two best local optima of 1-EI ($x_1 = 0.3$ and $x_2 = 0.7$). This is not a general fact. The next example illustrates for instance how 2-EI maximization can yield two points located around (but different from) 1-EI's global optimum whenever 1-EI has one single peak of great magnitude (see fig. 6.4).

6.3.2 q-EI Computation by Monte Carlo Simulations

Extrapolating the calculation of 2-EI to the general case gives complex expressions depending on q-dimensional gaussian cumulative distribution functions. Hence, it seems that the computation of q-EI when q is large would have to rely on numerical multivariate integral approximation techniques anyway. Therefore, directly evaluating q-EI by Monte-Carlo Simulation makes sense. Thanks to eq. 6.15, the random vector $(Y(\mathbf{x}^{n+1}),...,Y(\mathbf{x}^{n+q}))$ can be simulated conditional on $Y(\mathbf{X}) = \mathbf{Y}$ using a decomposition (e.g. Mahalanobis) of the covariance matrix S_q:

$$\forall k \in [1, n_{sim}],\ M_k = (m_{OK}(\mathbf{x}^{n+1}),...,m_{OK}(\mathbf{x}^{n+q})) + [S_q^{\frac{1}{2}}N_k]^T, N_k \sim \mathcal{N}(\mathbf{0}_q, \mathbf{I}_q)\ i.i.d. \qquad (6.19)$$

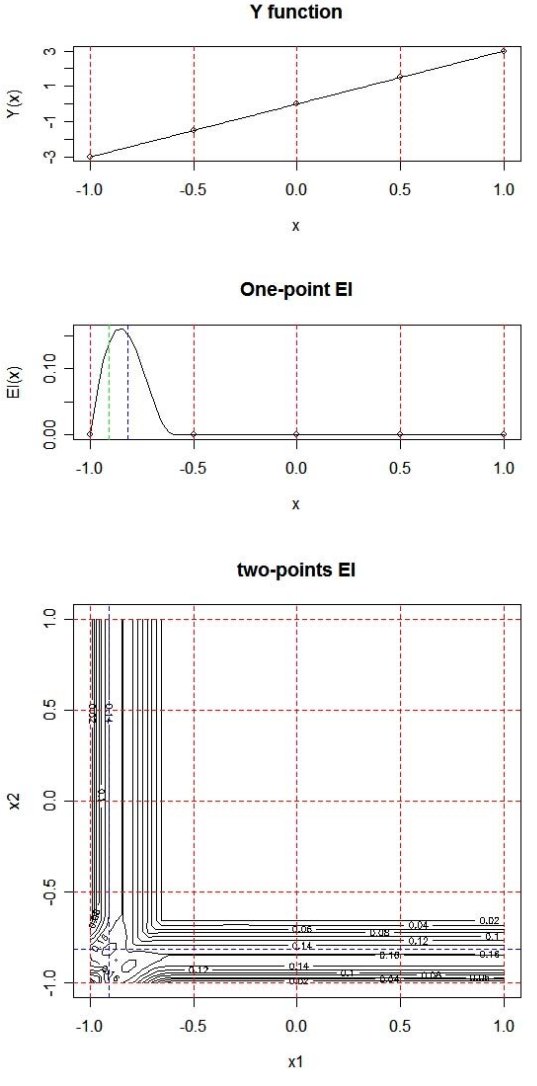


Fig. 6.4 1-point EI (lower left) and 2-points EI (right) functions associated with a monodimensional linear function ($y(x) = 3 \times x$) known at $\mathbf{X} = \{-1, -0.5, 0, 0.5, 1\}$. The OK metamodel has here a cubic covariance with parameters $\sigma^2 = 10,\ \text{scale} = 1.4$)

Computing the conditional expectation of any function (not necessarily linearly) of the conditioned random vector $(Y(\mathbf{x}^{n+1}), ..., Y(\mathbf{x}^{n+q}))$ knowing $Y(\mathbf{X}) = \mathbf{Y}$ can then be done in averaging the images of the simulated vectors by the considered function:

Function q-$\mathbb{E}$I($\mathbf{X}$, $\mathbf{Y}$, $\mathbf{X}^{new}$)
$L = \text{chol}(Var[Y(\mathbf{X}^{new})|Y(\mathbf{X}) = \mathbf{Y}])$ {Decomposition of S_q}
for $i \leftarrow 1, n_{sim}$ **do**
  $N \sim \mathcal{N}(0, I_q)$ {Drawing a standard gaussian vector N at random}
  $M_i = m_{OK}(\mathbf{X}^{new}) + LN$ {Simulating Y($\mathbf{X}^{new}$) conditional on $Y(\mathbf{X}) = \mathbf{Y}$}
  $qI_{sim}(i)$=[min($\mathbf{Y}$) − min(M_i)]$^+$ {Simulating the improvement at $\mathbf{X}^{new}$}
end for
$qEI_{sim} = \frac{1}{n_{sim}} \sum_{i=1}^{n_{sim}} qI_{sim}(i)$ {Estimation of q-EI}

A straightforward application of the Law of Large Numbers (LLN) yields indeed

$$qEI_{sim} = \sum_{i=1}^{n_{sim}} \frac{[\min(\mathbf{Y}) - \min(M_i)]^+}{n_{sim}} \xrightarrow[n_{sim} \to +\infty]{} EI(\mathbf{x}^1, \ldots, \mathbf{x}^q) \text{ a.s.}, \tag{6.20}$$

and the Central Limit Theorem (CLT) can finally be used to control the precision of the Monte Carlo approximation as a function of n_{sim} (see [40] for details concerning the variance estimation):

$$\sqrt{n_{sim}} \left(\frac{qEI_{sim} - EI(\mathbf{x}^1, \ldots, \mathbf{x}^q)}{\sqrt{Var[I(\mathbf{x}^1, \ldots, \mathbf{x}^q)]}} \right) \xrightarrow[n_{sim} \to +\infty]{} \mathcal{N}(0,1) \text{ in law.} \tag{6.21}$$

6.4 Approximated q-EI Maximization

The multi-points criterion that we have presented in the last section can potentially be used to deliver an additional design of experiments in one step through the resolution of the optimization problem

$$(\mathbf{x}'^{n+1}, \mathbf{x}'^{n+2}, \ldots, \mathbf{x}'^{n+q}) = argmax_{\mathbf{X}' \in D^q} [EI(\mathbf{X}')] \tag{6.22}$$

However, the computation of q-EI becomes intensive as q increases. Moreover, the optimization problem (6.22) is of dimension $d \times q$, and with a noisy and derivative-free objective function in the case where the criterion is estimated by Monte-Carlo. Here we try to find pseudo-sequential greedy strategies that approach the result of problem 6.22 while avoiding its numerical cost, hence circumventing the curse of dimensionality.

6.4.1 A First Greedy Strategy to Build a q-Points Design with the 1-Point EI

Instead of searching for the globally optimal vector $(\mathbf{x}'^{n+1}, \mathbf{x}'^{n+2}, ..., \mathbf{x}'^{n+q})$, an intuitive way of replacing it by a sequential approach is the following: first look for the next best single point $\mathbf{x}^{n+1} = argmax_{\mathbf{x}\in D} EI(\mathbf{x})$, then feed the model and look for $\mathbf{x}^{n+2} = argmax_{\mathbf{x}\in D} EI(x)$, and so on. Of course, the value $y(\mathbf{x}^{n+1})$ is not known at the second step (else we would be in a real sequential algorithm, like EGO). Nevertheless, we dispose of two pieces of information: the site $\mathbf{x}^{n+1}$ is assumed to have already been visited at the previous iteration, and $[Y(\mathbf{x}^{n+1})|\mathbf{Y} = Y(\mathbf{X})]$ has a known distribution. More precisely, the latter is $[Y(\mathbf{x}^{n+1})|Y(\mathbf{X}) = \mathbf{Y}] \sim \mathcal{N}(m_{OK}(\mathbf{x}^{n+1}), s^2_{OK}(\mathbf{x}^{n+1}))$. Hence, the second site $\mathbf{x}^{n+2}$ can be computed as:

$$\mathbf{x}^{n+2} = argmax_{\mathbf{x}\in D} \mathbb{E}\left[\mathbb{E}\left[(Y(\mathbf{x}) - \min(Y(\mathbf{X})))^+ | Y(\mathbf{X}) = \mathbf{Y}, Y(\mathbf{x}^{n+1})\right]\right], \quad (6.23)$$

and the same procedure can be applied iteratively to deliver q points, computing $\forall j \in [1, q-1]$:

$$\mathbf{x}^{n+j+1} = argmax_{\mathbf{x}\in D} \int_{\mathbf{u}\in\mathbb{R}^j} \left[\mathbb{E}\left[(Y(\mathbf{x}) - \min(Y(\mathbf{X})))^+ | Y(\mathbf{X}) = \mathbf{Y}\ Y(\mathbf{x}^{n+1}) \quad Y(\mathbf{x}^{n+j-1})\right]\right] f_{Y(\mathbf{X}^{1:j})|Y(\mathbf{X})=\mathbf{Y}}(\mathbf{u}) d\mathbf{u} \quad (6.24)$$

where $f_{Y(\mathbf{X}^{1:j})|Y(\mathbf{X})=\mathbf{Y}}$ is the multivariate gaussian density of the OK conditional distribution at $(\mathbf{x}^{n+1}, ..., \mathbf{x}^{n+j})$. Although eq. 6.24 is a sequentialized version of the q-points expected improvement maximization, it doesn't completely fulfill our objectives. There is still a multivariate gaussian density to integrate, which seems to be a typical curse in such problems dealing with dependent random vectors. We now present two classes of heuristic strategies meant to circumvent the computational complexity encountered in eq. 6.24.

6.4.2 The Kriging Believer (KB) and Constant Liar (CL) Strategies

Lying to escape intractable calculations: Starting from the principle of eq. 6.24, we propose to weaken the conditional knowledge taken into account at each iteration. This very elementary idea inspired two heuristic strategies that we expose and test in the next two subsections: the *Kriging Believer* and the *Constant Liar*.

6.4.2.1 Believing the OK Predictor: The KB Heuristic Strategy

The *Kriging Believer* strategy replaces the conditional knowledge about the responses at the sites chosen within the last iterations by deterministic values equal to the expectation of the Kriging predictor. Keeping the same notations as previously, the strategy can be summed up as follows:

Algorithm 1. The Kriging Believer algorithm: a first approximate solution of the multipoints problem $(\mathbf{x}^{'n+1}, \mathbf{x}^{'n+2}, ..., \mathbf{x}^{'n+q}) = argmax_{\mathbf{X}' \in D^q}[EI(\mathbf{X}')]$

Function KB($\mathbf{X}$, $\mathbf{Y}$, q)
for $i \leftarrow 1, q$ **do**
  $\mathbf{x}^{n+i} = argmax_{\mathbf{x} \in D} EI(\mathbf{x})$
  $m_{OK}(\mathbf{x}^{n+i}) = \mathbb{E}[Y(\mathbf{x}^{n+i}) | Y(\mathbf{X}) = \mathbf{Y}]$
  $\mathbf{X} = \mathbf{X} \bigcup \{\mathbf{x}^{n+i}\}$
  $\mathbf{Y} = \mathbf{Y} \bigcup \{m_{OK}(\mathbf{x}^{n+i})\}$
end for

This sequential strategy delivers a q-points design and is computationally affordable since it relies on the analytically known EI, optimized in d dimensions. However, there is a risk of failure, since believing an OK predictor that overshoots the observed data may lead to a sequence that gets trapped in a non-optimal region for many iterations (see 4.3). We now propose a second strategy that reduces this risk.

6.4.2.2 Updating the OK Metamodel with Fake Observations: The CL Heuristic Strategy

Let us now consider a sequential strategy in which the metamodel is updated (still without hyperparameter re-estimation) at each iteration with a value L exogenously fixed by the user, here called a "lie". The strategy referred to as the *Constant Liar* consists in lying with the same value L at every iteration: maximize EI (i.e. find $\mathbf{x}^{n+1}$), actualize the model as if $y(\mathbf{x}^{n+1}) = L$, and so on always with the same $L \in \mathbb{R}$:

Algorithm 2. The Constant Liar algorithm: another approximate solution of the multipoints problem $(\mathbf{x}^{'n+1}, \mathbf{x}^{'n+2}, ..., \mathbf{x}^{'n+q}) = argmax_{\mathbf{X}' \in D^q}[EI(\mathbf{X}')]$

Function CL($\mathbf{X}$, $\mathbf{Y}$, L, q)
for $i \leftarrow 1, q$ **do**
  $\mathbf{x}^{n+i} = argmax_{\mathbf{x} \in D} EI(\mathbf{x})$
  $\mathbf{X} = \mathbf{X} \bigcup \{\mathbf{x}^{n+i}\}$
  $\mathbf{Y} = \mathbf{Y} \bigcup \{L\}$
end for

The effect of L on the performance of the resulting optimizer is investigated in the next section. L should logically be determined on the basis of the values taken by y at $\mathbf{X}$. Three values, $min\{\mathbf{Y}\}$, $mean\{\mathbf{Y}\}$, and $max\{\mathbf{Y}\}$ are considered here. The larger L is, the more explorative the algorithm will be, and vice versa.

6.4.3 Empirical Comparisons with the Branin-Hoo Function

The four optimization strategies presented in the last section are now compared on the Branin-Hoo function which is a classical test-case in global optimization [22, 38, 47]:

$$\begin{cases} y_{BH}(x_1,x_2) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1-\frac{1}{8\pi})cos(x_1) + 10 \\ x_1 \in [-5,10],\ x_2 \in [0,15] \end{cases} \tag{6.25}$$

y_{BH} has three global minimizers $(-3.14,12.27)$, $(3.14,2.27)$, $(9.42,2.47)$, and the global minimum is approximately equal to 0.4. The variables are normalized by the transformation $x_1' = \frac{x_1+5}{15}$ and $x_2' = \frac{x_2}{15}$. The initial design of experiments is a 3×3 complete factorial design $\mathbf{X}_9$ (see 6.5), thus $\mathbf{Y} = y_{BH}(\mathbf{X}_9)$. Ordinary Kriging is applied with a stationary, anisotropic, gaussian covariance function

$$\forall h = (h_1,h_2) \in \mathbb{R}^2,\ c(h) = \sigma^2 e^{-\theta_1 h_1^2 - \theta_2 h_2^2} \tag{6.26}$$

where the parameters (θ_1,θ_2) are fixed to their Maximum Likelihood Estimate $(5.27,0.26)$, and σ^2 is estimated within Kriging, as an implicit function of (θ_1,θ_2) (like in [22]). We built a 10-points optimization design with each strategy, and additionally estimated by Monte Carlo simulations ($n_{sim} = 10^4$) the PI and EI values brought by the q first points of each strategy (here $q \in \{2,6,10\}$). The results are gathered in Tab. 6.4.3.

The four strategies (KB and the three variants of CL) gave clearly different designs and optimization performances. In the first case, *Constant Liar* (CL) sequences behaved as if the already visited points generated a repulsion, with a magnitude increasing with L. The tested values $L = \max(\mathbf{Y})$ and $L = mean(\mathbf{Y})$ forced the exploration designs to fill the space by avoiding $\mathbf{X_9}$. Both strategies provided space-filling, exploratory designs with high probabilities of improvement (10-*PI* near 100%) and promising q-*EI* values (see Table 1). *In fine*, they brought respective actual improvements of 7.86 and 6.25.

Of all the tested strategies, CL[min(**Y**)] gave here the best results. In 6 iterations, it visited the three locally optimal zones of y_{BH}. In 10 iterations, it gave the best actual improvement among the considered strategies, which is furthermore in agreement with the 10-points EI values simulated by Monte-Carlo. It seems in fact that the soft repulsion when $L = \min(\mathbf{Y})$ is the right tuning for the optimization of the Branin-Hoo function, with the initial design $\mathbf{X}_9$.

In the second case, the KB has yielded here disappointing results. All the points (except one) were clustered around the first visited point $\mathbf{x}^{n+1}$ (the same as in *CL*, by construction). This can be explained by the exaggeratedly low prediction given by Kriging at this very point: the mean predictor overshoots the data (because of the Gaussian covariance), and the expected improvement becomes abusively large in the neighborhood of $\mathbf{x}^{n+1}$. Then $\mathbf{x}^{n+2}$ is chosen near $\mathbf{x}^{n+1}$, and so on. The algorithm gets temporarily trapped at the first visited point. KB behaves in the same way as *CL* would do with a constant L below $\min(\mathbf{Y})$. As can be seen in Table 1 (last column), the phenomenon is visible on both the q-*PI* and q-*EI* criteria: they remain almost constant when q increases. This illustrates in particular how q-points criteria can help in rejecting unappropriate strategies.

In other respects, the results shown in Tab. 6.4.3 highlight a major drawback of the q-*PI* criterion. When q increases, the *PI* values associated with all 3 CL strategies quickly converge to 100%, such that it is not possible to discriminate between

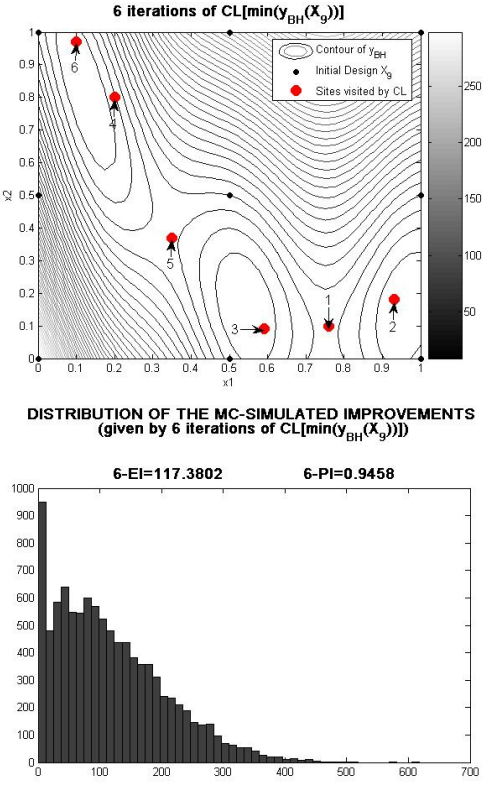


Fig. 6.5 (Left) contour of the y_{BH} function with the design $\mathbf{X}_9$ (small black points) and the 6 first points given by the heuristic strategy CL[min($y_{BH}(\mathbf{X}_9)$)] (large bullets). (Right) Histogram of 10^4 Monte Carlo simulated values of the improvement brought by the 6-points CL[min($y_{BH}(\mathbf{X}_9)$)] strategy. The corresponding estimates of 6-points PI and EI are given above

the good and the very good designs. The q-EI is a more selective measure thanks to taking the magnitude of possible improvements into account. Nevertheless, q-*EI* overevaluates the improvement associated with all designs considered here. This effect (already pointed out in [47]) can be explained by considering both the high value of σ^2 estimated from $\mathbf{Y}$ and the small difference between the minimal value reached at $\mathbf{X}_9$ (9.5) and the actual minimum of y_{BH} (0.4).

We finally compared CL[min], CL[max], latin hypercubes (LHS) and uniform random designs (UNIF) in terms of q-EI values, with $q \in [1,10]$. For every $q \in [1,10]$, we sampled 2000 q-elements designs of each type (LHS and UNIF) and compared the obtained empirical distributions of q-points Expected Improvement to the q-points Expected Improvement estimates associated with the q first points of both CL strategies.

Table 6.1 Multipoints PI, EI, and actual improvements for the 2, 6, and 10 first iterations of the heuristic strategies CL[min($\mathbf{Y}$)], CL[$mean$($\mathbf{Y}$)], CL[max($\mathbf{Y}$)], and Kriging Believer (here $\min(\mathbf{Y}) = \min(y_{BH}(\mathbf{X}_9))$). $q-PI$ and $q-EI$ are evaluated by Monte-Carlo simulations (Eq. 6.20, $n_{sim} = 10^4$)

	CL[min($\mathbf{Y}$)]	CL[$mean$($\mathbf{Y}$)]	CL[max($\mathbf{Y}$)]	KB
PI (first 2 points)	87.7%	87%	88.9%	65%
EI (first 2 points)	114.3	114	113.5	82.9
PI (first 6 points)	94.6%	95.5%	92.7%	65.5%
EI (first 6 points)	117.4	115.6	115.1	85.2
PI (first 10 points)	99.8%	99.9%	99.9%	66.5%
EI (first 10 points)	122.6	118.4	117	85.86
Improvement (first 6 points)	7.4	6.25	7.86	0
Improvement (first 10 points)	8.37	6.25	7.86	0

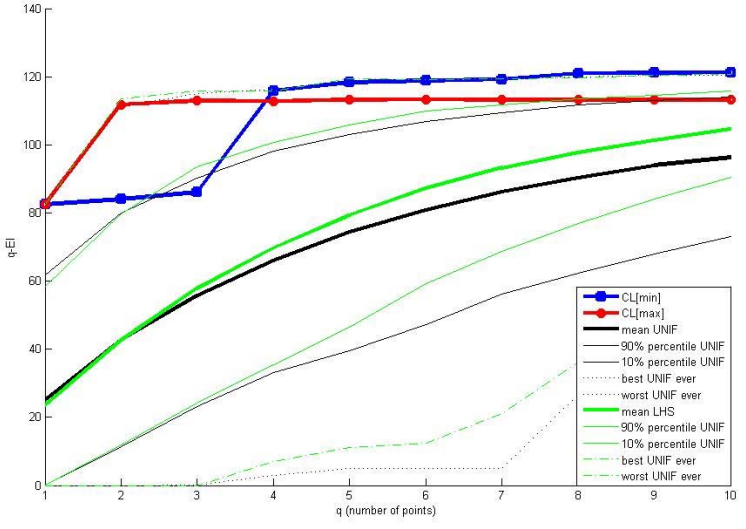


Fig. 6.6 Comparaison of the q-EI associated with the q first points ($q \in [1,10]$) given by the constant liar strategies (min and max), 2000 q-points designs uniformly drawn for every q, and 2000 q-points LHS designs taken at random for every q

As can be seen on fig. 6.6, CL[max] (light bullets) and CL[min] (dark squares) offer very good q-EI results compared to random designs, especially for small values of q. By definition, the two of them start with the 1-EI global maximizer, which ensures a q-EI at least equal to 83 for all $q \geq 1$. Both associated q-EI series then seem to converge to threshold values, almost reached for $q \geq 2$ by CL[max] (which dominates CL[min] when $q = 2$ and $q = 3$) and for $q \geq 4$ by CL[min] (which dominates CL[max] for all q s.t. $4 \leq q \leq 10$). The random designs have less promizing

q-EI expected values. Their q-EI distributions are quite dispersed, which can be seen for instance by looking at the $10\% - 90\%$ interpercentiles represented on fig. 6.6 by thin full lines (respectively dark and light for UNIF and LHS designs). Note in particular that the q-EI distribution of the LHS designs seem globally better than the one of the uniform designs. Interestingly, the best designs ever found among the UNIF designs (dark dotted lines) and among the LHS designs (light dotted lines) almost match with CL[max] when $q \in \{2,3\}$ and CL[min] when $4 \leq q \leq 10$. We haven't yet observed a design sampled at random that clearly provides better q-EI values than the proposed heuristic strategies.

6.5 Towards Kriging-Based Parallel Optimization: Conclusion and Perspectives

Optimization problems with objective functions stemming from expensive computer simulations strongly motivate the use of data-driven simplified mathematical representations of the simulator, or *metamodels*. An increasing number of optimization algorithms developed for such problems rely on metamodels, competing with and/or complementing population-based Computational Intelligence methods. A representative example is given by the EGO algorithm [22], a sequential black-box optimization procedure, which has gained popularity during the last decade and inspired numerous recent works in the field [10, 17, 18, 19, 20, 26, 28, 36, 44, 50]. EGO relies on a Kriging-based criterion, the expected improvement (EI), accounting for the exploration-exploitation trade-off[7]. The latter algorithm unfortunately produces only one point at each iteration, which prevents the user from taking advantage of parallel computation facilities. In the present work, we came back to the interpretation of Kriging in terms of Gaussian Process[39] in order to propose a framework for Kriging-based parallel optimization, and to prepare the work for parallel variants of EGO.

The probabilistic nature of the Kriging metamodel allowed us to calculate the joint probability distribution associated with the predictions at any set of points, upon which we could rediscover (see [47]) and characterize a criterion named here *multi-points expected improvement*, or q-*EI*. The q-*EI* criterion makes it possible to get an evaluation of the "optimization potential" given by any set of q new experiments. An analytical derivation of 2-*EI* was performed, providing a good example of how to manipulate joint Kriging distributions for choosing additional designs of experiments, and enabling us to shed more light on the nature of the q-*EI* thanks to selected figures. For the computation of q-*EI* in the general case, an alternative computation method relying on Monte-Carlo simulations was proposed. As pointed out and illustrated in the chapter, Monte-Carlo simulations offer indeed the opportunity to evaluate the q-*EI* associated with any given design of experiments, whatever its size n, and whatever the dimension of inputs d. However, deriving q-*EI*-optimal

[7] Other computational intelligence optimizers, e.g. evolutionary algorithms [9], address the exploration/exploitation trade-off implicitely through the choice of parameters such as the population size and the mutation probability.

designs on the basis of such estimates is not straightforward, and crucially depending on both n and d. Hence some greedy alternative problems were considered: four heuristic strategies, the "Kriging Believer" and three "Constant Liars" have been proposed and compared that aim at maximizing q-EI while being numerically tractable. It has been verified in the frame of a classical test case that the CL strategies provide q-EI values comparable with the best Latin Hypercubes and uniform designs of experiments found by simulation. This simple application illustrated a central practical conclusion of this work: considering a set of candidate designs of experiments, provided for instance by heuristic strategies, it is always possible — whatever n and d— to evaluate and rank them using estimates of q-EI or related criteria, thanks to conditional Monte-Carlo simulation.

Perspectives include of course the development of synchronous parallel EGO variants delivering a set of q points at each iteration. The tools presented in the chapter may constitute bricks of these algorithms, as it has very recently been illustrated on a succesful 6-dimensional test-case in the thesis [13]. An R package covering that subject is in an advanced stage of preparation and should be released soon [41]. On a longer term, the scope of the work presented in this chapter, and not only its modest original contributions, could be broaden. If the considered methods could seem essentially restricted to the Ordinary Kriging metamodel and concern the use of an optimization criterion meant to obtain q points in parallel, several degrees of freedom can be played on in order to address more general problems. First, any probabilistic metamodel potentially providing joint distributions could do well (regression models, smoothing splines, etc.). Second, the final goal of the new generated design might be to improve the global accuracy of the metamodel, to learn a quantile, to fill the space, etc : the work done here with the q-EI and associate strategies is just a particular case of what one can do with the flexibility offered by probabilistic metamodels and all possible decision-theoretic criteria. To finish with two challenging issues of Computationnal Intelligence, the following perspectives seem particularly relevant at both sides of the interface with this work:

- CI methods are needed to maximize the q-EI criterion, which inputs live in a $(n \times d)$-dimensional space, and which evaluation is noisy, with tunable fidelity depending on the chosen n_{sim} values,
- q-EI and related criteria are now at disposal to help pre-selecting good points in metamodel-assisted evolution strategies, in the flavour of [10].

Aknowledgements: This work was conducted within the frame of the DICE (Deep Inside Computer Experiments) Consortium between ARMINES, Renault, EDF, IRSN, ONERA, and Total S.A. The authors wish to thank X. Bay, R. T. Haftka, B. Smarslok, Y. Richet, O. Roustant, and V. Picheny for their help and rich comments. Special thanks to the R project people [6] for developing and spreading such a useful freeware. David moved to Neuchâtel University (Switzerland) for a postdoc, and he gratefully thanks the Mathematics Institute and the Hydrogeology Department for letting him spend time on the revision of the present chapter.

6.6 Appendix

6.6.1 Gaussian Processes for Machine Learning

A real-valued random process $(Y(\mathbf{x}))_{\mathbf{x}\in D}$ is called a *Gaussian Process* (GP) whenever all its finite-dimensional distributions are gaussian. Consequently, for all $n \in \mathbb{N}$ and for all set $\mathbf{X} = \{\mathbf{x}^1, ..., \mathbf{x}^n\}$ of n points of D, there exists a vector $\mathbf{m} \in \mathbf{R}^n$ and a symmetric positive semi-definite matrix $\Sigma \in \mathscr{M}_n(\mathbb{R})$ such that $(Y(\mathbf{x}^1), ..., Y(\mathbf{x}^n))$ is a gaussian Vector, following a multigaussian probability distribution $\mathscr{N}(\mathbf{m}, \Sigma)$. More specifically, for all $i \in [1,n]$, $Y(\mathbf{x}^i) \sim \mathscr{N}(\mathbb{E}[Y(\mathbf{x}^i)], Var[Y(\mathbf{x}^i)])$ where $\mathbb{E}[Y(\mathbf{x}^i)]$ is the ith coordinate of $\mathbf{m}$ and $Var[Y(\mathbf{x}^i)]$ is the ith diagonal term of Σ. Furthermore, all couples $(Y(\mathbf{x}^i), Y(\mathbf{x}^j))$ $i, j \in [1,n], i \neq j$ are multigaussian with a covariance $Cov[Y(\mathbf{x}^i), Y(\mathbf{x}^j)]$ equal to the non-diagonal term of Σ indexed by i and j.

A Random Process Y is said to be *first order stationary* if its mean is a constant, i.e. if $\exists \mu \in \mathbb{R} |\ \forall \mathbf{x} \in D,\ \mathbb{E}[Y(\mathbf{x})] = \mu$. A *first order stationary* process Y is said to be *second order stationary* if there exists furthermore a function of positive type, $c : D - D \longrightarrow \mathbb{R}$, such that for all pairs $(\mathbf{x}, \mathbf{x}') \in D^2$, $Cov[Y(\mathbf{x}), Y(\mathbf{x}')] = c(\mathbf{x} - \mathbf{x}')$. We then have the following expression for the covariance matrix of the observations at $\mathbf{X}$:

$$\Sigma := (Cov[Y(\mathbf{x}_i)\ Y(\mathbf{x}_j)])_{i\ j\in[1\ n]} = (c(\mathbf{x}_i - \mathbf{x}_j))_{i\ j\in[1\ n]} = \begin{pmatrix} \sigma^2 & c(\mathbf{x}_1 - \mathbf{x}_2) & c(\mathbf{x}_1 - \mathbf{x}_n) \\ c(\mathbf{x}_2 - \mathbf{x}_1) & \sigma^2 & c(\mathbf{x}_2 - \mathbf{x}_n) \\ & & \\ c(\mathbf{x}_n - \mathbf{x}_1) & c(\mathbf{x}_n - \mathbf{x}_2) & \sigma^2 \end{pmatrix} \tag{6.27}$$

where $\sigma^2 := c(0)$. Second order stationary processes are sometimes called *weakly stationary*. A major feature of GPs is that their *weak stationarity* is equivalent to *strong stationarity*: if Y is a weakly stationary GP, the law of probability of the random variable $Y(\mathbf{x})$ doesn't depend on $\mathbf{x}$, and the joint distribution of $(Y(\mathbf{x}^1), ..., Y(\mathbf{x}^n))$ is the same as the distribution of $(Y(\mathbf{x}^1 + \mathbf{h}), ..., Y(\mathbf{x}^n + \mathbf{h}))$ whatever the set of points $\{\mathbf{x}^1, ..., \mathbf{x}^n\} \in D^n$ and the vector $\mathbf{h} \in \mathbb{R}^n$ such that $\{\mathbf{x}^1 + \mathbf{h}, ..., \mathbf{x}^n + \mathbf{h}\} \in D^n$. To sum up, a stationary GP is entirely defined by its mean μ and its covariance function $c(.)$. The classical framework of Kriging for Computer Experiments is to make predictions of a costly simulator y at a new set of sites $\mathbf{X}_{new} = \{\mathbf{x}^{n+1}, ..., \mathbf{x}^{n+q}\}$ (most of the time, $q = 1$), on the basis of the collected observations at the initial design $\mathbf{X} = \{\mathbf{x}^1, ..., \mathbf{x}^n\}$, and under the assumption that y is one realization of a stationary GP Y with known covariance function c (in theory). Simple Kriging (SK) assumes a known mean, $\mu \in \mathbb{R}$. In Ordinary Kriging (OK), μ is estimated.

6.6.2 Conditioning Gaussian Vectors

Let us consider a centered Gaussian vector $V = (V_1, V_2)$ with covariance matrix

$$\Sigma_V = \mathbb{E}[VV^T] = \begin{pmatrix} \Sigma_{V_1} & \Sigma_{cross}^T \\ \Sigma_{cross} & \Sigma_{V_2} \end{pmatrix} \tag{6.28}$$

Key properties of Gaussian vectors include that the orthogonal projection of a Gaussian vector onto a linear subspace is still a Gaussian vector, and that the orthogonality of two subvectors V_1, V_2 of a Gaussian vector V (i.e. $\Sigma_{cross} = \mathbb{E}[V_2 V_1^T] = \mathbf{0}$) is equivalent to their independence. We now express the conditional expectation $\mathbb{E}[V_1|V_2]$. $\mathbb{E}[V_1|V_2]$ is by definition such that $V_1 - \mathbb{E}[V_1|V_2]$ is independent of V_2. $\mathbb{E}[V_1|V_2]$ is thus fully characterized as orthogonal projection on the vector space spanned by the components of V_2, solving the so called *normal equations*:

$$\mathbb{E}[(V_1 - \mathbb{E}[V_1|V_2])V_2^T] = 0 \tag{6.29}$$

Assuming linearity of $\mathbb{E}[V_1|V_2]$ in V_2, i.e. $\mathbb{E}[V_1|V_2] = AV_2$ ($A \in \mathscr{M}_n(\mathbb{R})$), a straightforward development of (eq.6.29) gives the matrix equation $\Sigma_{cross}^T = A\Sigma_{V_2}$, and hence $\Sigma_{cross}^T \Sigma_{V_2}^{-1} V_2$ is a suitable solution provided Σ_{V_2} is full ranked[8]. We conclude that

$$\mathbb{E}[V_1|V_2] = \Sigma_{cross}^T \Sigma_{V_2}^{-1} V_2 \tag{6.30}$$

by uniqueness of the orthogonal projection onto a closed linear subspace in a Hilbert space. Using the independence between $(V_1 - \mathbb{E}[V_1|V_2])$ and V_2, one can calculate the conditional covariance matrix $\Sigma_{V_1|V_2}$:

$$\begin{aligned}\Sigma_{V_1|V_2} &= \mathbb{E}[(V_1 - \mathbb{E}[V_1|V_2])(V_1 - \mathbb{E}[V_1|V_2])^T|V_2] = \mathbb{E}[(V_1 - AV_2)(V_1 - AV_2)^T] \\ &= \Sigma_{V_1} - A\Sigma_{cross} - \Sigma_{cross}^T A^T + A\Sigma_{V_2}A^T = \Sigma_{V_1} - \Sigma_{cross}^T \Sigma_{V_2}^{-1} \Sigma_{cross}\end{aligned} \tag{6.31}$$

Now consider the case of a non-centered random vector $V = (V_1, V_2)$ with mean $m = (m_1, m_2)$. The conditional distribution $V_1|V_2$ can be obtained by coming back to the centered random vector $V - m$. We then find that $\mathbb{E}[V_1 - m_1|V_2 - m_2] = \Sigma_{cross}^T \Sigma_{V_2}^{-1}(V_2 - m_2)$ and hence $\mathbb{E}[V_1|V_2] = m_1 + \Sigma_{cross}^T \Sigma_{V_2}^{-1}(V_2 - m_2)$.

6.6.3 Simple Kriging Equations

Let us come back to our metamodeling problem and assume that y is one realization of a Gaussian Process Y, defined as follows:

$$\begin{cases} Y(\mathbf{x}) = \mu + \varepsilon(\mathbf{x}) \\ \varepsilon(\mathbf{x}) \text{ centered stationary GP with covariance function } c(.) \end{cases} \tag{6.32}$$

where $\mu \in \mathbb{R}$ is known. Now say that Y has already been observed at n locations $\mathbf{X} = \{\mathbf{x}^1, ..., \mathbf{x}^n\}$ ($Y(\mathbf{X}) = \mathbf{Y}$) and that we wish to predict Y a q new locations $\mathbf{X}_{new} = \{\mathbf{x}^{n+1}, ..., \mathbf{x}^{n+q}\}$. Since $(Y(\mathbf{x}^1), ..., Y(\mathbf{x}^n), Y(\mathbf{x}^{n+1}), ..., Y(\mathbf{x}^{n+q}))$ is a Gaussian Vector with mean $\mu \mathbf{1}_{n+q}$ and covariance matrix

[8] If Σ_{V_2} is not invertible, the equation holds in replacing $\Sigma_{V_2}^{-1}$ by the pseudo-inverse $\Sigma_{V_2}^{\dagger}$.

$$\Sigma_{tot} = \begin{pmatrix} \Sigma & \Sigma_{cross}^T \\ \Sigma_{cross} & \Sigma_{new} \end{pmatrix} = \begin{pmatrix} \sigma^2 & c(\mathbf{x}_1 - \mathbf{x}_2) & \dots & c(\mathbf{x}_1 - \mathbf{x}_{n+q}) \\ c(\mathbf{x}_2 - \mathbf{x}_1) & \sigma^2 & \dots & c(\mathbf{x}_2 - \mathbf{x}_{n+q}) \\ \dots & \dots & \dots & \dots \\ c(\mathbf{x}_{n+q} - \mathbf{x}_1) & c(\mathbf{x}_{n+q} - \mathbf{x}_2) & \dots & \sigma^2 \end{pmatrix} \quad (6.33)$$

We can directly apply eq. (6.30) and eq. (6.31) to derive the Simple Kriging Equations:

$$[Y(\mathbf{X}_{new})|Y(\mathbf{X}) = \mathbf{Y}] \sim \mathcal{N}(m_{SK}(\mathbf{X}_{new}), \Sigma_{SK}(\mathbf{X}_{new})) \quad (6.34)$$

with $m_{SK}(\mathbf{X}_{new}) = \mathbb{E}[Y(\mathbf{X}_{new})|Y(\mathbf{X}) = \mathbf{Y}] = \mu \mathbf{1}_q + \Sigma_{cross}^T \Sigma^{-1}(\mathbf{Y} - \mu \mathbf{1}_q)$ and $\Sigma_{SK}(\mathbf{X}_{new}) = \Sigma_{new} - \Sigma_{cross}^T \Sigma^{-1} \Sigma_{cross}$. When $q = 1$, $\Sigma_{cross} = \mathbf{c}(\mathbf{x}^{n+1}) = Cov[Y(\mathbf{x}^{n+1}), Y(\mathbf{X})]$ and the covariance matrix reduces to $s_{SK}^2(\mathbf{x}) = \sigma^2 - \mathbf{c}(\mathbf{x}^{n+1})^T \Sigma^{-1} \mathbf{c}(\mathbf{x}^{n+1})$, which is called the *Kriging Variance*. Note that when μ is constant but not known in advance, it is not mathematically correct to sequentially estimate μ and plug in the estimate in the Simple Kriging equations. Ordinary Kriging addresses this issue.

6.6.4 Ordinary Kriging Equations

Compared to Simple Kriging, Ordinary Kriging (OK) is used when the mean of the underlying random process is constant and unknown. We give here a derivation of OK in a Bayesian framework, assuming that μ has an improper uniform prior distribution $\mu \sim \mathcal{U}(\mathbb{R})$. y is thus seen as a realization of a random process Y, defined as the sum of μ and a centered GP[9]:

$$\begin{cases} Y(\mathbf{x}) = \mu + \varepsilon(\mathbf{x}) \\ \varepsilon(\mathbf{x}) \text{ centered stationary GP with covariance function } c(.) \\ \mu \sim \mathcal{U}(\mathbb{R}) \text{ (prior), independent of } \varepsilon \end{cases} \quad (6.35)$$

Note that conditioning with respect to μ actually provides SK equations. Letting μ vary, we aim to find the law of $[Y(\mathbf{X}_{new})|Y(\mathbf{X}) = \mathbf{Y}]$. Starting with $[Y(\mathbf{X}) = \mathbf{Y}|\mu] \sim \mathcal{N}(\mu \mathbf{1}_n, \Sigma)$, we get μ's posterior distribution:

$$[\mu|Y(\mathbf{X}) = \mathbf{Y}] \sim \mathcal{N}\left(\hat{\mu}, \sigma_\mu^2\right) = \mathcal{N}\left(\frac{\mathbf{1}^T \Sigma^{-1} \mathbf{Y}}{\mathbf{1}^T \Sigma^{-1} \mathbf{1}}, \frac{1}{\mathbf{1}_q^T \Sigma^{-1} \mathbf{1}_q}\right) \text{ (posterior)} \quad (6.36)$$

We can re-write the SK equations $[Y(\mathbf{X}_{new})|Y(\mathbf{X}) = \mathbf{Y}, \mu] \sim \mathcal{N}(m_{SK}(\mathbf{X}_{new}), \Sigma_{SK}(\mathbf{X}_{new}))$. Now it is very useful to notice that the random vector $(Y(\mathbf{X}_{new}), \mu)$ is Gaussian conditional on $Y(\mathbf{X}) = \mathbf{Y}$[10]. It follows that $[Y(\mathbf{X}_{new})|Y(\mathbf{X}) = \mathbf{Y}]$ is Gaussian, and its mean and covariance matrix can finally be calculated with the help of classical conditional calculus results. Hence using $m_{OK}(\mathbf{X}_{new}) = \mathbb{E}[Y(\mathbf{X}_{new})|Y(\mathbf{X}) = \mathbf{Y}] = \mathbb{E}_\mu[\mathbb{E}[Y(\mathbf{X}_{new})|Y(\mathbf{X}) = \mathbf{Y}, \mu]]$, we find that $m_{OK}(\mathbf{X}_{new}) = \hat{\mu} + \Sigma_{cross}^T \Sigma^{-1}(\mathbf{Y} - \hat{\mu} \mathbf{1}_n)$. Similarly, $\Sigma_{OK}(\mathbf{X}_{new})$ can be obtained using that $Cov[A, B] = Cov[\mathbb{E}[A|C], \mathbb{E}[B|C]] + \mathbb{E}[Cov[A, B|C]]$ for all random variables

[9] The resulting random process Y is not Gaussian.

[10] Which can be proved by considering its Fourier transform.

A,B, C such that all terms exist. We then get for all couples of points $(\mathbf{x}^{n+i}, \mathbf{x}^{n+j})$ $(i, j \in [1,q])$:

$$\begin{aligned} &Cov[Y(\mathbf{x}^{n+i}), Y(\mathbf{x}^{n+j})|Y(\mathbf{X}) = \mathbf{Y}] \\ =&\mathbb{E}\left[Cov[Y(\mathbf{x}^{n+i}), Y(\mathbf{x}^{n+j})|Y(\mathbf{X}) = \mathbf{Y}, \mu]\right] + Cov\left[\mathbb{E}[Y(\mathbf{x}^{n+i})|Y(\mathbf{X}) = \mathbf{Y}, \mu], \mathbb{E}[Y(\mathbf{x}^{n+j})|Y(\mathbf{X}) = \mathbf{Y}, \mu]\right]. \end{aligned} \tag{6.37}$$

The left term $Cov[Y(\mathbf{x}^{n+i}), Y(\mathbf{x}^{n+j})|Y(\mathbf{X}) = \mathbf{Y}, \mu]$ is the conditional covariance under the Simple Kriging Model. The right term is the covariance between $\mu + \mathbf{c}(\mathbf{x}^{n+i})^T \Sigma^{-1}(\mathbf{Y} - \mu \mathbf{1}_q)$ and $\mu + \mathbf{c}(\mathbf{x}^{n+j})^T \Sigma^{-1}(\mathbf{Y} - \mu \mathbf{1}_q)$ conditional on the observations $Y(\mathbf{X}) = \mathbf{Y}$. Using eq. 6.36, we finally obtain:

$$\begin{aligned} &Cov[Y(\mathbf{x}^{n+i}), Y(\mathbf{x}^{n+j})|Y(\mathbf{X}) = \mathbf{Y}] \\ =&Cov_{SK}[Y(\mathbf{x}^{n+i}), Y(\mathbf{x}^{n+j})|Y(\mathbf{X}) = \mathbf{Y}] \\ &+Cov[\mathbf{c}(\mathbf{x}^{n+i})^T \Sigma^{-1}(\mathbf{Y}) + \mu(1 + \mathbf{c}(\mathbf{x}^{n+i})^T \Sigma^{-1} \mathbf{1}_q), \mathbf{c}(\mathbf{x}^{n+j})^T \Sigma^{-1}(\mathbf{Y}) + \mu(1 + \mathbf{c}(\mathbf{x}^{n+j})^T \Sigma^{-1} \mathbf{1}_q)] \\ =&c(\mathbf{x}^{n+i} - \mathbf{x}^{n+j}) - \mathbf{c}(\mathbf{x}^{n+i})^T \Sigma^{-1} \mathbf{c}(\mathbf{x}^{n+j}) + \frac{(1 + \mathbf{c}(\mathbf{x}^{n+i})^T \Sigma^{-1} \mathbf{1}_q)(1 + \mathbf{c}(\mathbf{x}^{n+j})^T \Sigma^{-1} \mathbf{1}_q)}{\mathbf{1}_q^T \Sigma^{-1} \mathbf{1}_q}. \end{aligned} \tag{6.38}$$

References

1. Abrahamsen, P.: A review of gaussian random fields and correlation functions, 2nd edn. Tech. Rep. 917, Norwegian Computing Center, Olso (1997)
2. Antoniadis, A., Berruyer, J., Carmona, R.: Régression non linéaire et applications. Economica, Paris (1992)
3. Baker, C., Watson, L.T., Grossman, B., Mason, W.H., Haftka, R.T.: Parallel global aircraft configuration design space exploration. Practical parallel computing, 79–96 (2001)
4. Bishop, C.: Neural Networks for Pattern Recognition. Oxford Univ. Press, Oxford (1995)
5. Blum, C.: Ant colony optimization: introduction and recent trends. Physics of Life Review 2, 353–373 (2005)
6. development Core Team R: R: A language and environment for statistical computing (2006), http://www.R-project.org
7. Cressie, N.: Statistics for spatial data. Wiley series in probability and mathematical statistics (1993)
8. Dreyfus, G., Martinez, J.M.: Réseaux de neurones. Eyrolles (2002)
9. Eiben, A., Smith, J.: Introduction to Evolutionary Computing. Springer, Heidelberg (2003)
10. Emmerich, M., Giannakoglou, K., Naujoks, B.: Single-and multiobjective optimization assisted by gaussian random field metamodels. IEEE Transactions on Evolutionary Computation 10(4), 421–439 (2006)
11. Geman, D., Jedynak, B.: An active testing model for tracking roads in satellite images. Tech. rep., Institut National de Recherches en Informatique et Automatique (INRIA) (December 1995)
12. Genton, M.: Classes of kernels for machine learning: A statistics perspective. Journal of Machine Learning Research 2, 299–312 (2001)

13. Ginsbourger, D.: Multiples métamodèles pour l'approximation et l'optimisation de fonctions numériques multivariables. PhD thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne (2009)
14. Ginsbourger, D., Le Riche, R., Carraro, L.: A multipoints criterion for parallel global optimization of deterministic computer experiments. In: Non-Convex Programming 2007 (2007)
15. Goria, S.: Evaluation d'un projet minier: approche bayésienne et options réelles. PhD thesis, Ecole des Mines de Paris (2004)
16. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer, Heidelberg (2001)
17. Henkenjohann, N., Göbel, R., Kleiner, M., Kunert, J.: An adaptive sequential procedure for efficient optimization of the sheet metal spinning process. Qual. Reliab. Engng. Int. 21, 439–455 (2005)
18. Huang, D., Allen, T., Notz, W., Miller, R.: Sequential Kriging optimization using multiple fidelity evaluations. Sructural and Multidisciplinary Optimization 32, 369–382 (2006)
19. Huang, D., Allen, T., Notz, W., Zheng, N.: Global optimization of stochastic black-box systems via sequential Kriging meta-models. Journal of Global Optimization 34, 441–466 (2006)
20. Jones, D.: A taxonomy of global optimization methods based on response surfaces. Journal of Global Optimization 21, 345–383 (2001)
21. Jones, D., Pertunen, C., Stuckman, B.: Lipschitzian optimization without the lipschitz constant. Journal of Optimization Theory and Application 79(1) (1993)
22. Jones, D., Schonlau, M., Welch, W.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization 13, 455–492 (1998)
23. Journel, A.: Fundamentals of geostatistics in five lessons. Tech. rep., Stanford Center for Reservoir Forecasting (1988)
24. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE Intl. Conf. on Neural Networks, vol. 4, pp. 1942–1948 (1995)
25. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220, 671–680 (1983)
26. Knowles, J.: Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionnary Computation (2005)
27. Koehler, J., Owen, A.: Computer experiments. Tech. rep., Department of Statistics, Stanford University (1996)
28. Kracker, H.: Methoden zur analyse von computerexperimenten mit anwendung auf die hochdruckblechumformung. Master's thesis, Dortmund University (2006)
29. Krige, D.: A statistical approach to some basic mine valuation problems on the witwatersrand. J. of the Chem., Metal. and Mining Soc. of South Africa 52(6), 119–139 (1951)
30. Martin, J., Simpson, T.: A monte carlo simulation of the Kriging model. In: 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY, AIAA, AIAA-2004-4483, August 30 - September 2 (2004)
31. Martin, J., Simpson, T.: Use of Kriging models to approximate deterministic computer models. AIAA Journal 43(4), 853–863 (2005)
32. Matheron, G.: Principles of geostatistics. Economic Geology 58, 1246–1266 (1963)
33. Matheron, G.: La théorie des variables régionalisées et ses applications. Tech. rep., Centre de Morphologie Mathématique de Fontainebleau, Ecole Nationale Supérieure des Mines de Paris (1970)

34. O'Hagan, A.: Bayesian analysis of computer code outputs: a tutorial. Reliability Engineering and System Safety 91(91), 1290–1300 (2006)
35. Paciorek, C.: Nonstationary gaussian processes for regression and spatial modelling. PhD thesis, Carnegie Mellon University (2003)
36. Ponweiser, W., Wagner, T., Biermann, D., Vincze, M.: Multiobjective optimization on a limited budget of evaluations using model-assisted S-metric selection. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 784–794. Springer, Heidelberg (2008)
37. Praveen, C., Duvigneau, R.: Radial basis functions and Kriging metamodels for aerodynamic optimization. Tech. rep., INRIA (2007)
38. Queipo, N., Verde, A., Pintos, S., Haftka, R.: Assessing the value of another cycle in surrogate-based optimization. In: 11th Multidisciplinary Analysis and Optimization Conference, AIAA (2006)
39. Rasmussen, C., Williams, K.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2006)
40. Ripley, B.: Stochastic Simulation. John Wiley and Sons, New York (1987)
41. Roustant, O., Ginsbourger, D., Deville, Y.: The DiceKriging package: Kriging-based metamodeling and optimization for computer experiments. In: The UseR! Conference, Agrocampus-Ouest, Rennes, France (2009)
42. Sacks, J., Welch, W., Mitchell, T., Wynn, H.: Design and analysis of computer experiments. Statistical Science 4(4), 409–435 (1989)
43. Santner, T., Williams, B., Notz, W.: The Design and Analysis of Computer Experiments. Springer, Heidelberg (2003)
44. Sasena, M.: Flexibility and efficiency enhancements for constrained global design optimization with Kriging approximations. PhD thesis, University of Michigan (2002)
45. Sasena, M.J., Papalambros, P., Goovaerts, P.: Exploration of metamodeling sampling criteria for constrained global optimization. Journal of Engineering Optimization (2002)
46. Sasena, M.J., Papalambros, P.Y., Goovaerts, P.: Global optimization of problems with disconnected feasible regions via surrogate modeling. In: Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA (2002)
47. Schonlau, M.: Computer experiments and global optimization. PhD thesis, University of Waterloo (1997)
48. Schonlau, M., Welch, W., Jones, D.: A data-analytic approach to bayesian global optimization. In: Proceedings of the A.S.A. (1997)
49. Ulmer, H., Streichert, F., Zell, A.: Evolution strategies assisted by gaussian processes with improved pre-selection criterion. Tech. rep., Center for Bioinformatics Tuebingen, ZBIT (2003)
50. Villemonteix, J.: Optimisation de fonctions coûteuses: Modèles gaussiens pour une utilisation théorie et pratique industrielle. PhD thesis, Université Paris-sud XI, Faculté des Sciences d'Orsay (2008)
51. Villemonteix, J., Vazquez, E., Walter, E.: An informational approach to the global optimization of expensive-to-evaluate functions. Journal of Global Optimization 44(4), 509–534 (2009)
52. Wahba, G.: Spline Models for Observational Data. SIAM, Philadelphia (1990)
53. Williams, C., Rasmussen, C.: Gaussian processes for regression. In: Advances in Neural Information Processing Systems, vol. 8 (1996)

Chapter 7
Analysis of Approximation-Based Memetic Algorithms for Engineering Optimization

Frederico Gadelha Guimarães, David Alister Lowther, and Jaime Arturo Ramírez

Abstract. This chapter discusses the treatment of expensive optimization problems in Computer-Aided Design (CAD) problems by combining two strategies. First, we perform the whole optimization varying the accuracy with which a given candidate solution is evaluated by the expensive black-box function, rather than using the same accuracy for all evaluations. This idea follows from the fact that evolutionary algorithms, in general, employ more searching effort on the most promising regions of the search domain. We can adopt the same principles for allocating more computational effort when evaluating candidate solutions within these regions. The second strategy is the employment of local approximations within the local search operator of memetic algorithms. Since the points in the data are evaluated with different accuracies, the approximation methodology should give greater weight to samples evaluated with higher precision. The chapter proceeds to the formal analysis of approximation-based memetic algorithms, in which we investigate the effect of the local search operators on the global convergence properties of evolutionary algorithms via Markov chain theory, and also study the computational complexity of the approximation-based local search operator. The chapter concludes with the illustration of the methodology in the design of electromagnetic devices, as an example of an expensive optimization problem.

Frederico Gadelha Guimarães
Department of Computer Science, Universidade Federal de Ouro Preto, Ouro Preto, MG, Brazil
e-mail: frederico.g.guimaraes@gmail.com

David Alister Lowther
Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada
e-mail: david.lowther@mcgill.ca

Jaime Arturo Ramírez
Department of Electrical Engineering, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil
e-mail: jramirez@ufmg.br

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 163–191.
springerlink.com

7.1 Memetic Algorithms and Computer-Aided Design

In a more general context, computer-aided design (CAD) refers to the application of computers to the design process. In this sense, a CAD system is a computational system used in engineering, geology, and architecture for technical designs. Here, we refer to CAD processes in a more strict sense, intending to refer in a special manner to the contexts of systems engineering and engineering optimization, in which the CAD process is viewed as an automated process involving the association of a mathematical and computational model of the device to be designed and an adequate optimization method for finding the optimal values of the design variables [21, 31]. In the context of this chapter, the device represents an electromagnetic device such as an electrical motor, a transformer, an antenna, a wave guide, etc., but the methodology presented is general enough to be applied to computationally intensive optimization problems in other contexts. Figure 7.1 presents a flowchart description of the basic design process. In this flowchart, we can identify the fundamental blocks of a complete design process. The process starts with the establishment of specifications and requirements, which leads to the definition of a prototype, a first design model to solve the problem at hand. These initial steps depend on the knowledge and intervention of a human designer, or a team of designers. The main purpose of the prototype, and its associated computational model, is to provide a parameterized search space, and a set of objective and constraint functions for the optimization process. The goal of the optimization process is to find the solution, in the given search space, that moves the current design to a prototype which will satisfy the requirements as closely as possible. Finally, if the optimized prototype satisfies the specifications and requirements defined by the designer, then the design process stops and the final solution is achieved.

The product specifications and requirements are translated into objective and constraint functions, and the design variables into optimization variables with their respective ranges. In mathematical terms, the design problem can be defined as an optimization problem of the form

$$\begin{aligned} &\min_{\mathbf{x}} \mathbf{f}(\mathbf{x}) \in \mathbb{R}^{n_f} \\ &\text{subject to: } \mathbf{x} \in \mathscr{F}_x \subset \mathbb{R}^{n_x} \end{aligned} \tag{7.1}$$

in which $\mathbf{x}$ is the vector of optimization variables, $\mathbf{f}(\cdot) : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_f}$ are the objective functions and $\mathscr{F}_x$ is the feasible set, mathematically defined by the constraint functions. This optimization problem requires an adequate optimization algorithm to search for the best solutions. During the search process, the evaluation of each possible design demands the solution of partial differential equations that describe the laws that physically govern its behavior. When nonlinearities and accuracy requirements are incorporated, the computational cost becomes even more relevant. In coupled problems, the analysis step may also involve the association of different analysis softwares as in [2, 41], which further increases the computational cost. This synthesis-analysis cycle is shown in the inner loop of the flowchart in Figure 7.1. When the prototype is optimized, its performance is evaluated to verify if it meets

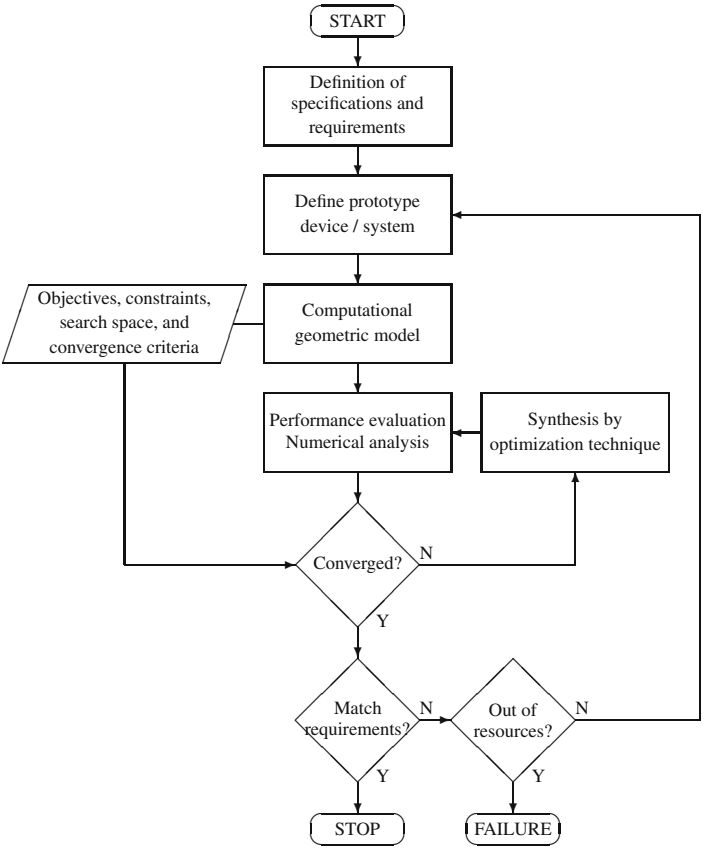


Fig. 7.1 Flowchart description of the computer-aided design process

the requirements defined in the beginning of the CAD process. If it does not meet the requirements, we can either give up, due to lack of resources and rethink the design specifications, or return to step 2 to modify the prototype, hence modifying the objective and contraint functions and the search space.

For the sake of simplicity, the software responsible for solving the analysis problem will be treated as an expensive "black-box", of which we know only the outputs - the behavior it produces - due to different inputs. From the optimization point of view, it does not matter what is inside this black-box, as long as it provides consistent outputs. In some problems, all objective and constraint functions are functions of the black-box output value. In this case, the time to evaluate one individual in a population-based evolutionary algorithm is simply the time consumed by the black-box. In other cases, only some of these functions depend on the black-box software, while the others are analytically defined and usually inexpensive.

Evolutionary Algorithms (EAs) play an important role in the solution of complex CAD problems [10, 17, 31], because they can deal with problems that are

discontinuous, nonconvex, multimodal, noisy, and present correlations or higher-order dependencies among variables. Moreover, these algorithms can explore simultaneous regions of the search space in a single run and evolve a population of candidate solutions that represent sub-optimal solutions or trade-off solutions in the context of multiobjective problems. The bottom line is that the same principles in nature that are responsible for the diversity and complexity of biological systems can be applied to the solution of complex design problems in engineering. Thanks to the work of many researchers since the 1950's, and contributions of the last two decades that helped to popularize and mature these techniques, evolutionary algorithms are today an important computational intelligence methodology for complex designs and an established field of research [4, 6, 9].

Memetic Algorithms (MAs) [24, 25], a term initially coined by Pablo Moscato in 1989[1], represent a particular class of EAs that employ local search methods inside their cycle. MAs first appeared for combinatorial optimization problems [14, 23, 25], but it did not take long for continuous search space versions of such algorithms to appear [18, 22, 27]. MAs for multiobjective problems have also been proposed, see [19, 39]. Any successful global search meta-heuristic method, including evolutionary techniques, should find a good balance between its *exploitation* and *exploration* components[2]. MAs approach the *exploration-exploitation* balance by trying to find a good association of global search mechanisms and local search operators, which can favor the optimization process as a whole. In fact, some deterministic search methods, although capable of local search only, present fast and precise convergence properties that surpass the properties of EAs, which, in contrast, have poorer convergence and accuracy. There are very specialized methods for treating constraints in optimization problems, including equality constraints, that rely on assumptions that favor their utilization as local search techniques [5, 37]. All these characteristics suggest the use of hybrid strategies, in order to enhance the local search properties of typical EAs.

In the first versions of MAs, the local search method is applied to each individual generated by the reproductive operators, leading this offspring to its closest local optimum. Such MAs have presented good performance in some contexts, since the search space to be explored by the global search side of the hybrid algorithm is then greatly reduced. This reduction has a price to be paid, which is the computational cost of the local search, limiting the application of MAs in many expensive-to-evaluate CAD problems involving continuous-variable search spaces. Later versions of MAs have relaxed this definition, by not applying the local search to all individuals and sometimes not applying the local search in all generations. The specification of the local search intensity and frequency is referred to as the balance between global and local search [15]. Some approaches relax the accuracy of the local optimizer [18] and other strategies relax the requirement that the local search is applied up to local optimality, only a local enhancement is required [20]. In

[1] Moscato was inspired by the concept of memes as proposed by Richard Dawkins in 1976 [7].

[2] *Exploration* is related to the global search capability of the algorithm, while *Exploitation* means spending more searching effort in the most promising regions.

this chapter, we follow the taxonomy presented by Krasnogor in [20], and adopt the term Memetic Algorithm for evolutionary algorithms that employ any local search method to some or all individuals of the population, but this is not a consensus, and different opinions exist [27, 29].

This chapter deals with Approximation-Based Memetic Algorithms ($\mathscr{A}$-MAs), which is a category of MAs that employ approximation techniques within the local search phase, using samples produced by the algorithm itself in previous generations. These $\mathscr{A}$-MAs have arisen recently to deal specifically with expensive-to-evaluate black-box functions in continuous-variable search spaces, in order to reduce the computational overhead of the local search phase while still benefitting from the principles of MAs [11, 12, 28, 38, 39, 42]. The use of approximations, or surrogate models, in evolutionary optimization is well discussed in the literature [16], but the employment of local approximations, specially in the context of MAs is rather recent [36]. The algorithms in this particular class of MAs rely strongly on the sampling properties of population-based EAs, whose heuristic operators tend to concentrate more samples in the most promising regions of the search space. Given the computational effort spent to obtain each of these samples, it is a good idea to store them and use some of this information to build local approximations of these black-box functions for the local search operator. With these computationally less expensive-to-evaluate local approximations in hand, the local search can be used to enhance some individuals of the population. By using approximations, the local search is not exact, but this local search engine can potentially increase the convergence properties of typical EAs without the high computational cost in terms of function evaluations required when applying the local search method directly to the real functions. Note that, unlike usual MAs, the local search phase in $\mathscr{A}$-MAs is intended to be as inexpensive as possible, considering the number of function evaluations, usually not requiring any additional evaluation of the black-box functions.

We propose a methodology for the local search in $\mathscr{A}$-MAs that is based on Radial Basis Function (RBF) approximations [13, 30] of the expensive black-box functions. The auxiliary local search problem based on these RBF approximations is then solved by the Sequential Quadratic Programming (SQP) method [8], which is a fast and accurate method for constrained mono-objective problems, having a quite predictable behavior. This procedure locally enhances one individual of the population, making it an MA according to [20].

Adittionally, we combine this framework of $\mathscr{A}$-MAs with another strategy for treating expensive optimization problems: we perform the whole optimization varying the accuracy with which a given candidate solution is evaluated by the expensive black-box function, rather than using the same accuracy for all evaluations. We allocate more computational effort for evaluating the candidate solutions around the best regions and also as the evolutionary search converges. When high accuracy is required for the *final* solution of the CAD problem, we can reduce the overall computational cost by varying the accuracy (and thus the computational cost) of the objective function when analyzing points tested by the algorithm *during* the optimization process. This has an impact in the approximation methodology used in the local search.

The chapter proceeds to the formal analysis of $\mathscr{A}$-MAs. This analysis is divided in two main parts. The first part investigates the effect of the local search operators on the global convergence properties of evolutionary algorithms via Markov chain theory [26]. The second part studies the computational cost of $\mathscr{A}$-MAs. In this second part, the computational complexity of the approximation-based local search operators is derived and expressions for the overhead of the local search are presented. The chapter concludes with the illustration of the methodology in the design of electromagnetic devices, in which the evaluation of candidate solutions requires the numerical solution of partial differential equations, making it an expensive optimization problem.

7.2 Approximation-Based Memetic Algorithms

The general structure of instances of $\mathscr{A}$-MAs is detailed in Algorithm 7.1. This structure is very general and can accomodate many well-known evolutionary algorithms. P_k is the online (or parent) population of size μ, Q_k is the offspring population of size λ. For instance, genetic algorithms have $\mu = \lambda$, while evolution strategies usually have $\mu < \lambda$. Q_k is produced after the application of the reproduction operators on P_k, see lines 6 and 17. The reproduction procedure makes reference to rather general procedures in EAs. In genetic algorithms, reproduction comprises fitness-proportional selection, crossover and mutation operators. In other EAs, different specific operators apply. The search space is defined by the lower and upper bounds of each variable such as

$$\mathscr{S} = \{\mathbf{x} \in \mathbb{R}^{n_x} : x_i^- \leq x_i \leq x_i^+\} \tag{7.2}$$

The inequality constraints $\mathbf{g}(\cdot) : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_g}$ and the equality constraints $\mathbf{h}(\cdot) : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_h}$ define the feasible region:

$$\mathscr{F}_x = \mathscr{S} \bigcap \left(\bigcap_{i=1}^{n_g} \mathscr{G}_i \right) \bigcap \left(\bigcap_{j=1}^{n_h} \mathscr{H}_j \right) \tag{7.3}$$

where $\mathscr{G}_i = \{\mathbf{x} \in \mathbb{R}^{n_x} : g_i(\mathbf{x}) \leq 0\}$ and $\mathscr{H}_j = \{\mathbf{x} \in \mathbb{R}^{n_x} : h_j(\mathbf{x}) = 0\}$.

The archive or offline population A_k stores the best solution set found by the algorithm. Usually, in mono-objective optimization problems, A_k has size $\xi = 1$, that is, only the global best solution found to date is stored and maintained. When we are interested in finding not only the global one but also a set of optima, A_k may store more than one solution, as is the case in some niching evolutionary algorithms or multimodal immune-based algorithms. The update method, see lines 5 and 18, is slightly more complicated, since some diversity mechanism should be imposed on the solution set. In multi-objective problems, we have $\xi > 1$ and the update method is also more complicated, since we need to consider dominance relations and a good representation of the Pareto front.

Algorithm 7.1. Generic structure of an approximation-based MA.

Data: population size μ, offspring size λ, maximum archive size ξ, search space $\mathscr{S}$, objective and constraint functions $\mathbf{f}(\cdot)$, $\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$.
Result: Estimate(s) of $\mathscr{S}^*$ in the archive population A_k.

1 $k \leftarrow 0$ /* Generation counter */
2 $P_k = \left\{\mathbf{p}_k^{(1)}, \ldots, \mathbf{p}_k^{(\mu)}\right\} \leftarrow$ Initialize population$(\mu, \mathscr{S})$;
3 $A_k = \varnothing \leftarrow$ Initialize archive /* Keep the best solution(s) */
4 $\Phi_{P_k} \leftarrow$ Evaluate fitness$(P_k, \mathbf{f}, \mathbf{g}, \mathbf{h})$;
5 $A_k \leftarrow$ Update(A_k, P_k, ξ);
// Reproduction comprises selection and variation
6 $Q_k = \left\{\mathbf{q}_k^{(1)}, \ldots, \mathbf{q}_k^{(\lambda)}\right\} \leftarrow$ Reproduction(P_k, A_k, Φ_{P_k});
7 $\mathscr{D} \leftarrow$ Update data$(\{P_k; \mathbf{f}(P_k); \mathbf{g}(P_k); \mathbf{h}(P_k)\})$ /* Global data set */
8 **while** $\neg$ *stop criteria* **do**
9 $\quad \Phi_{Q_k} \leftarrow$ Evaluate fitness$(Q_k, \mathbf{f}, \mathbf{g}, \mathbf{h})$;
10 $\quad \mathscr{D} \leftarrow$ Update data$(\{Q_k; \mathbf{f}(Q_k); \mathbf{g}(Q_k); \mathbf{h}(Q_k)\})$;
11 $\quad$ **if** mod $(k, n_L) = 0$ **then**
12 $\quad\quad$ **for** *each of the* σ *best individuals* **do**
13 $\quad\quad\quad$ Approximation-based local search operator /* See Algorithm 7.3 */
14 $\quad\quad$ **end**
15 $\quad$ **end**
16 $\quad P_{k+1} \leftarrow$ Substitution(P_k, Q_k);
17 $\quad Q_{k+1} \leftarrow$ Reproduction(P_{k+1});
18 $\quad A_{k+1} \leftarrow$ Update$(A_k, P_k \cup Q_k, \xi)$;
19 $\quad k \leftarrow k+1$;
20 **end**

The substitution operator creates the next parent population. In genetic algorithms, the offspring simply substitutes for the previous parent population. In evolution strategies, substitution schemes such as the $ES(\mu+\lambda)$ or $ES(\mu,\lambda)$ are employed [4]. Other deterministic substitution schemes are used in differential evolution and evolutionary programming algorithms.

The main point that is expressed in this general structure is that the basic evolutionary operators, as well as the generic evolutionary structure, are essentially preserved in the structure of $\mathscr{A}$-MAs. The local search phase is described in lines 11–15, after the fitness assignment of the offspring population Q_k. It introduces the following parameters:

- the interval of generations in which the local search is to be applied, denoted by $n_L \geq 0$. For example, if $n_L = 0$, the local search operator is applied at every generation. If $n_L = 4$, the local search operator is applied at every four generations.
- the number of individuals in the Q_k population that will be subject to local search, denoted by $0 \leq \sigma \leq \mu$.

There is no general rule for choosing the values of these parameters. They are related to the frequency and the intensity of the local search in the memetic algorithm. The value of n_L cannot be too small, because the local search is more efficient when new information is available, that is, when new samples are generated. It cannot be too big either, because the local search has to be applied in a number of generations that is enough for producing some effect on the performance of the algorithm. The value of σ should be less than μ. An intuitive argument is that the approximation-based local search operators rely on the samples available, and therefore they would not work properly if applied to all individuals, simply because there are not enough samples to generate local approximations around each individual. Therefore, the local search in $\mathscr{A}$-MAs should be applied to few individuals. In the next section we provide a formal argument based on convergence properties.

In this section, we discuss the treatment of expensive optimization problems by combining two strategies. First, we do not perform the whole optimization with great accuracy for all evaluations, but only for some of them. The black-box functions are evaluated with varying accuracy instead of using a fixed accuracy in the whole process. The second strategy is the employment of local RBF approximations within the local search operator of $\mathscr{A}$-MAs. The approximations should be flexible given that the points in the data were not evaluated with the same accuracy.

7.2.1 Varying Accuracy in Black-Box Functions

In general, expensive black-box functions have a computational cost that is approximately constant for a given accuracy requirement. In this section, we propose an adaptive variation of the accuracy when evaluating candidate solutions. We argue that, when high accuracy solutions are required to a given problem, one does not need to perform the whole evolutionary search making all evaluations at this high accuracy level, but only for some of them, thus saving computational effort. This idea follows from the key observation that EAs in general employ more searching effort on the most promising regions of the search domain. We can adopt the same principles of the exploration-exploitation balance to allocate more computational effort when evaluating candidate solutions within these most promising regions.

The parameters that define the accuracy, and hence the computational cost, of the objective function are provided together with the evaluation point $\mathbf{x}$. We assume that some accuracy parameters are available to the user of the black-box in the form of the vector ε. These parameters that define the accuracy of the numerical method used in the analysis problem may include the mesh density, the order of the finite elements, the number of iterations of a nonlinear method, etc.

Each individual in the parent population P_k has an associated accuracy, which can take any value within a list of discrete values provided by the designer within the range $[\varepsilon^-, \varepsilon^+]$, related to the maximum and minimum accuracies acceptable for the problem. Suppose there are two functions available: the function $next(\varepsilon)$, which returns the next value in the list, and $prev(\varepsilon)$, which returns the previous value in the list. Also, $next(\varepsilon^+) = \varepsilon^+$ and $prev(\varepsilon^-) = \varepsilon^-$.

In this way, the evolutionary algorithm can specify the values of ε. The searching process can then dedicate more computational effort when evaluating some solutions in the search space $\mathscr{S}$. The parameters ε may vary with time and/or within the population at a given generation. For example, as the time increases the parameters ε are selected in order to increase the accuracy of the numerical method. Moreover, in a given generation, we can dedicate more effort and time to those solutions in the most promising regions, and solutions generated by exploratory mutations are evaluated with less computational effort. These general ideas in the utilization of varying accuracy cost functions lead to the specification of the following general rules or *heuristics*:

1. Initially, the basis value ε assumes the first value in the list, i.e., ε^{-}.
2. During the first generations, the algorithm is in its exploration phase, sampling the search space in a more exploratory way and the diversity of the population is high. As the number of generations increases, we expect the population have converged toward a given optimum, and we can therefore increase the accuracy for the evaluation of the population. Therefore, after T_ε successive generations, we change the base value to $next(\varepsilon)$.
3. Individuals in better regions of the search space should be evaluated with higher accuracy, while we should not spend much computational effort evaluating individuals in poor regions. In this case, we use the fitness values to decide which individuals are evaluated with higher accuracy. The offspring of the N_ε best individuals are evaluated using $next(\varepsilon)$, while the remaining ones are evaluated using the current base value ε. This follows from the principle of heredity, which states that the offspring tend to be similar to their parents.
4. Each offspring individual changed by mutation is evaluated using $prev(\varepsilon)$, thus with less effort.

This approach does give rise to some noise in the objective function, but evolutionary algorithms are relatively robust in the presence of noise. Moreover, as the algorithm converges towards a given optimum, this noise is reduced because more and more points around this optimum are evaluated with the same accuracy.

7.2.2 Approximation-Based Local Search

Considering the class of expensive optimization problems, the accumulated information represented by the samples gathered by the algorithm is very valuable. We can progressively reconstruct the black-box input-output relationship to an arbitrarily high precision, which in turn may be exploited to speed up the searching process via local search operators.

By viewing the evolutionary process as an intelligent sampling process, we note that there will be more samples in the most promising regions of the search space. In these regions, the evolutionary convergence is slow, with poor precision due to genetic drift caused by the variation operators. As the generations advance, we obtain more samples in these regions, making the approximations more accurate. It is

reasonable to believe that the local approximations will improve as the search progresses, making the solutions achieved by indirect (i.e., using approximations) local search arbitrarily close to the ones that would be obtained by direct local search.

In the context of CAD optimization problems using $\mathscr{A}$-MAs, we point out some important requirements [11]:

- The time spent in the evaluation step is dominant in the whole process;
- The time spent to generate the approximations should be as small as possible;
- The total time spent in the optimization process when using the hybrid algorithm with approximation-based local search must be less than that spent when using the standard algorithm.

The first observation is often true in complex CAD problems. It implies that some additional complexity in the algorithm operations is justifiable. The second requirement is important because the time needed to generate and evaluate the approximations must be small in comparison to the time consumed in evaluating solutions directly. Finally, the third requirement means that the hybrid algorithm must converge with less evaluations than the standard algorithm, or at least provide a better solution for the same number of evaluations. We detail next how the approximations are generated and how the local search is performed. We consider constrained mono-objective problems only.

All evaluations performed by the evolutionary algorithm are stored in a global data set:

$$\mathscr{D}_k = \left\{ \mathbf{x}^{(i)}; f(\mathbf{x}^{(i)}); g_1(\mathbf{x}^{(i)}), \ldots, g_{n_g}(\mathbf{x}^{(i)}); h_1(\mathbf{x}^{(i)}), \ldots, h_{n_h}(\mathbf{x}^{(i)}) \right\}_{i=1}^{N_k} \tag{7.4}$$

where N_k is the total number of samples collected by the algorithm at iteration k. $\mathscr{D}_k$ contains each solution $\mathbf{x}^{(i)}$ in the search space tested by the algorithm and its respective values for the objective and constraint functions. Only nonlinear functions are stored. Linear constraints are explicitly provided by the user and are not approximated. Additionally, the value of the accuracy parameter used to evaluate $\mathbf{x}^{(i)}$ is also stored. $\mathscr{D}_k$ is the global data set representing all information acquired by the algorithm until time k. Part of this information, the data within the neighborhood of the individual, is used for building the local approximations, specifically the RBF approximation.

Define $\mathbf{z}$ as the solution represented by the individual $\mathbf{p}_k^{(i)} \in P_k$ selected for local search. The neighborhood region is centered at $\mathbf{z}$ and is generally defined as follows:

$$\mathscr{V}(\mathbf{z}, \rho) = \{ \mathbf{x} : \|\mathbf{x} - \mathbf{z}\| \leq R(\rho) \} \tag{7.5}$$

where $R(\rho)$ is a region whose size is parameterized by ρ. Considering $\|\cdot\|_\infty$, we have a rectangular neighborhood:

$$\mathscr{V}(\mathbf{z}, \rho) = \left\{ \mathbf{x} : z_i - \rho(x_i^+ - x_i^-) \leq x_i \leq z_i + \rho(x_i^+ - x_i^-),\ i = 1, \ldots, n_x \right\} \tag{7.6}$$

and considering $\|\cdot\|_2$, we have an ellipsoidal neighborhood

$$\mathscr{V}(\mathbf{z},\rho) = \left\{\mathbf{x} : (\mathbf{x}-\mathbf{z})^T \boldsymbol{\Delta} (\mathbf{x}-\mathbf{z}) \leq 1\right\} \tag{7.7}$$

in which the matrix $\boldsymbol{\Delta}$ is given by:

$$\Delta_{ij} = \begin{cases} [\rho(x_i^+ - x_i^-)]^{-1}, & i = j \\ 0, & i \neq j \end{cases} \tag{7.8}$$

The parameter $0 < \rho < 1$ defines the size of the local neighborhood with respect to the parameter range. This parameter is usually set to a small value, typically 0.1.

Algorithm 7.2 shows how the local data set $\mathscr{L}$ is assembled from $\mathscr{D}$. Identical points and points closer than a threshold ζ do not enter the local data set. The elimination of similar points is important, because they cause numerical ill-conditioning in approximation techniques such as neural networks and RBF models.

Algorithm 7.2. Building local data set

```
1  L = ∅ /* Stores local data set                                   */
2  for i = 1,...,N_k do
3      if x^(i) ∈ V(z,ρ) then
4          calculate the distance between x^(i) and each data point in L;
5          if x^(i) is at least an amount ζ distant from all points in L then
6              put x^(i) and their respective evaluations in L;
7          else
8              keep the point evaluated with the greater accuracy in L;
9          end
10     end
11 end
12 return L;
```

The approximations are generated to fit data in $\mathscr{L}$. We employ RBF approximations of the form:

$$p(\mathbf{x}) = \sum_{i=1}^{m} v_i r_i \left(\|\mathbf{x} - \mathbf{c}_i\|\right) = \mathbf{r}^T \mathbf{v} \tag{7.9}$$

where $\mathbf{x}$ is the vector of optimization variables, $\mathbf{c}_i$ is the center of the radial basis function $r_i\left(\|\mathbf{x} - \mathbf{c}_i\|\right) : \mathbb{R}^{n_x} \mapsto \mathbb{R}$, and $\mathbf{v}$ is the vector of parameters of the RBF model. There are many types of RBF available [13, 30], we have selected the multiquadric function in the methodology. For training the RBF approximation, we adopt the following error cost function:

$$C(\mathbf{v}) = \sum_{i=1} w_{ii} e_i^2 = \mathbf{e}^T \mathbf{W} \mathbf{e} \tag{7.10}$$

where e_i is the error between the RBF model and the desired value in the local data set $\mathscr{L}$. The weighted squared error is used because some points in the data are evaluated with more accuracy than others. Therefore, these points should have

a greater weight in the RBF model. This problem can be solved with the Weighted Least Squares (WLS) method:

$$\hat{\mathbf{v}} = \left[\mathbf{R}^T\mathbf{W}\mathbf{R}\right]^{-1}\mathbf{R}^T\mathbf{W}\mathbf{y} \tag{7.11}$$

where $\mathbf{R}$ is the matrix with the values of the radial basis functions and $\mathbf{y}$ is the vector with the desired outputs.

Each nonlinear expensive-to-evaluate function in the optimization problem is locally approximated using the RBF model trained with the WLS method. One advantage of this approach is that it is a fast way to obtain the approximations, as required by the observations made before. With the approximations in hand, we can define the local search problem as:

$$\begin{aligned} &\min_{\mathbf{x}} \tilde{f}(\mathbf{x}) \\ &\text{subject to: } \mathbf{x} \in \widetilde{\mathscr{F}_x} \cap \mathscr{V}(\mathbf{z},\rho) \end{aligned} \tag{7.12}$$

in which $\mathscr{V}(\mathbf{z},\rho)$ is the local neighborhood, $\tilde{f}(\cdot)$ is the approximation for the objective function, and $\widetilde{\mathscr{F}_x}$ is the approximated feasible set generated by the approximated constraints.

Observe that the local search problem, see (7.12), has an additional constraint in comparison to the original optimization problem: the local problem is restricted to the region $\mathscr{V}(\mathbf{z},\rho)$, that is, the neighborhood of the individual selected for local search. This problem can be easily and directly solved by employing the Sequential Quadratic Programming (SQP) method [8]. The SQP method is an extension of the Newton's method for unconstrained optimization to constrained problems. The method replaces the objective function with its quadratic approximation and replaces the constraint functions by linear approximations. A nonlinear problem in which the objective function is quadratic and the constraints are linear is called a Quadratic Program (QP). The SQP method solves a QP at each iteration until a satisfactory solution is found. The local convergence properties of the SQP are well understood. Therefore, although the solution of the problem (7.12) is obtained numerically, it is found by a fast and accurate method, allowing us to call it a *semi-analitycal solution*. Furthermore, since the expressions of the RBF approximations are known, their quadratic and linear models are easily and analytically obtained and used within the SQP procedure. We summarize the local search operator in Algorithm 7.3.

Algorithm 7.3. Approximation-based local search operator

1 $\mathscr{L} \leftarrow$ `Build local data(`$\mathscr{V}(\mathbf{z},\rho)$`);`
2 $\tilde{f}, \tilde{\mathbf{g}}, \tilde{\mathbf{h}} \leftarrow$ `Generate local approximations(`$\mathscr{L}$`);`
3 $\mathbf{z}^* = \arg\min \tilde{f}(\mathbf{x})$ subject to: $\mathbf{x} \in \widetilde{\mathscr{F}_x} \cap \mathscr{V}(\mathbf{z},\rho)$ `/* Enhanced solution */`
4 **return** $\mathbf{z}^*$;

This section presented the approximation-based local search operator and its use in evolutionary algorithms. The local enhancement of some individuals in $\mathscr{A}$-MAs can be achieved in an indirect way, by using a local representation of the problem, based on the knowledge acquired by the algorithm along successive generations. This indirect local search via local approximations reduces the computational cost associated with the exploitation component of memetic algorithms. However, it is important to analyze the theoretical effects of the proposed local search on the global convergence properties of evolutionary algorithms. This issue is addressed in the next section. We also discuss the computional cost of the proposed methodology.

7.3 Analysis of Memetic Algorithms

In the previous section we discussed $\mathscr{A}$-MAs in general and presented the approximation-based local search methodology, which can be coupled with usual evolutionary algorithms following the generic structure in Algorithm 7.1. In this section, we employ the Markov chain theory, which is a popular theoretical tool for analyzing EAs [1, 32], to analyze the effect of the local search operators on global convergence properties of memetic algorithms. We also present the computational cost analysis of memetic algorithms, especially those employing the local search methodology in the context of CAD optimization problems.

7.3.1 Convergence Analysis

Let $\mathscr{X}$ be a finite set of states with cardinality $|\mathscr{X}|$ and $\{\mathsf{X}_n \in \mathscr{X} : n \in \mathbb{N}\}$ a random sequence or stochastic process. X_n is a Markov chain if it is a stochastic process in which the future state depends only on the present state, and it is independent of the past states. Observe that the sequence of populations of an evolutionary algorithm falls in this definition, since the transitions from one population to another is stochastic and independent of previous populations[3]. We can write transition probabilities as $\mathscr{P}\{\mathsf{X}_{n+1} = \mathsf{s}_j | \mathsf{X}_n = \mathsf{s}_i\} = \tau_{ij}$, with $i, j \in \{1, \ldots, |\mathscr{X}|\}$ and $\mathsf{s}_i, \mathsf{s}_j \in \mathscr{X}$, where $|\mathscr{X}|$ is the number of states. Since the state space is finite, we can conveniently represent the transition probabilities in a $|\mathscr{X}| \times |\mathscr{X}|$ matrix $\mathbf{T}$. Each entry of the stochastic transition matrix $\mathbf{T}$ gives the probability that the next state is s_j given that the current state is s_i.

The state of an algorithm in the time step n is represented by the population at time n. The operations in the algorithm define its transition matrix $\mathbf{T}$. The state space $\mathscr{X}$ is the set of all possible populations with size μ for the search space $\mathscr{S}$.

The generic structure presented in Algorithm 7.1 has an archive population A_n, with size $\xi \geq 1$. Considering this population, the state representation changes to

[3] In general, adaptive evolutionary algorithms cannot be analyzed by Markov chain theory because their transition matrices are time-dependent. In these situations, we need to reccur to other analysis methods. Adaptive memetic algorithms are analyzed with Markov chains in [29] but assuming that the transition matrix produced by the genetic operators is positive.

$\mathsf{s}_i = \left(\mathsf{s}_i^A; \mathsf{s}_i^P\right)$, where s_i^A is the part of the state associated with A_n, and s_i^P is the part associated with P_n. After the archive update function, we can say that the individuals represented by s_i^A are the best individuals in the state s_i.

Before proceeding, we give some useful definitions:

Definition 1. *A state s_j is said to be accesible from the state s_i if $\exists k < \infty$ such that: $\mathscr{P}\{\mathsf{X}_{n+k} = \mathsf{s}_j | \mathsf{X}_n = \mathsf{s}_i\} = \tau_{ij}^{(k)} > 0$, where $\tau_{ij}^{(k)}$ is the element ij of the matrix $\mathbf{T}^k$, and we write $\mathsf{s}_i \rightarrow \mathsf{s}_j$.*

Definition 2. *A state s_j is said to be a communicating state with the state s_i if $\mathsf{s}_i \rightarrow \mathsf{s}_j$ and $\mathsf{s}_j \rightarrow \mathsf{s}_i$, and we write $\mathsf{s}_i \leftrightarrow \mathsf{s}_j$.*

Definition 3. *A Markov chain $\{\mathsf{X}_n \in \mathscr{X} : n \in \mathbb{N}\}$ is irreducible if its state space is a communicating class, that is, all its states are communicating states.*

The last definition means that it is possible to get to any state from any state in an irreducible Markov chain. Therefore, every state will be visited in finite time regardless of the initial state [26]. Due to the use of the archive population A_n, some states are not communicating states, because of elitism. Thus, states containing worse solutions in A_n cannot be reached from states containing better solutions in A_n. These intermediate states in the search process are called transient states, since $\mathsf{s}_i \rightarrow \mathsf{s}_j$, but $\mathsf{s}_j \not\rightarrow \mathsf{s}_i$.

Let $\mathscr{S}^*$ be the set of optimal solutions for the optimization problem. It can represent: (i) all the global optima if there are more than one; (ii) all the global and local optima (if the algorithm is designed to find them); or (iii) all global Pareto-optimal solutions in a multi-objective context. We adopt the following notation:

Definition 4. *Those states that represent an A_n whose elements belong to $\mathscr{S}^*$ are called essential states. They form the set $\mathscr{E}$. Those states that represent an A_n whose elements do not belong to $\mathscr{S}^*$ are called inessential states, hence they are intermediate states. They form the set $\mathscr{I}$.*

From [33], we know that $\mathscr{P}\{\mathsf{X}_n \in \mathscr{I}\} \rightarrow 0$, when $n \rightarrow \infty$. Since all essential states represent archive populations that are optimal, whose meaning depends on the context, the algorithm will finally converge to one of the essential states:

$$\mathscr{P}\{\mathsf{X}_n^A \subseteq \mathscr{X}^*\} = \mathscr{P}\{\mathsf{X}_n \in \mathscr{E}\} = 1 - \mathscr{P}\{\mathsf{X}_n \in \mathscr{I}\} \tag{7.13}$$

which becomes equal to one, when $n \rightarrow \infty$, and we say that the algorithm is globally convergent. The probability that the archive population A_n has converged to a subset of the optimal solution set is equal to one when time goes to infinity. Therefore, we say that the online population P_n *locates* the optimal solution, while the offline population A_n *converges* to the optimal solution.

All essential states should be accessible from any inessential state. So, to improve the solutions in $A(n)$, the following transition must be possible:

$$\left(\mathsf{s}_i^A; \mathsf{s}_i^P\right) \rightarrow \left(\mathsf{s}_i^A; \mathsf{s}_j^P\right) \rightarrow \left(\mathsf{s}_j^A; \mathsf{s}_j^P\right) \tag{7.14}$$

where s_j^P represents a population with better solutions than those in s_i^A.

Observe that the archive population does not undergo the selection and variation steps, and the last transition is performed by the update operator. Therefore, s_j^P must be accessible from s_i^P, in order to validate the complete transition. Thus, although the complete transition matrix is not irreducible, the transition matrix associated with the online population P_n must be irreducible, in order to guarantee that the online population will visit all states in the search space with finite time.

Let $\mathbf{G}$ be the transition matrix for P_n. It is a function of the operators of the algorithm during one iteration, i.e., it is obtained by the product of transition matrices associated with the selection and variation steps (crossover and mutation in the case of genetic algorithms, for instance). Since we will analyze the product of stochastic matrices, it is important to provide some helpful definitions and properties.

Definition 5. *A matrix* $\mathbf{A}$ *is said to be positive*[4] *if* $a_{ij} > 0$, $\forall i, j$.

Definition 6. *A matrix* $\mathbf{A}$ *is said to be non-negative if* $a_{ij} \geq 0$, $\forall i, j$.

Definition 7. *A square matrix* $\mathbf{A}$ *is said to be diagonal positive if all elements of its diagonal are positive.*

Definition 8. *A stochastic square matrix* $\mathbf{A}$*, representing the transition probabilities of a Markov chain process with state space* $\mathscr{X}$*, is said to be irreducible if* $\forall \mathsf{s}_i, \mathsf{s}_j \in \mathscr{X}$*,* $\exists n \in \mathbb{N}$ *such that* $a_{ij}^{(n)} > 0$.

A positive matrix $\mathbf{A}$ is hence irreducible since it satisfies $a_{ij}^{(n)} > 0$, with $n = 1$. The converse is not necessarily true.

Definition 9. *A matrix is said to be column-allowable (row-allowable) if each column (row) contains at least one positive entry. A matrix that is both column-allowable and row-allowable is an allowable matrix.*

From the definition above, we see that a diagonal positive matrix is both column-allowable and row-allowable. Also, the product of two diagonal positive matrices is also a diagonal positive matrix.

Let us consider a standard genetic algorithm, using selection, crossover, and mutation operators in the evolutionary iteration. In this case, the transition matrix $\mathbf{G}$ is:

$$\mathbf{G} = \mathbf{SCM} \tag{7.15}$$

where $\mathbf{S}$, $\mathbf{C}$, and $\mathbf{M}$ are respectively the transition matrices of selection, crossover, and mutation steps.

We know from [32] that this transition matrix is positive if the mutation operator produces a positive transition matrix. There are mutation operators whose transition matrices are not positive, but they are irreducible. Since there is a positive probability that the selection and crossover operators do not change the individual, then there is a positive probability that any point in the search space be attained in $n > 1$ steps, by consecutive mutations. Hence $\exists n > 0$ such that $\tau_{ij}^{(n)} > 0$ and the transition matrix is irreducible. In this case, it is possible to show that if $\mathbf{M}$ is irreducible and the product $\mathbf{SC}$ is diagonal positive, then $\mathbf{G}$ is also an irreducible matrix [1].

[4] Do not confuse with the concept of positive (or negative) definiteness.

7.3.1.1 Analyzing the Local Search Operator

In this section, we analyze the case when a local search is explicitly used in the evolutionary cycle. In the canonical hybrid algorithm, see Algorithm 7.1, the local search is performed before the selection and variation steps. Thus, assuming that the product **SCM** is irreducible, we need to analyze what happens with the product:

$$\mathbf{H} = \mathbf{LG} = \mathbf{L}(\mathbf{SCM}) \tag{7.16}$$

where **L** is the transition matrix associated with the local search phase, **G** is the transition matrix associated with the global search algorithm, and **H** is the transition matrix associated with the hybrid algorihm, which results from the global-local search interaction.

Considering the Lamarckian approach for hybrid evolutionary algorithms, the population is modified by the local search. If the local search operator is deterministic, we can say that **L** is at least row-allowable. Based on these characteristics, we can state the following:

Theorem 1. *If* **SCM** *is positive and the local search is deterministic,* **L** *is at least row-allowable, then* **H** *is also positive and thus irreducible. Therefore, the archive population will converge to the solution set and the hybrid algorithm is globally convergent.*

Proof. Let $\mathbf{D} = \mathbf{SCM}$. Since **L** is row-allowable, there is at least one k such that l_{ik} is positive, then:

$$h_{ij} = \sum_k l_{ik} d_{kj} > 0$$

$\forall i, j$. Thus, **H** is positive and, following the Definition 8, it is irreducible. □

However, when $\mathbf{G} = \mathbf{SCM}$ is irreducible, due to a mutation operator that produces a non-positive but irreducible transition matrix, and **L** is row-allowable, we cannot state that **H** is irreducible. Therefore, we cannot prove global convergence in general. This situation can be understood with an illustrative example, see Figure 7.2. In this Figure, we consider that the population is concentrated in the region of attraction of a local optimum, which is not the global optimum. Using a mutation with compact support, the population is able to escape from the local minimum in $n > 1$ steps. But if the local search is applied to all individuals at every generation, the population will never escape from the local minimum in this situation.

Nevertheless, it is possible to guarantee convergence if the number of individuals selected for local search σ is smaller than the population size μ. The result is stated in the following theorem:

Theorem 2. *If* $\sigma < \mu$, **SCM** *is irreducible and* **L** *is at least row-allowable, then* **H** *is also irreducible (but not positive in general). Therefore, the archive population will converge to the solution set and the hybrid algorithm is globally convergent.*

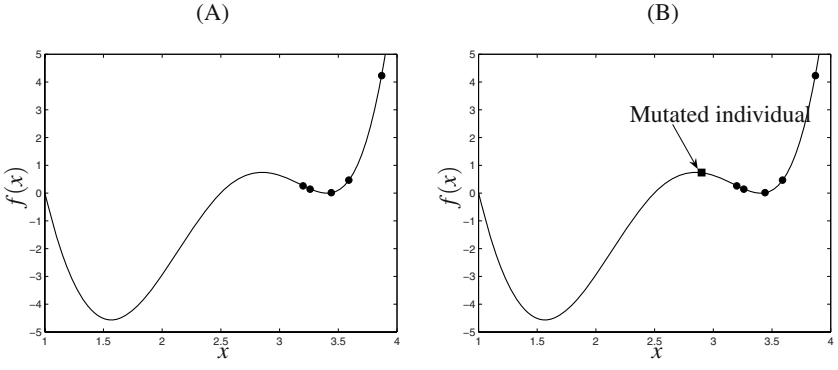


Fig. 7.2 (A) The population of an evolutionary algorithm at a given iteration is stuck in the region of attraction of a local minimum. (B) A mutation operator with compact support produces new solutions inside the same region of attraction, as shown in the figure. In the next evolutionary cycle, the local search will concentrate the population, including the mutated individuals, around the local minimum. The population has no chance of escaping the local minimum in this situation

Proof. In this case, there is a positive probability that the individual(s) not selected for local search is(are) able to escape from the local minimum in $n > 1$ steps due to the irreducible mutation. Therefore, $\exists n$ such that $h_{ij}^{(n)} > 0$ and hence the product $\mathbf{H} = \mathbf{L}(\mathbf{SCM})$ will be irreducible when $\sigma < \mu$. □

Finally, we need to analyze the situation when the local search operator is not applied at every generation, but at every constant number of generations, let us say, at every n_L generations. The transition matrix becomes:

$$\mathbf{H} = \mathbf{L}\underbrace{\mathbf{SCM} \times \mathbf{SCM} \times \cdots \times \mathbf{SCM}}_{n_L \text{ times}} \tag{7.17}$$

then:

$$\mathbf{H}^n = (\mathbf{L}\mathbf{G}^{n_L})^n \tag{7.18}$$

For this situation, we have the following result:

Theorem 3. *If $\sigma < \mu$, $\mathbf{G} = \mathbf{SCM}$ is irreducible, $\mathbf{L}$ is at least row-allowable, and the local search is applied at every n_L generations, then $\mathbf{H}$ is also irreducible (but not positive in general). Therefore, the archive population will converge to the solution set and the hybrid algorithm is still globally convergent.*

Proof. The hybrid algorithm using local search at every n_L generations can be seen as a hybrid algorithm with an extended generation, that is, one in which the local search is applied first and then the selection, crossover and mutation operators are applied n_L times. Since $\mathbf{G}$ is irreducible, the sequence of n_L iterations of $\mathbf{G}$ is also irreducible, i.e., $\mathbf{G}^{n_L}$ is irreducible. Consequently, $\mathbf{H} = \mathbf{L}\mathbf{G}^{n_L}$ remains irreducible from Theorem 2. □

Therefore, given the considerations above, the hybrid algorithm using an explicit local search phase is also globally convergent as long as the non-hybrid algorithm is globally convergent. The local search phase will not affect this property except when the mutation is irreducible and the local search is applied to all individuals.

This global convergence analysis by means of Markov chain theory allows us to state only if the algorithm is globally convergent or not under the very general criterion considered previously. The analysis does not say anything about the convergence rate of the algorithm. For example, the random search algorithm, using uniform sampling in the search space, is globally convergent under the Markov chain analysis, because its transition matrix is positive. As long as we use an archive population to store the best solutions, the random search is globally convergent. Nonetheless, assessing the global convergence property of an algorithm is an obvious requirement to calculate its convergence rate. Moreover, strategies like the random search and simple enumeration are very inefficient in practice. The transition matrices **L**, **S**, and **C** do not affect the global convergence of the algorithm, which stands on the transition matrix **M**. This is important, but they may affect its convergence time. As a consequence of the *No Free Lunch* theorems [40], an evolutionary algorithm can outperfom the random search for a given class of problems, when the local search and crossover operators exploit some knowledge of that class of problems, or implicitly rely on some common structure of these problems. On the other hand, simple random search does not exploit any structure at all. Evolutionary algorithms are far from being random search methods, because the matrices **L**, **S**, and **C** can improve the performance of the algorithm in comparison to simple random search for a specific class of problems.

7.3.2 Computational Cost

The MA has the potential of converging in less generations, because reaching any of the basins around the global optima gives more chance of achieving the optimal solutions through local search. Of course, that comes at the price of a more expensive generation. Consequently, even though the hybrid algorithm converges in less iterations, on average, it is not useful in practice if it takes more time than the conventional one does. For practical purposes, we require that the total optimization time, on average, by using the hybrid algorithm be smaller than the total time, on average, needed when using the non-hybrid algorithm. We discuss this relation in this section.

Before proceeding, we turn to computational complexity issues first. The quantity $\langle N \rangle$ is hereafter termed the averaged number of generations. We introduce $\langle T \rangle$ as the averaged total optimization time, which is of more practical interest. We can estimate the time complexity of an evolutionary algorithm to converge in a given problem as:

$$O(\mu, \lambda, \xi, n_x)\langle N \rangle = [O_r(\mu, \lambda, \xi, n_x) + O_e(\varepsilon)]\,\langle N \rangle \tag{7.19}$$

where $O_r(\cdot)$ is the number of operations in the reproduction step and $O_e(\cdot)$ is the number of operations in the evaluation step. They are related to parameters of the algorithm and the number of variables n_x. The complexity of the reproduction operators of EAs is polynomial.

The number of solutions generated by Algorithm 7.1 at generation k is $\mu + \lambda k$. The operation of building the local data has a worst case complexity of $O((\mu + \lambda k) N_L n_x)$, where N_L is the maximum number of points allowed in the local data set $\mathscr{L}$. The cost of the generation of each approximation by the multiquadric RBF technique is dominated by the assembling of the $\mathbf{R}$ matrix, which is of $O(n_x N_L m)$ because we need to calculate the pairwise distances of N_L points and m centers. Of course, many local approximations are generated with less than N_L points. The computation of the pseudo-inverse has complexity of $O(m^2)$. Finally, considering all objective and constraint functions, we get $O((1+n_g+n_h)(n_x N_L m + m^2))$.

Finally, the cost of the solution of the auxiliary problem by the SQP method can be roughly estimated as follows. The cost of evaluating the multiquadric model is of $O(n_x m)$. The computation of the gradients of all functions in the auxiliary problem requires $(1+n_g+n_h)$ evaluations of the multiquadric approximations, since the gradients can be obtained inexpensively together with the evaluations of the approximations. Therefore, we can estimate a complexity of $O((1+n_g+n_h)n_x^2 m)$ for solving the local search problem.

We can see that the complexity of the approximation-based local search is small and the polynomial complexity of usual EAs is preserved in the $\mathscr{A}$-MA. The approximation-based local search adds the operations associated with local data assembling and approximation building, which have polynomial complexity, and increase storage requirements, but this increase in storage is linear. The computational complexity of the evaluation step depends on the calculations made by the black-box function. The overall cost of this step is reduced in our framework by evaluating some individuals with low accuracy and thus less computational effort.

The complexity analysis is important but limited because the complexity of the evaluation step is unknown. The overhead of the local search can be quantified as follows:

$$\langle T_G \rangle = (\mu t_e + t_r)\langle N_G \rangle \tag{7.20}$$

$$\langle T_H \rangle = \left(\mu t_e + t_r + \frac{\sigma}{n_L} t_{ls}\right)\langle N_H \rangle \tag{7.21}$$

where $\langle T_G \rangle$ and $\langle T_H \rangle$ are, respectively, the averaged total time consumed by the standard EA and the MA to converge to the solution within a given accuracy; $\langle N_G \rangle$ and $\langle N_H \rangle$ are, respectively, the averaged number of generations required by the standard EA and the MA to converge[5]; t_e is the time consumed to evaluate one solution; t_r is the time consumed by the reproduction step; t_{ls} is the computational time consumed by the local search of one individual.

[5] $\langle N_G \rangle$ and $\langle N_H \rangle$ can represent the theoretical mean time in the Markov chain or the sample mean over a given number of independent runs.

By imposing the condition $\langle T_H \rangle < \langle T_G \rangle$ and assuming $t_e >> t_r$, which is usually the case in CAD problems, we have

$$\langle N_H \rangle < \langle N_G \rangle \frac{\mu t_e}{\left(\mu t_e + \frac{\sigma}{n_L} t_{ls}\right)} = \langle N_G \rangle \frac{1}{\left(1 + \frac{\sigma}{\mu n_L} \frac{t_{ls}}{t_e}\right)} \tag{7.22}$$

Based on this relation, we observe that even if the hybrid algorithm converges in less generations (on average) compared to the basic algorithm, $\langle T_H \rangle$ can be greater. The more significant the amount $(\sigma/n_L)t_{ls}$, compared to μt_e, the smaller the value $\langle N_H \rangle$ should be in comparison to $\langle N_G \rangle$ to satisfy the initial condition $\langle T_H \rangle < \langle T_G \rangle$.

When using an indirect local search, by means of approximated functions, we can have $t_{ls} << t_e$, that is, the time to evaluate a solution is dominant in the problem. In this case, if $\langle N_H \rangle$ is slightly smaller than $\langle N_G \rangle$, then the initial condition will be satisfied and thus the additional complexity introduced by the methodology would be justifiable. This relation emphasizes the context of problems in which the proposed methodology can be considered helpful.

This section showed that as long as we can say that the EA is globally convergent under Markov chain analysis, the local search operator preserves this characteristic. Moreover, the approximation-based local search operator preserves the polynomial complexity of standard evolutionary algorithms, and the additional complexity in the memetic algorithm is not dramatic and would be acceptable in some applications. The final relations developed show under which conditions the hybrid algorithm is advantageous. These relations show that the proposed methodology can be very interesting for expensive optimization problems, in which the evaluation time t_e of a single solution is very big in comparison to the time required for performing the normal operations of the algorithm, including the local search. This scenario, commonly found in CAD problems, is the one in which the additional complexity of the memetic algorithm is justifiable.

7.4 Numerical Results

7.4.1 Analytical Problems

In this first experiment we apply a typical genetic algorithm configuration, from the framework discussed in Section 2, to analytical unconstrained optimization problems. The goal here is to observe the effect of the local search on the convergence speed of the evolutionary algorithm in terms of the number of generations. For this purpose, we consider the minimization of the following functions:

$$f_1(\mathbf{x}) = \sum_{i=1}^{n_x - 1} 100\left(x_i^2 - x_{i+1}\right)^2 + (1 - x_i)^2, \quad -2 \le x_k \le 2, \forall k \tag{7.23}$$

$$f_2(\mathbf{x}) = 2.6164 + \frac{1}{n_x} \sum_{i=1}^{n_x} 0.01\left[(x_i + 0.5)^4 - 30x_i^2 - 20x_i\right], \quad -6 \le x_k \le 6, \forall k \tag{7.24}$$

$$f_3(\mathbf{x}) = 10n_x + \sum_{i=1}^{n_x} \left[x_i^2 - 10\cos(2\pi x_i)\right], \quad -5 \leq x_k \leq 5, \forall k \tag{7.25}$$

The function f_1 is the Rosenbrock function, which is a unimodal and non-convex function. Despite the apparent simplicity of this function, it is notoriously hard for evolutionary algorithms in general. The function f_2 is a multimodal function with a moderate degree of multimodality, with local minima located at the corners of the hypercube $C = [\pm 4.4538 \ldots \pm 4.4538]$. The function f_3 is the Rastrigin function, with a high degree of multimodality. The Rastrigin function is another difficult problem as it defines a search space with many local optima, due to the combination of a sinusoidal and a quadratic function. All problems were solved with $n_x = 4$.

Both the GA and the $\mathscr{A}$-MA shared the following setup: $\mu = \lambda = 50$ and $\xi = 1$; ES(μ, μ) for the substitution; roulette wheel as selection operator; real-biased convex crossover; Gaussian mutation with $\rho_m = 0.1$. The only stop criterion is the maximum number of generations, set as 200. For the $\mathscr{A}$-MA, the maximum number of points used to build the local approximation is $N_L = 200$. The local search was applied to the best individual in the offspring at every generation, hence $\sigma = 1$ and $n_L = 1$.

Due to the stochastic behavior of both algorithms, we need to perform a given number of runs to draw meaningful conclusions. Therefore, each algorithm was executed 40 times on each problem. Figure 7.3 shows the mean convergence of the GA and the $\mathscr{A}$-MA on each problem. As we can see, the memetic algorithm converged faster than the typical GA on these problems. These results confirm the hypothesis that the local search speeds up the convergence of the genetic algorithm.

Table 7.1 shows the success rate of both algorithms on all problems, as well as an estimate of the average number of generations to converge $\langle N \rangle$. The maximum number 200 is used for those runs in which the algorithm has not converged. This Table shows that the local search not only has improved the convergence speed of the algorithm but also it has increased the success rate.

The smallest difference in $\langle N \rangle$ is observed in the Rastrigin function, while the biggest difference appears in the minimization of f_2. The Rastrigin function is highly multimodal, therefore one would expect that the local search would not present a great impact on the performance. The observed effect on the performance may be understood by noting that the Rastrigin function is in fact a quadratic function plus noise. The local approximation is not capable of modeling this noise, due to undersampling, specially in the first generations, but can capture the global trend of the quadratic component in the objective function. The mean curves in Figure 7.3 clearly show a fast decrease in the first 15 generations. After that, the slopes of the mean convergence velocity curves for both algorithms are similar. After some generations, the noise component was also "learned" during the evolutionary process, then reducing the impact of the local search because the basins of attraction become small. That is an interesting side effect of the approximation-based local search: it can filter a highly multimodal function, capturing the global trend in the objective function.

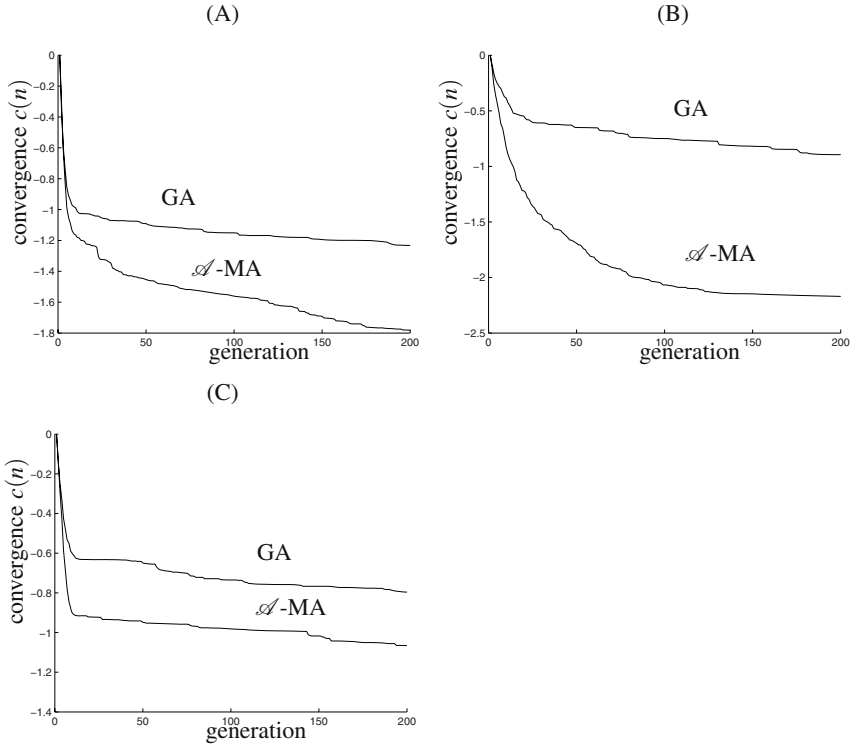


Fig. 7.3 Mean convergence for the GA and its memetic version for (A) $f_1(\mathbf{x})$, (B) $f_2(\mathbf{x})$, and (C) $f_3(\mathbf{x})$

Table 7.1 Results for the analytical functions

Problem	Algorithm	Success rate	$\langle N \rangle$
$f_1(\mathbf{x})$	GA	22.5%	171.9
	$\mathscr{A}$-MA	67.5%	105.0
$f_2(\mathbf{x})$	GA	67.5%	106.8
	$\mathscr{A}$-MA	95.0%	21.5
$f_3(\mathbf{x})$	GA	7.5%	186.9
	$\mathscr{A}$-MA	20.0%	166.5

The mean velocities for f_1 and f_2 present different slopes for a wider range of generations, showing that the local search is having an important impact in the search process. This is because f_1 is unimodal and f_2, although multimodal, has a low to medium degree of multimodality. The basins of attractions in f_2 are fairly convex and present no obstacle for a gradient-based local search. This is why the performance in f_2 is even better than in f_1. The non-convexity of f_1 poses some difficulty for the approximation, due to the gradual slope of the "banana" region, and also for the local search.

In these analytical problems, the local search was the most time-consuming step. The overhead of the local search is in this case very high, because the evaluation of the objective functions is very fast. Therefore, in these problems, although the memetic algorithms converged in less iterations, they took much more time to do so. Nonetheless, we can estimate the minimal value for t_e that would make the memetic algorithms converge in less time using the relation in (7.22). For each analytical problem, we have monitored the mean times in seconds taken to perform the local search of one individual t_{ls} and we can use the values in Table 7.1 as rough estimates of $\langle N_H \rangle$ and $\langle N_G \rangle$. Using these values in (7.22), we get $t_{e,\min} = 71ms$ for $f_1(\mathbf{x})$, $t_{e,\min} = 12ms$ for $f_1(\mathbf{x})$, and $t_{e,\min} = 310ms$ for $f_1(\mathbf{x})$. Therefore, if the time to evaluate f_1, f_2, and f_3 was at least some tenths of a second, the memetic algorithm would have converged in less time than the genetic algorithm (on average), showing that the overhead of the local search is not that significant in expensive optimization problems.

7.4.2 Electromagnetic Benchmark Problem

The TEAM Workshop problem 22 was proposed in [3] and is known as a benchmark optimization problem in electromagnetics. The problem consists in optimizing the dimensions of a super-conducting magnetic energy storage (SMES) device with two solenoids. Such a device has many relevant applications and an optimal configuration for it is an important issue to resolve. This problem has three distinct objectives:

- The stray field, evaluated $10m$ away from the device, must be as small as possible;
- The stored energy must be equal to 180 MJ (50 kWh);
- The physical condition that guarantees the superconductivity, named the quench condition, must not be violated.

Figure 7.4 shows the geometry of the problem for the computational model. The stray field is evaluated at 21 points along the lines $(y = 10m; 0 \leq z \leq 10m)$ and $(z = 10m; 0 \leq y \leq 10m)$. This radiated field by the SMES may be formulated as the summation of the contributions of each of the two loops, given by the numeric solution of integrals derived from the Biot-Savart law. We may evaluate the energy through the Finite Element Method (FEM), using a magneto-static formulation with magnetic vector potential $\mathbf{A}$, considering the axis-symmetry to simplify it into a 2D problem, as shown in Figure 7.4.

The full-version of the problem has eight optimization variables: the current density in both coils and the coil shapes, defined by the radius, height and width of their 2D cuts. Nonetheless, a simplified version with only three variables can also be defined, in which the dimensions of the outer coil are optimized. The variables $r_1 = 2.0m$, $h_1 = 1.6m$, $d_1 = 0.27m$ are respectively the radius, the height and the width of the inner coil. The variables $2.6m \leq r_2 \leq 3.4m$, $0.4m \leq h_2 \leq 2.2m$, $0.1m \leq d_2 \leq 0.4m$ represent the same for the outer coil. J_1 and J_2 are respectively the current density in the inner and outer coil, both equal to $22.5MA/m^2$.

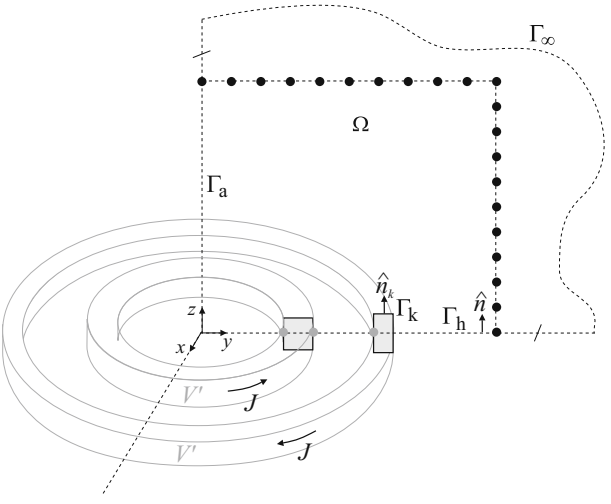


Fig. 7.4 Finite Element Method (FEM) model of the SMES device

The specifications above are translated into the optimization problem[6]:

$$\begin{aligned} &\min f(\mathbf{x}) = \sqrt{\tfrac{1}{21}\textstyle\sum_{i=1}^{21} \left|B_{stray_i}\right|^2} \\ &\text{subject to: } \begin{cases} \mathbf{x} \in \mathscr{S} \\ g(\mathbf{x}) = B_{\max} - 4.92T \leq 0 \\ h(\mathbf{x}) = |E - 180MJ|/180MJ = 0 \end{cases} \end{aligned} \tag{7.26}$$

where B_{stray_i} is the value for the magnetic flux density evaluated in one of the 21 points, $B_{\max}$ is the biggest value for the flux density and is related to the quench condition, it cannot be greater than the value of $4.92T$, and E is the energy value obtained by the FEM.

The accuracy parameter is the mesh density, and thus the number of nodes in the mesh, for the FEM. Fig. 7.5 shows the convergence of the energy value with the number of nodes. The list of values for the accuracy parameter is $\varepsilon = \{1.0, 0.5, 0.2, 0.1\}$. Table 7.2 shows how the number of nodes and the computation time vary with ε.

In order to compare instances of EAs and MAs in this problem, we use the mean convergence given by

$$c(n) = \log\left(\sqrt{\frac{f(\mathbf{x} \in A_k)}{f(\mathbf{x} \in A_1)}}\right)$$

We applied two instances of the generic structure in Algorithm 7.1 with and without the approximation-based local search. The parameters of the local search are $\sigma = 5$, $n_L = 4$, $N_L = 400$. One instance is a Genetic Algorithm (GA) with

[6] The square root is not present in the original formulation [3]. We use it here just to make the objective function smoother.

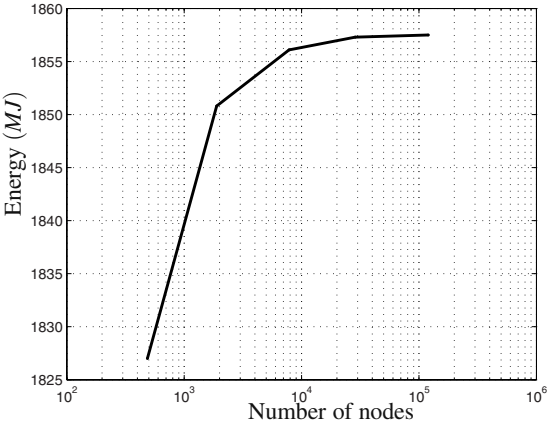


Fig. 7.5 Convergence with respect to the mesh density (energy value)

Table 7.2 Varying the mesh density

ε	nodes	$t_e(s)^a$	$\bar{t}_e$
1.0	1024	0.7	1
0.5	3777	1.0	≈ 1.5
0.2	22998	3.6	≈ 5
0.1	91246	16.0	≈ 23

[a] Running on a Windows computer, Intel Core 2 Duo, 2 GB RAM.

real-biased crossover [35], Gaussian mutation, and mutation rate of 0.1. The other is a Differential Evolution Algorithm (DEA) [34], with convex crossover for the mutant vectors, Gaussian mutation, and mutation rate of 0.1. In Fig. 7.6 we illustrate the mean convergence of GA, DEA, and their memetic versions. One can see that DEA performed better than the GA in this problem. Also, both algorithms presented higher convergence rate when using the approximation-based local search.

In the experiments in Fig. 7.6, all evaluations were performed with the same accuracy for all individuals ($\varepsilon = 0.2$ and $t_e = 3.6s$). Also, t_{ls} was about $1.4s$ on average and t_r was less than $1ms$ on average, making $t_e >> t_r$. Using these values in (7.22), we get $\langle N_H \rangle < 0.988 \langle N_G \rangle$, showing a negligible overhead for the local search. Therefore, if the hybrid algorithm takes less generations, it will take less time to converge in this problem. Using direct local search, i.e. without approximations, the time to perform the local search of one individual t_{ls} becomes two times greater than the time to evaluate the whole population in this problem, greatly increasing the overhead of the local search phase. Although the MA with direct local search usually converges in less generations than the $\mathscr{A}$-MA, its local search is very expensive, increasing the total time for the optimization process.

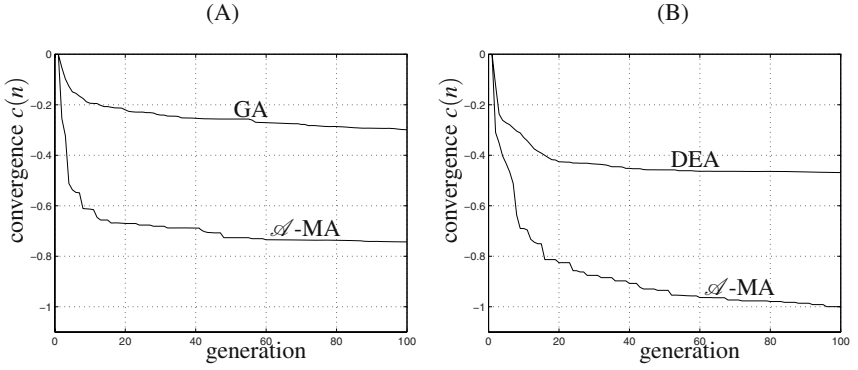


Fig. 7.6 (A) Mean convergence for the GA and its memetic version. (B) Mean convergence for the DEA and its memetic version

Finally, we introduce the varying accuracy approach with the parameters $T_\varepsilon = 20$ and $N_\varepsilon = 40\%$. The mean convergence curves and the mean number of evaluations were similar, but the overall optimization time was greater when using fixed accuracy. By comparing the memetic version of DEA using fixed accuracy ($\varepsilon = 0.1$) all the way and using the varying accuracy approach, we observe that the latter takes only 9% of the computational time consumed by the former to achieve the same level of solution quality – the level $c(n) = -1$ in the graph. Using the list $\varepsilon = \{1.0, 0.5, 0.2\}$ and comparing again the fixed approach ($\varepsilon = 0.2$) and the varying approach, the latter still takes about 70% of the computational time consumed by the former. This illustrates how the combination of varying accuracy in the evaluation of candidate solutions and approximation-based local search operators can save computational effort in expensive CAD problems, while achieving solutions of good quality.

7.5 Final Remarks

This chapter described a framework for expensive optimization that is based on the combination of $\mathscr{A}$-MAs and varying the accuracy of the black-box functions during the optimization. The approximation-based local search should take into account the fact that some evaluations are performed with greater accuracy than others, when building the local approximations. The methodology described in this chapter has the following benefits:

- It can tackle constrained mono-objective optimization problems with any number of inequality and equality constraints. The approximation-based local search can be seen as a constraint-handling technique for expensive problems;
- Different approximation techniques can be combined, such as quadratic models and neural networks, but the WLS method should be incorporated into the generation of these models;

- It is possible to choose which nonlinear functions are approximated. If there are analytical or fast-to-evaluate functions in the problem, they do not need to be approximated. Situations like this are easily accomodated by the framework described in this chapter.

Finally, it is worth noting that even though the problem studied here is relatively fast to evaluate, the overhead of the local search is still negligible. More complex real-world CAD problems, ranging from medium to large scale applications, could take minutes to hours to complete one evaluation, specially when dealing with nonlinear and transient 3D problems. In this scenario, any new approach for saving computational effort is welcome, and MAs employing approximation-based local search operators are very promising. Also, we illustrate that high accuracy evaluations are not needed throughout the whole optimization process, saving additional effort. It is reasonable to conjecture that other complex problems in engineering optimization will benefit even more from the proposed methodology.

References

1. Agapie, A.: Genetic algorithms: Minimal conditions for convergence. In: Hao, J.-K., Lutton, E., Ronald, E., Schoenauer, M., Snyers, D. (eds.) AE 1997. LNCS, vol. 1363, pp. 183–193. Springer, Heidelberg (1998)
2. Almaghrawi, A., Lowther, D.A.: Heterogeneity and loosely coupled systems in electromagnetic device analysis. IEEE Transactions on Magnetics 44(6), 1402–1405 (2008)
3. Alotto, P., Kuntsevitch, A.V., Magele, C., Molinari, G., Paul, C., Preis, K., Repetto, M., Richter, K.R.: Multiobjective optimization in magnetostatics: a proposal for benchmark problems. IEEE Transactions on Magnetics 32(3), 1238–1241 (1996)
4. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, New York (1996)
5. Coello, C.A.C.: Theoretical and numerical constraint handling techniques used with evolutionary algorithms: a survey of the state of the art. Computer Methods in Applied Mechanics and Engineering 191(11-12), 1245–1287 (2002)
6. Coello, C.A.C.: Evolutionary multi-objective optimization: a historical view of the field. IEEE Computational Intelligence Magazine 1(1), 28–36 (2006)
7. Dawkins, R.: Memes: the new replicators. The Selfish Gene, ch. 11. Oxford University Press, Oxford (1976)
8. Fletcher, R.: Practical Methods of Optimization, 2nd edn. John Wiley & Sons, Chichester (1987)
9. Gen, M., Cheng, R.: Genetic Algorithms and Engineering Design. John Wiley & Sons, New York (1997)
10. Graham, I.J., Case, K., Wood, R.L.: Genetic algorithms in computer-aided design. Journal of Materials Processing Technology 117(1-2), 216–221 (2001)
11. Guimarães, F.G., Wanner, E.F., Campelo, F., Takahashi, R.H.C., Igarashi, H., Lowther, D.A., Ramírez, J.A.: Local learning and search in memetic algorithms. In: Proceedings of the IEEE Congress on Evolutionary Computation. IEEE World Congress on Computational Intelligence, pp. 2936–2943. IEEE Press, Los Alamitos (2006)
12. Guimarães, F.G., Campelo, F., Igarashi, H., Lowther, D.A., Ramírez, J.A.: Optimization of cost functions using evolutionary algorithms with local learning and local search. IEEE Transactions on Magnetics 43(4), 1641–1644 (2007)

13. Haykin, S.: Neural Networks and Learning Machines, 3rd edn. Prentice-Hall, Englewood Cliffs (2008)
14. Ishibuchi, H., Murata, T.: A multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Transactions on Systems, Man, and Cybernetics – Part C 28(3), 392–403 (1998)
15. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Transactions on Evolutionary Computation 7(2), 204–223 (2003)
16. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing: A Fusion of Foundations, Methodologies and Applications 9(1), 3–12 (2005)
17. Johnson, J.M., Rahmat-Samii, V.: Genetic algorithms in engineering electromagnetics. IEEE Antennas and Propagation Magazine 39(4), 7–21 (1997)
18. Kimura, S., Konagaya, A.: High dimensional function optimization using a new genetic localsearch suitable for parallel computers. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 1, pp. 335–342. IEEE Press, Los Alamitos (2003)
19. Knowles, J.D., Corne, D.W.: Memetic algorithms for multiobjective optimization: issues, methods and prospects. In: Hart, W.E., Krasnogor, N., Smith, J.E. (eds.) Recent Advances in Memetic Algorithms. Studies in Fuzziness and Soft Computing, vol. 166, pp. 313–352. Springer, Heidelberg (2004)
20. Krasnogor, N., Smith, J.: A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Transactions on Evolutionary Computation 9(5), 474–488 (2005)
21. Lowther, D.A.: Automating the design of low frequency electromagnetic devices – a sensitive issue. COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering 22(3), 630–642 (2003)
22. Lozano, M., Herrera, F., Krasnogor, N., Molina, D.: Real-coded memetic algorithms with crossover hill-climbing. Evolutionary Computation Journal 12(3), 273–302 (2004)
23. Merz, P., Freisleben, B.: A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 3, pp. 2063–2070. IEEE Press, Los Alamitos (1999)
24. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Tech. Rep. C3P Report 826, Caltech Concurrent Computation Program (1989)
25. Moscato, P.: Memetic algorithms: a short introduction. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization. Advanced Topics in Computer Science, pp. 219–234. McGraw-Hill, New York (1999)
26. Norris, J.R.: Markov Chains. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge (1997)
27. Ong, Y.S., Keane, A.J.: Meta-lamarckian learning in memetic algorithms. IEEE Transactions on Evolutionary Computation 8(2), 99–110 (2004)
28. Ong, Y.S., Nair, P.B., Keane, A.J.: Evolutionary optimization of computationally expensive problems via surrogate modeling. American Institute of Aeronautics and Astronautics Journal 41(4), 687–696 (2003)
29. Ong, Y.S., Lim, M.H., Zhu, N., Wong, K.W.: Classification of adaptive memetic algorithms: a comparative study. IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics 36(1), 141–152 (2006)
30. Park, J., Sandberg, I.W.: Universal approximation using radial-basis-function networks. Neural Computation 3(2), 246–257 (1991)

31. Renner, G., Ekart, A.: Genetic algorithms in computer aided design. Computer-Aided Design 35(8), 709–726 (2003)
32. Rudolph, G.: Convergence properties of canonical genetic algorithms. IEEE Transactions on Neural Networks 5(1), 96–101 (1994)
33. Seneta, E.: Non-negative Matrices and Markov Chains. Springer Series in Statistics. Springer, Heidelberg (1981)
34. Storn, R.M., Price, K.V.: Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)
35. Takahashi, R.H.C., Vasconcelos, J.A., Ramírez, J.A., Krahenbuhl, L.: A multiobjective methodology for evaluating genetic operators. IEEE Transactions on Magnetics 39(3), 1321–1324 (2003)
36. Wang, G.G., Shan, S.: Review of metamodeling techniques in support of engineering design optimization. Journal of Mechanical Design 129(4), 370–380 (2007)
37. Wanner, E.F., Guimarães, F.G., Takahashi, R.H.C., Saldanha, R.R., Fleming, P.J.: Constraint quadratic approximation operator for treating equality constraints with genetic algorithms. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 3, pp. 2255–2262. IEEE Press, Los Alamitos (2005)
38. Wanner, E.F., Guimarães, F.G., Takahashi, R.H.C., Ramírez, J.A.: Hybrid genetic algorithms using quadratic local search operators. COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering 26(3), 773–787 (2007)
39. Wanner, E.F., Guimarães, F.G., Takahashi, R.H.C., Fleming, P.J.: Local search with quadratic approximations into memetic algorithms for optimization with multiple criteria. Evolutionary Computation Journal 16(2), 185–224 (2008)
40. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)
41. Zhou, P., Fu, W.N., Lin, D., Stanton, S., Cendes, Z.J.: Numerical modeling of magnetic devices. IEEE Transactions on Magnetics 40(4), 1803–1809 (2004)
42. Zhou, Z., Ong, Y.S., Lim, M.H.: Memetic algorithm using multi-surrogates for computationally expensive optimization problems. Soft Computing (11), 957–971 (2007)

Chapter 8
Opportunities for Expensive Optimization with Estimation of Distribution Algorithms

Alberto Ochoa

Abstract. The chapter claims that the search distributions of Estimation of Distribution Algorithms (EDAs) contain much information that can be obtained with the help of modern statistical techniques to create powerful strategies for expensive optimization. For example, it shows how the regularization of some parameters of the EDAs probabilistic models can yield dramatic improvements in efficiency. In this context a new class, Shrinkage EDAs, based on shrinkage estimation is presented. Also, a novel mutation operator based on a regularization of the entropy is discussed. Another key contribution of the chapter is the development of a new surrogate fitness model based on the search distributions. With this method the evolution starts in the fitness landscape, switches to the log-probability landscape of the model and then backtracks to continue in the original landscape if the optimum is not found. For the sake of completeness the chapter reviews other techniques for improving the sampling efficiency of EDAs. The theoretical presentation is accompanied by numerical simulations that support the main claims of the chapter.

8.1 Introduction

Nowadays, the optimization of computationally expensive black-box functions has become a task of great practical importance that requires new theoretical developments. In particular, in the area of evolutionary algorithms (EAs) several approaches have been studied so as to reduce the number of function evaluations.

The main message of this chapter is: the area of Estimation of Distribution Algorithms (EDAs) [12, 18, 26, 27, 37], a state-of-the-art branch of EAs, can play a leadership role in expensive optimization. However, it is important to recognize that dealing with expensive optimization problems within the framework of EDAs is

Alberto Ochoa
Institute of Cybernetics, Mathematics and Physics, Calle 15 No. 551, CP 10400,
Ciudad Habana, Cuba
e-mail: ochoa@icmf.inf.cu

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 193–218.
springerlink.com

a very complex issue that has to be attacked from several different directions at the same time. Therefore, our main point is the following affirmation: the search distributions contain a lot of information that can be extracted with the help of modern statistical techniques to be used to elaborate powerful strategies for expensive optimization.

To begin with, it is crucial to understand that *in EDAs the distribution is the central concept, as opposed to populations and individuals*. We believe that it is from this understanding that these new research opportunities will arise.

In [32] several developments of this idea related to both the entropy of the search distributions and the entropy of the optimization problems were presented. Two interesting issues were raised: maximum entropy and entropic mutation, and the important role of the Boltzmann distribution in the theory of EDAs was stressed.

The influence of the quality of the utilized probabilistic model in the performance of an EDA algorithm was another issue investigated in [32]. It was made clear that building a ”good” model is important, but at the same time it was recognized that ”good” does not always point out the model with the most information.

Throughout the chapter the above-mentioned issues are revisited and other unpublished or published elsewhere are presented. Our goal is to highlight the main directions that can be taken today to boost the capabilities of current EDAs to cope with expensive problems.

The outline of the chapter is as follows. To begin with, a brief introduction on Estimation of Distribution Algorithms (EDAs) is given in Sect. 8.2. Then, we review three different perspectives on a basic problem: enhancing the sampling efficiency of EDAs. This occurs in Sect. 8.3, which is divided in three subsections: improving selection methods (Sect. 8.3.1), accelerating the convergence (Sect. 8.3.2) and building statistically efficient sampling distributions (Sect. 8.3.3).

Section 8.4 presents the first of the two main contributions of this chapter: a method and an algorithm that do not evaluate all visited search points. The evolution occurs in two different landscapes, in one of them the fitness is estimated. The method is called Partial Evaluation with Backtracking from Log-Probability Landscapes.

Section 8.5 is devoted to `regularization`, a new important area of research in EDAs. We will show how the regularization of the probabilistic models used in EDAs can yield dramatic improvements in efficiency. In Sect. 8.5.1 we show how to design mutation operators that do not destroy the learned distributions. This technique, called `Linear Entropic Mutation` (LEM), increases the entropy of the probabilistic models and produces higher percentage of success rates and fewer function evaluations. It is a natural operator for EDAs because it mutates distributions instead of single individuals. Finally, in Sect. 8.5.2 the second main contribution of our work is presented: an efficient technique for accurate model building with small populations. With this we are opening the doors of EDAs research to a new class of efficient algorithms based on shrinkage estimation. We propose the name `Shrinkage Estimation of Distribution Algorithms` (SEDA) for the whole class. We give a brief introduction of the SEDAs by means of a detailed discussion of one of its members: $SEDA_{mn}$ (an algorithm that uses the multivariate normal distribution).

The chapter includes an appendix dedicated to the `B-functions` -a new benchmark function model especially designed for EDAs. Some of the functions used in our simulations are of this type. Finally, the conclusions of the chapter are given.

8.2 Estimation of Distribution Algorithms

A fundamental concept in the field of EDAs is that of `search distribution`, which is the distribution from where new points are sampled at every evolutionary step. In this chapter we are interested in EDAs that construct Bayesian network models of the search distributions.

The basic evolutionary cycle of an EDA is as follows. The algorithm uniformly samples the solution space to create an initial population. The population is then updated for a number of generations. First, a set of promising solutions (`the selected set`) is chosen using truncation, tournament or Boltzmann selection. A Bayesian network that captures the correlations of the selected set is constructed and new solutions are simulated from it. Finally, the new solutions are combined with the old ones and a new population – with better properties – is constructed. The algorithm iterates until some termination criteria are met.

Learning the Bayesian network is the critical step. Some algoritms use a greedy procedure that, starting from an empty (complete) graph, at each step adds (deletes) the edge that improve a certain metric, which is defined over the set of all acyclic graphs. Other algorithms, like the one we shall present in Sect. 8.4.3, use statistical tests instead.

After more than ten years of research there exists a tremendous amount of literature about EDAs. The interested reader can easily find many detailed presentatios on the topic. The references of this chapter might be good starting points.

The Boltzmann distribution has been recognized by some authors [22, 25, 27, 32] as a crucial ingredient of EDAs. In Sect. 8.4.2 it will play an important role. Therefore, in the following section we give a brief introduction to a Bayesian EDA that uses Boltzmann selection.

8.2.1 Boltzmann Estimation of Distribution Algorithms

Hereafter, X_i represents a scalar random variable and $p(x_i) = p(X_i = x_i)$ its probability mass function with $x_i \in X = \{0, 1, \ldots, K\}$, Note that $p(x_i)$ and $p(x_j)$ refer to two different random variables, and have in fact different probability mass functions, $p(X_i = x_i)$ and $p(X_j = x_j)$, respectively. Similarly, $X = (X_1, X_2, \cdots, X_n)$ denotes a n-dimensional random variable, $x = (x_1, x_2, \cdots, x_n)$ is a configuration and $p(x_1, x_2, \cdots, x_n)$ represents a joint probability mass.

Definition 1. *For $\beta \geq 0$ define the Boltzmann distribution of a function $f(x)$ as*

$$p_{\beta,f}(x) := \frac{e^{\beta f(x)}}{\sum_y e^{\beta f(y)}} = \frac{e^{\beta f(x)}}{Z_f(\beta)}$$

where $Z_f(\beta)$ is the partition function.

We also use $Z_{\beta,f}$, but to simplify the notation β and f can be omitted. If we follow the usual definition of the Boltzmann distribution, then $-f(x)$ is called the free energy and $1/\beta$ the temperature of the distribution. The parameter β is usually called the inverse temperature.

Closely related to the Boltzmann distribution is Boltzmann selection:

Definition 2. *Given a distribution $p(x)$ and a selection parameter γ, Boltzmann selection calculates a new distribution according to*

$$p^s(x) = \frac{p(x)e^{\gamma f(x)}}{\sum_y p(y)e^{\gamma f(y)}}$$

Boltzmann selection is important because the following holds [27]:

Theorem 1. *Let $p_{\beta,f}(x)$ be a Boltzmann distribution. If Boltzmann selection is used with parameter γ, then the distribution of the selected points is again a Boltzmann distribution with*

$$p^s(x) = \frac{e^{(\beta+\gamma)f(x)}}{\sum_y e^{(\beta+\gamma)f(y)}}$$

On the basis of the above, in [27] the Boltzmann Estimation of Distribution Algorithm (BEDA) was introduced. It is an EDA with Boltzmann selection and an annealing schedule for the temperature [22].

Lemma 1. *$\Delta\beta(t) = c/\sqrt{Var_f(\beta(t))}$ leads to an annealing schedule where the average fitness, $W_f(\beta(t))$, increases approximatively proportional to the standard deviation:*

$$W_f(\beta(t+1)) - W_f(\beta(t)) \approx c\sqrt{Var_f(\beta(t))}$$

where c is a constant and $Var_f(\beta(t)) = \sigma_f^2(\beta(t))$ is the variance of the fitness function. This annealing schedule has been called Standard Deviation Schedule (SDS).

The exponential complexity of computing the partition function can be avoided if the Boltzmann distribution is approximated with a tractable distribution. There are several ways of accomplishing this approximation, However, for the purposes of this chapter we restrict ourselves to the special case covered by the Factorization Theorem [27], for dealing with additively decomposable functions of bounded complexity. This theorem defines how and under what conditions the search distributions associated to discrete functions can be factorized. The factorization follows the structure of the function and is only exact if the function obeys certain structural constraints known as the running intersection property [14]. Besides, we shall assume that we have a good Bayesian-network-learning algorithm capable of discovering the underlying function's structure, which is not distorted (under the above conditions) by Boltzmann selection. A BEDA with a Bayesian network probabilistic model is a Bayesian BEDA algorithm. One example is given in Sect. 8.4.3.

8.3 Three Opportunities for Enhancing the Efficiency of EDAs

The main contributions of this chapter will be presented in Sect. 8.4 and in Sect. 8.5. In one case we build an accurate approximation of the search distribution to construct a surrogate model for the fitness function. In the other case, the challenge is to build accurate models of the search distributions from small samples.

For the sake of completeness, in this section, we want to discuss briefly three others techniques aimed to increase the sampling efficiency of EDAs. The central concept is again the probability distribution. We believe that all these methods deserve more attention from those scientists interested in experimenting with EDAs in the area of expensive optimization.

8.3.1 Right-Sized Selection

What are the correct number of points that the selected set should have?

The answer that one can give to this question based on the current practice depends on the selection method used. For truncation selection a fraction of the population size (often 30%) is the popular choice, whereas for tournament and Boltzmann selection the widely accepted practice is to create selected sets as large as the corresponding populations.

The experiment shown in Table 8.1 and the discussion of this section is taken from [44]. An EDA with Boltzmann selection that learns a Bayesian network by optimizing the BDe scoring metric is investigated with the DeceptiveK[1] function of order three. The size of the selected set is equal to $f * N$, where N is the population size.

To improve the 83% success rate (%Success) found in the table we have two options: we can increase the population size or the size of the selected set. The later gives the best result: a success rate of 96% is obtained without making any additional function evaluation. In fact, the number of evaluations drops. The alternative produces an increase of the number of function evaluations. Note that according to the experiment, the increase of f does not seem to influence the convergence time, G_c.

Table 8.1 A Boltzmann EDA with Bayesian learning and the DeceptiveK ($K = 3$). The cell triplets mean %Success, G_c (convergence time), #Evaluations (number of function evaluations). The sizes of the population and selected set are N and $f * N$ respectively

N	$f = 1$	$f = 4$	$f = 12$
400	83, 7, 2689	93, 6, 2546	**96, 6, 2408**
450	94, 6, 2844	99, 6, 2645	99, 6, 2614
500	99, 6, 3025	100, 6, 2920	100, 6, 2750

[1] Müehlenbein's K-Deceptive function, which is easier than Goldberg's Deceptive3.

Table 8.2 Probabilistic Elitism. PADA with Goldberg's Deceptive3. (See [42] page 94)

Elit-size	Prob-Elit-size	G_c	%Success	#Evaluations
0	0	6.75 ± 1.43	79	**3375**
150	0	7.04 ± 1.46	100	2614
300	0	9.07 ± 2.29	100	**2114**
450	0	14.90 ± 4.91	**20**	-
300	5	8.05 ± 1.60	96	1945
300	10	7.36 ± 1.68	98	**1835**

An important equation in EDAs is the following:

$$p(x,t+1) \approx p^s(x,t). \tag{8.1}$$

We call this equation *hypothesis of similarity*, others authors refer to it indirectly by claiming that EDAs do the search in promising zones of the space. Note that the symbol $\approx$ means there are two distributions, that despite their similarity, are not the same. This subtle detail seems to be ignored by most of the papers on EDAs. Some of them usually claim that the distribution of the selected set $p^s(x,t)$ is the one that is used as the distribution $p(x,t+1)$; but is this really so? Actually, what EDA does is to construct the distribution $p(x,t+1)$ from a sample, the selected set, which comes from the distribution $p^s(x,t)$. This implementation has the following important problem: if the sample size is reasonably smaller than the size determined by the sample complexity of $p^s(x,t)$, then $p(x,t+1) \approx p^s(x,t)$ will no longer hold. It does not matter how well we know $p^s(x,t)$; if there are not enough points in the sample, the equation will not be correct. Let us recall that the sample complexity is defined as the minimum sample size needed in order to recover the distribution where the sample comes from.

Fortunately, it is possible to improve the quality of the selected set, so that it allows a better estimation of $p(x,t+1)$, without increasing the number of functions evaluations. In Boltzmann selection, for example, it is enough to increase the number of points sampled by selection. All this means that, in principle, we have the capability of increasing the efficiency of the EDA algorithm without affecting its efficacy. Another much more important fact is that this gives the possibility of using EDAs with small populations.

The right-sized selection may create favorable conditions for the technique we will introduce in Sect. 8.4 because it allows a better estimation of the distribution $p(x,t+1)$.

8.3.2 Probabilistic Elitism

Even a simple evolutionary operator like elitism can profit from the EDA emphasis on distributions. If we can compute the modes of the search distribution why do not

Table 8.3 The impact of using maximum-entropy search distributions. (See [34])

N	*%Success*	*Gc*	$\%Success^{ME}$	Gc^{ME}
200	0	-	2	8.5 ± 0.7
600	8	9.7 ± 1.5	69	7.4 ± 1.1
800	10	8.7 ± 3.2	**90**	$\mathbf{7.0 \pm 1.2}$
5000	**92**	$\mathbf{7.2 \pm 1.2}$	100	5.8 ± 0.9

^{ME} Results when maximum-entropy search distributions are used.

grant them the right to be in the next population as it is done with the best current individuals? The term probabilistic elitism was chosen for this method in [42].

Table 8.2, shows an interesting experiment about the synergy between traditional and probabilistic elitism. The algorithm used is PADA, the Polytree Approximation Distribution Algorithm [43], one of the pioner EDAs. The algorithm learns polytree Bayesian networks from the selected populations. It makes independence tests to construct the skeleton and to orient some edges, then a BIC score guides a hillclimbing local search to orient the remainder edges.

Notice that increasing the classic elitism decreases the number of function evaluations, but this has an upper bound beyond which the efficacy (%Success) of the algorithm drops (see $Elitism = 450$). Adding the five or ten best configurations of the search distributions accelerates the convergence of the algorithm and produces an additional reduction of the number of evaluations. As a result of the combined use of the methods 1540 function evaluations were saved. Simple, but effective! It is worth noting from the point of view of expensive optimization (see Sect. 8.4.2), that the probabilistic elite can be included in the new population without being evaluated.

When it is possible to build a junction tree for a discrete problem, the probabilistic elite –the M most probable configurations– can be computed with the Nilsson's algorithm [28]. Alternatively, the algorithm introduced in [45], which uses max-marginals instead of a junction tree, can be used for arbitrary discrete graphical models. For some continuos problems it is also possible to sample zones of high probability with similar results, for example, around the mean of the multivariate normal distribution.

We look at the probabilistic elitism as an acceleration method that compensates for the delay introduced by classic elitism and the regularization techniques that will be discussed later in this chapter.

8.3.3 Maximum Entropy

In EDAs, two good reasons for building a sampling distribution of maximum entropy are: 1) the sample size is limited and we know that we will not be able to estimate correctly marginals of order larger than a given low integer K; 2) a model of higher order has been estimated with statistics of lower order. The first scenario

corresponds to a common situation in expensive optimization. An example of the second is shown in Table 8.3.

Two EDAs use the same structure learning algorithm, which only makes independence tests of order two to approximate a polytree distribution. The point is that the estimation of the conditional probability associated to a variable with more than one parent involves marginal distributions of order larger than two. Therefore, although the structure can be obtained with a small population the sampling distribution may need a larger population size to learn its parameters.

Parameter learning at the simulation step is different for each EDA. The first algorithm does probabilistic logic sampling with the maximum likelihood (MLE) estimates of the conditional probabilities. The second algorithm, builds a junction tree, fixes the second order marginals and for each clique computes the maximum entropy distribution that is consistent with the fixed marginals. Note that without maximum-entropy (second and third columns in Table 8.3) the algorithm needs six times more function evaluations to achieve a 90% success. For details see page 29 in [32].

If we have the joint distribution with the maximum-entropy among all the joints that fulfill a given collection of marginals, choosing a joint with less entropy amounts to adding some information that is not justified by the marginal constraints. The Iterative Proportional Fitting (IPF) algorithm can be used to find the maximum-entropy distribution [8, 9, 16, 40]. For large distributions, an efficient implementation of the maximum-entropy algorithm was developed in [10, 23]. The general idea is to improve the performance of IPF by combining it with the junction tree technique. It consists of performing IPF locally on the nodes and passing messages to the neighboring nodes. It has been proved that this converges to the unique maximum-entropy solution, so it is equivalent to IPF. The reader is referred to [34] for details on the implementation of the method for computing maximum-entropy distributions of polytrees.

8.4 Evolutionary Backtracking from Log-Probability Landscapes

In this section we present a novel computational scheme for decreasing the number of function evaluations of an EDA algorithm without harming, or even improving its convergence properties.

At every generation, if the EDA does not find the optimum it restarts the search in the logarithmic space of the probabilities assigned to every configuration by the current model of the selected set. If after searching for some generations in the log-space the optimum is not found, the algorithm backtracks to its previous state in the fitness function space.

Among other things, we will show that the success of the method depends on the quality of the probability model built by the EDA algorithm. Therefore, as a first step, we must choose an algorithm capable of building accurate models of the search distributions.

8.4.1 Selecting a Bayesian EDA Algorithm

Powerful Bayesian EDA algorithms for integer and real valued optimization were introduced in [20] and [21] (two of them, MMHC-EDA and TPDA-EDA, are shown in bold in Table 8.4). Unlike most of the existing approaches these methods are mainly based on tests of (in)dependence, in contrast to other well-known algorithms like BOA [36] and EBNA [2] that only use scoring metrics. In the new algorithms, the structure of the search distribution is learned with a modification of the algorithms reported in [3, 7]. The interested reader is refered to [19] for a detailed discusion of sequential and parallel versions of these algorithms. The presentation that follows is partially adapted from [20] and [21].

The experiments of this section use the class of random B-functions that was introduced in [32], later extended in [33] and further investigated in [30]. We have included a short introduction to this benchmark in the appendix. We recommend reading it before the remainder of this section.

Table 8.4 shows that MMHC-EDA and TPDA-EDA do many fewer function evaluations than BOA and EBNA in a collection of ease uncorrelated problems. The UMDA is the best, which is not a surprise because it knows the problem structure.

Table 8.5 presents the success rates the MMHC-EDA and other algorithms achieved with the hard random polytree B-function class: BF2B30s4-1245, for different population sizes. Notice the high value of the univariate entropies. As far as the pairwise mutual informations belong to the interval $[0.1, 0.2]$ we can conclude that the bivariate entropies are also high. Therefore, the algorithm must discover the conditional higher order dependence structure of the problem to be able to find the optimum.

The results are conclusive. UMDA does nothing due to the high univariate entropies. BOA with $k = 1$ deals with bivariate marginals, which also have high entropy. Interestingly, neither EBNA nor the other BOA behave better than BOA with $k = 1$. The MMHC-EDA algorithm is the winner, it succesfully learns the problem structure although it needs a large population due to the high entropy.

Table 8.4 Average results on 10 instances of a random class BF2B100s0-0012

	N^{*a}	%Success	$G_c{}^b$	%Evaluations
UMDA	100	98.70	11.98	1198.78±3.01
MMHC-EDA	250	95.30	10.97	2742.65±9.61
TPDA-EDA	250	98.60	10.54	2635.65±8.64
$BOA_{(k=1)}$	280	97.80	10.83	3034.19±11.48
$BOA_{(k=3)}$	700	97.80	11.09	7765.13±27.16
EBNA	150	97.10	28.85	4328.53±36.37

a critical population size - Minimum size to achieve 95% success rate.
b generation where the optimum is found.

Table 8.5 Minimization in the hard class of random polytree B-functions, BF2B30s4-1245. The cells contain the succcess rate

N	$UMDA$	$\frac{BOA}{k=1}$	$\frac{BOA}{k=2}$	$\frac{BOA}{k=3}$	$EBNA$	$\frac{MMHC}{EDA}$
300	3.33	46.67	23.33	13.33	30.00	40.00
500	3.33	50	56.67	43.33	26.67	53.33
1000	3.33	43.33	50	50	40.00	73.33
2000	3.33	43.33	50	53.33	43.33	100

In summary, we believe that MMHC-EDA is the best candidate for our purposes. It seems to obtain accurate approximations of the search distributions, which is highly convenient for the technique we will introduce in the next section.

8.4.2 Partial Evaluation in Boltzmann EDAs

The term `Partial Evaluation` (PE) was introduced in evolutionary computation to deal with the problem of evaluating evolutionary algorithms with expensive fitness functions [31]. Some related techniques are fitness inheritance [11, 41], variable-fidelity [6] and surrogate models [4, 17, 35, 47].

The name partial evaluation, which was borrowed from the literature on logic programming, emphasizes the fact that a certain amount of individuals and/or populations are not completely evaluated during the execution of the algorithm. In [31] the authors explored the feasibility of building a neural network model to predict the fitness of individuals, based on information of the genotypes and fitness values of their parents and the genetic operations performed.

Many PE strategies can be and have been implemented with genetic algorithms. However, although there have been some interesting works with EDAs too, we believe that there are still many research opportunities in this area.

In EDAs, a basic idea underlies the method: building a fitness surrogate model using the probability distribution of the selected set. This has been called Partial Evaluation of an EDA algorithm in [42].

In what follows, we will discuss an experiment presented in [42]. The results are shown in Table 8.6.

The Onemax function counts the number of **1**s in its input variable. The Boltzmann Univariate Marginal Distribution Algorithm (BUMDA) is an EDA that assumes probability independence among the variables and uses Boltzmann selection. In the experiment the fitness function is evaluated during the first few (first column in the table) generations. Afterwards, the fitness surrogate model that will be presented below is used to evaluate the population.

The table presents average results for 100 runs. The last row is left for the case when no partial evaluation is used: convergence time $G_c = 7.86$ (the count

Table 8.6 Partial Evaluation of the OneMax function with the Boltzmann Univariate Marginal Distribution Algorithm (BUMDA). (See [42], page 83)

G_{PE}	G_c	%Success	#Evaluations	%Estimated
1	4.77	48	5770	83
2	4.95	80	5950	66
3	5.47	95	6470	54
4	5.97	100	6070	43
-	7.86	100	8860	0

starts at 0), number of function evaluations $\#Evals = 8860$, percentage of times the function's model was used $\%Estimated = 0$ and the percentage of success $\%Success = 100$. The problem and population sizes were set at $n = 60$ and $N = 1000$ respectively. We used no elitism.

The third row, shows the results when the estimation starts at the fourth generation. This time the algorithm made 6470 function calls, but only 46% of them were actual fitness evaluations. Therefore, 5883 functions evaluations were saved and still we got a high percentage of success, 95%. Another interesting observation is that with PE the algorithm converged faster. The reduction in the number of evaluations is the result of the combination of a faster convergence and the use of the fitness model. Notice, that in this case the algorithm converged two and a half generations after the start of PE, but still before than the required number of generations without PE.

It is worth noting that this example is just an academic illustration of the PE idea. In reality, the function is easy and without PE such a large population size is not needed. The point is that the quality of the PE fitness function model depends on the quality of the search distribution as it will be shown below. As far as the structure of the model is known, a large population size guarantees and adequate estimation of its parameters.

We now introduce the used surrogate model. From Definition 1, it is easy to see that with BEDA, the optimization of $f(x)$ amounts to the optimization of the function $\frac{\log\left(p_{\beta,f}(x)\right)}{\beta}$. The computation of the standard deviation annealing schedule now should be carried out with the formula:

$$\triangle\beta = \frac{c.\beta}{\sqrt{\sigma^2_{\log\left(p_{\beta,f}(x)\right)}}}$$

The problem with the above is that in general we do not know the distribution $p_{f,\beta}(x)$. However, we do know that the Factorization Theorem [27] tells us that under certain mild conditions the Boltzmann distribution of additively decomposable functions has a computable Bayesian factorization. Denoting it by $p^a_{f,\beta}(x)$, we

propose to use the function $\frac{\log\left(p^a_{\beta,f}(x)\right)}{\beta}$ as a PE model of $f(x)$. This was the model used in the experiment.

In practice, we deal with finite populations and therefore BEDA approximates the Boltzmann distribution gradually. This is why better results are obtained when this strategy of partial evaluation starts at the latter generations. There exist other PE strategies. For example, it can be shown that for high probable configurations the approximation is better. Thus, one could estimate a fraction of the most probable configurations at the earlier generations.

We now try to quantify the magnitude of the error. Denoting $\triangle_{f,\beta}(x) = f^a(x) - f(x)$ and $f^a(x) = \log\left(p^a_{\beta,f}(x)\right)/\beta + \log\left(Z_f(\beta)\right)/\beta$, we obtain

$$\triangle_{f,\beta}(x) = \frac{\log\left(\frac{p^a_{\beta,f}(x)}{p_{\beta,f}(x)}\right)}{\beta}$$

and taking the expectation with respect to the approximating distribution, we get

$$\left\langle \triangle_{f,\beta}(x) \right\rangle_{p^a_{\beta,f}(x)} = \frac{D\left(p^a_{\beta,f}(x) \left\| p_{\beta,f}(x) \right|\right)}{\beta} \tag{8.2}$$

The error expectation equals the product of the temperature times the Kullback-Liebler divergence between the two distributions. Notice that the last two equations further explain why the function approximation is better toward the end of the run: β is larger.

One interesting observation, which deserves more study, is the fact that for this function the search in the log-probability space seems easier than in the original fitness landscape. In [24] the authors reported a similar behaviour of the UMDA with the Saw function. According to these authors, the rugged fitness landscape of the function is implicitly transformed into a fairly smooth landscape in a space whose coordinates are the univariate probabilities. The algorithm performs a gradient ascent on the transformed landscape and easily gets to the optimum. In this regard the apparent merit of our PE technique is that it explicitly performs the optimization in the probability space. These issues are the subject of ongoing research.

Now the obvious question is whether or not the method can be applied to cases different from those covered by the Factorization theorem. Equation 8.2 drops us a hint. We can apply the PE method if our approximating distribution is close enough to the Boltzmann distribution. This is valid for both discrete and real valued problems. In [46] a variant of Boltzmann selection with an annealing schedule for real variables was reported. The algorithm proposed by these authors computes the multivariate normal distribution that minimizes the Kullback-Liebler distance to the Boltzmann search distribution.

8.4.3 A Simple Algorithm for Partial Evaluation with Backtracking

We begin this section with a similar experiment to the one commented in the previous section. However, there are important differences: 1) truncation selection is used instead of Boltzmann selection; 2) the fitness model is $\log\left(p^a_{\beta,f}(x)\right)$; and 3) the fitness function has proven to be quite challenging even when its dependence structure is known.

It is interesting to explore the feasibility of using truncation selection with PE because it is much faster than Boltzmann selection.

The function used in the experiment was designed in [19] using the B-function formalism developed by the present author and outlined in the appendix. It is called Ten Little Niggers (TLN) because its dependence structure is composed of disjoint groups of 10 variables. The distribution is given by:

$$p(x) = p(x_{10} \mid x_1, x_2 \cdots, x_9) \cdot p(x_{20} \mid x_{11}, x_{12} \cdots, x_{19}) \cdots p(x_{50} \mid x_{41}, x_{42} \cdots, x_{49})$$

In [19] this function was proved to be very difficult for the well-known state-of-the-art Bayesian EDAs: BOA and EMNA, which were not able to optimize (95% success) the function with a population size of 10000 points. The MMHC-EDA, however has a critical population size of 3000 points and makes about 19990 function evaluations[2].

Table 8.7 presents the numerical results. The first column is the generation where the PE starts from. For example, the first row shows the case when the fitness function is evaluated in two generations and the model is applied from the third until convergence (second column). Under this conditions the algorithm converges as average nine times in 100 runs. This is not too much, but when the PE starts at generation five the success rate is already 83%.

The results of Table 8.7 suggest the following simple algorithm. At every generation, if EDA does not find the optimum it restarts the search in the logarithmic space of the probabilities assigned to every configuration by the current model of the selected set. In other words, the fitness model is used instead of the fitness function. Now, if after searching until G_{max} in the log-space the optimum is not found neither, the algorithm backtracks to its previous state in the fitness function space. We can easily obtain an estimate of the average number of function evaluations that this algorithm would make according to the results of the table. Taking the forth column of the table as the percentage of times the algorithm converges in K runs, we obtain after simple arithmetic operations an estimate of about 11000 evaluations which amount to a saving of 45%.

We extended the MMHC-EDA with the described PE algorithm and produce Algorithm 1. Only one new parameter has to be added: G_{PE}, the generation where

[2] The TLN function and C code to work with it is available from the author.

Table 8.7 Optimization of the B-function TLN with 50 variables. The reported averages are computed over 100 runs. Algorithm: **MMHC**, N=3000, Elit=0, G_{max}=20, τ=0.3

PE_{start}	Mean Gen_c	#Evaluations	%Success
3	6.77	6005.8	9
4	6.45	9004.5	42
5	6.66	12003.7	83
6	6.59	15002.5	93
7	6.49	17937.7	94

Table 8.8 Optimization of the function Trap5 with 30 variables. The reported averages are computed over 20 successfuls runs. Algorithm: **MMHC**, N=4000, Elit=0, G_{max}=12, G_{PE}=2

Selection	Partial Evaluation	#Evaluations	%Success
$\tau = 0.3$	yes	19360	60
$\tau = 0.3$	no	21201	100
Boltzmann ($N = N_s$)	yes	**11611**	100
Boltzmann ($N = N_s$)	no	16401	100

the application of the method starts from. Using this algorithm, we have confirmed the predictions made in the previous paragraph.

At this point we conclude that the log-probability fitness model can be effective with truncation selection at least for the investigated function. Unfortunately, this is not always the case as the following example shows.

We investigate our method with the Trap function. It is a separable deceptive problem proposed in [5]. Its global maximum is located at the point $(1,1,\ldots,1)$. Given the function

$$trap(u) = \begin{cases} k, & for\ u = k \\ k-1-u, & otherwise \end{cases}$$

The function (we use $k = 5$) is defined as follows:

$$Trap(\overrightarrow{x}) = \sum_{i=1}^{m} trap(x_{ki-k+1} + x_{ki-k+2} + \ldots + x_{ki}) \tag{8.3}$$

This function has been used extensively in testing of EDAs and genetic algorithms, and solving them has proven to be quite challenging in the absence of a correct knowledge of its dependence structure.

Table 8.8 reveals several interesting issues. The first thing to notice is the significant reduction in the success rate when PE is applied with truncation selection. Besides, the decrease in the number of function evaluations is smaller than the one obtained with Boltzmann selection. In the later case, a reduction of about 5000

evaluations is achieved with the same 100% success. Notice that the best value is about 46% of the maximum number of function evaluations for truncation selection.

It is worth noting, that the results shown in Table 8.8 were computed for a population size less than the critical value for 30 variables. For truncation selection the critical value (97% success in 100 runs) is equal to 5000 and the average number of function evaluations in this case is 23042. We have found that with Boltzmann selection and PE we get a 100% success and an average of 11500 function evaluations. This means that the average saving is of 51%. We also present, in Fig. 8.1, the histograms of the number of function evaluations per optimization run. As it can be seen near 90% of the runs are below 15000 function evaluations when PE is combined with Boltzmann selection. In contrast, more than 95% of the runs without PE require more than 20000 function evaluations and 70% of the PE runs need only half of this amount.

Algorithm 1. MMHC-EDA + Partial Evaluation with Backtracking from Log-Probability Landscapes (**PEBLPL**)

```
Set t ← 1
Randomly generate N ≫ 0 configurations
while stop criteria are not valid do
  one-MMHC-EDA-step(f(x)) {// Evolution in fitness landscape}
  if stop criteria are not valid AND (t ≥ G_PE) then
    save(t, current Population)
    while stop criteria are not valid do
      one-MMHC-EDA-step(log p(x)) {// Evolution in log-probability landscape}
    end while
    if optima are not found then
      restore(t, Population) {// Backtracking}
    end if
  end if
end while
```

```
function one-MMHC-EDA-step(g(x))
  Evaluate the current population with the input function g(x)
  According to a selection method build the selected set SS
  Find the structure of the probability model BN = MMHC(SS)
  Estimate the parameters of p^ss(x,t) using BN and SS
  Generate N new configurations from p(x,t+1) ≈ p^ss(x,t)
  Set t ← t+1
endfunction
```

According to the results shown in this section we cannot still say under what conditions it is possible to use truncation selection with the PE scheme proposed here. We only know so far that for some functions this is posible. Obviously, this topic needs more research. At this point, it is important to say that with the TLN function the PE also works with Boltzmann selection. This is another confirmation

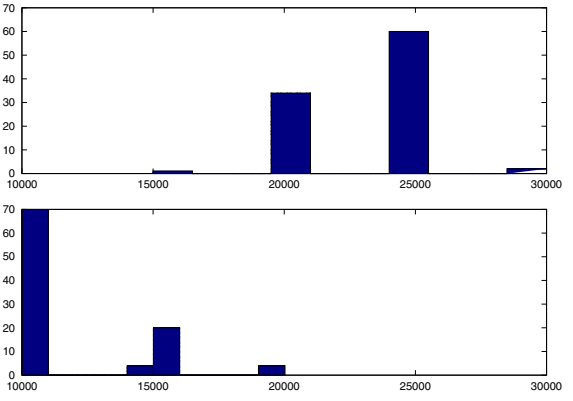


Fig. 8.1 Histograms of the number of function evaluations per optimization run of the Trap function. N = 5000. **top**) Truncation selection without PE. **bottom**) Boltzmann selection with PE

of the theoretical ideas developed in Sect. 8.4.2 and of the important role of the Boltzmann distribution in the theory/practice of EDAs.

It is worth noting, that the proposed Partial Evaluation scheme is beneficial when the actual fitness evaluation is expensive, in which case the above costs are indeed negligible and the model developed in this section valid.

8.5 Regularization in Estimation of Distribution Algorithms

In this section we present theoretical and empirical evidences supporting the following claim:

> The regularization of the parameters of the search distributions can bring enormous savings with respect to the number of function evaluations.

We discuss two examples. In the first one the entropy of the search distribution is the parameter to regularize. Due to space constraints, we give just a small introduction to the topic and leave the details for a forthcoming publication. In the second example, the covariance matrix of a multivariate normal search distribution is regularized. In this case we give more details and introduce a new EDA algorithm.

8.5.1 Entropic Mutation

Thinking of distributions when talking about mutation suggest mutating distributions instead of individuals. In [32] we proposed a method for doing so. The approach changes linearly the entropy of the distribution to achieve the amount of disorder usually understood as mutation.

Table 8.9 Entropic mutation of the search distributions. UMDA with OneMax function for fixed population size

α	0	0.05	0.1	0.125	0.2
%Success	17	74	97	100	99
#Evaluations	300	465	425	412	450

Table 8.10 A Bayesian EDA and the LEM mutation for the Goldberg's Deceptive 3 function, $n = 18$

N	%Success	#Evaluations
500 ($\alpha = 0$)	86	2211
600 ($\alpha = 0$)	88	2815
700 ($\alpha = 0$)	96	3186
800 ($\alpha = 0$)	100	3306
420 ($\alpha = 0.08$)	100	2600
420 ($\alpha = 0.10$)	100	2450
420 ($\alpha = 0.12$)	100	2400
420 ($\alpha = 0.14$)	100	2300
420 ($\alpha = 0.20$)	100	2200

Table 8.9 presents the results of a small experiment. For a fixed small population size, which is not enough to obtain a good success rate without mutation, it is possible to boost the efficacy of the Univariate Marginal Distribution Algorithm by increasing the mutation intensity, α.

The approach was called linear entropic mutation (LEM) in [32]. The LEM acts as a regularizer of the entropy of the system and computes a convex sum of the current and the maximum entropy with the regularization parameter α. In this way the distribution is shrunk toward the maximum entropy distribution. It turns out that this process can be interpreted as a mutation process as far as it increases the level of uncertainty or randomness in the system. For multivariate discrete systems the following definition introduces the LEM.

Definition 3. *Let* $p(x_1, x_2, \cdots, x_n)$ *and* $p_\alpha(x_1, x_2, \cdots, x_n)$ *denote a discrete joint probability mass and its LEM-mutation with mutation intensity* α. *If* $H(X)$ *and* $H_\alpha(X)$ *are their respective entropy values, then the following holds:*

$$\delta H(X) = (n - H(X))\,\alpha \quad \text{and} \quad H_\alpha(X) = (1-\alpha)H(X) + n\alpha \tag{8.4}$$

The computation of $p_\alpha(x_1, x_2, \cdots, x_n)$ given $p(x_1, x_2, \cdots, x_n)$ is challenging. Notice that the distributions must be similar, which means that the mutation does not destroy the learned distributions. For $n = 1$ it is easy. It was explained in [32]. In [29] we report a method for the case $n > 1$. Here we use it in a very simple example,

just to show that the LEM is another regularization technique that can reduce the number of function evaluations.

The Goldberg's Deceptive3 function with 18 variables is optimized with an EDA that learns a Bayesian network using a scoring metric. All the results are averages over 30 runs.

Table 8.10 shows that without mutation to achieve a success rate larger than 95% more than 3000 function evaluations are needed. In contrast, for a fixed population size equal to 420 and for all the mutation intensities shown (starting at 0.08) a 100% success is obtained with many fewer function evaluations. For $\alpha = 0.2$, the table shows a saving of more than 30%!

8.5.2 Shrinkage Estimation of Distribution Algorithms

In this section we propose a new class of EDAs that we believe will soon find its place in the arsenal of tools for expensive optimization problems.

Equation (8.4) was to the best of our knowledege the first attempt to regularize the entropy in such a way, i.e. shrinking it toward the maximum entropy. It is also the only attempt so far to model mutation as an entropic variation in the area of evolutionary algorithms. In this approach the entropy is first shrunk linearly and then the probabilities are computed. An alternative could be a linear shrink of the probabilities followed by the computation of the new value of entropy.

The equation underlying the idea of modelling mutation with the concept of entropy was derived based only on evolutionary arguments [32]. However, there is a strong connection between Linear Entropic Mutation and the general principles behind shrinkage estimation and the analytic approach by Ledoit and Wolf [15] for determining the optimal shrinkage level.

The connection between shrinkage estimation (SE) and LEM have suggested us the existence of a general class of EDAs based on the many different aspects of SE. We have called the new class Shrinkage Estimation of Distribution Algorithms (SEDAs).

Shrinkage Estimation gives EDAs the ability of building better models of the search distributions under small populations. Having better models is important for implementing partial evaluation strategies.

Our main claim is that the synergy between shrinkage estimation, small populations and partial evaluation offers a great research opportunity with regard to expensive optimization problems.

Due to space constraints the complex issues of the combination of the above mentioned methods are not discussed in the chapter. We recall that the material presented in Sect. 8.3 is relevant to the small population issue. Hereafter, we concentrate ourselves on the impact of shrinkage estimation alone.

The now well-known "Small n, Large p" problem of machine learning can be mapped to the "small population" problem of EDAs. The direct link between the later and the optimization of expensive functions is obvious. Therefore, most of the developments in one of the fields must be useful in the other.

The class of SEDAs is large because almost any EDA algorithm can be improved by shrinking some of its distribution parameters. As far as, in this chapter we have focused so far in discrete optimization problems, we decided to introduce in this section a continuos member of the SEDA class: an algorithm that estimates the multivariate normal distribution: $SEDA_{mn}$.

The excellent work by Schäfer and Korbinian [39] gave us the tools to initiate the work.

At each generation the algorithm estimates the vector of means and the variance-covariance matrix. This means that the number of parameters to estimate is quadratic in the problem size: $p = 2n + \binom{n-1}{2}$. Thus, it is easy to get to the "small n, large p" scenario. The $EMNA_{global}$ algorithm [13] computes the MLE estimates of these parameters. Another possibility is to use the unbiased estimator of the covariance matrix. However, it is well-known that both are statistically inefficient, because for medium to small sample sizes they are far from being optimal estimators for recovering the population covariance.

The merit of the shrinkage method is that it improves the efficiency and accuracy of the estimation and yields a covariance matrix that is well-conditioned and positive definite, which is importante to compute its inverse.

The idea of the shrinkage estimation is simple. We now follow [39]. Assume we have an unrestricted high-dimensional model and a lower dimensional restricted submodel. By fitting each of the two different models to the observed data associated estimates are obtained. Clearly, the unconstrained estimate will exhibit a comparatively high variance due to the larger number of parameters that need to be fitted, whereas its low-dimensional counterpart will have lower variance but potentially also considerable bias as an estimator of the true unrestricted model.

Instead of choosing between one of these two extremes, the linear shrinkage approach suggests combining both estimators in a weighted average

$$U^* = \lambda R + (1-\lambda)U$$

Compare this equation with (8.4)!

In addition, it is possible to choose the parameter λ in a data-driven fashion by explicitly minimizing a risk function.

$SEDA_{mn}$ uses the shrinkage estimates based on the optimal λ. From an evolutionary point of view it is a simple algorithm. The current implementation uses truncation selection and elitism. It is like the $EMNA_{global}$ [13], but with a different method for variance and covariance estimation. A simplified pseudo-code of the current implementation is shown in Algorithm 2.

In what follows we present some experimental results to illustrate the power of $SEDA_{mn}$ for expensive optimization.

Algorithm 2. $SEDA_{mn}$ - Srinkage Estimation of Distribution Algorithm for the Multivariate Normal (current implementation).

Set $t \leftarrow 1$
Randomly generate $N \gg 0$ configurations
while stop criteria are not valid **do**
- Evaluate the population with the fitness function $f(x)$
- Using truncation selection construct the selected set, SS.
- Compute the mean, μ, and the covariance matrix, S, of SS:
$s^*_{ii} = s_m \lambda^*_{var} + (1 - \lambda^*_{var}) s_{ii}$
$s^*_{ij} = r_{ij} min(1, max(0, 1 - \lambda^*)) \sqrt{s_{ii} s_{jj}}$
where:
$\lambda^*_{var} = (\sum_{k=1}^{p} \widehat{Var}(s_{kk})) / \sum_{k=1}^{p} (s_{kk} - s_m)^2$,
$\lambda^* = \sum_{k \neq l} \widehat{Var}(r_{kl}) / \sum_{k \neq l} r^2_{kl}$
{s_m denotes the median of the empirical variances and the coefficients s_{ii} and r_{ij} denote the empirical variance (unbiased) and correlation, respectively. See [39] for the computation of $\widehat{Var}(s_{ij})$.}
- Generate N new configurations (the new population) from the multivariate normal with parameters μ and S.
- Set $t \leftarrow t + 1$
end while

We study our algorithm with the well-known benchmark functions: Sphere, Rosenbrock, Ackley, Griewangk and Rastrigin. The authors of [13] (page 183) use these functions to investigate the $EMNA_{global}$ and six other EDA algorithms. They present results for $n = 10$ and $n = 50$.

Table 8.11 shows impressive results for 50 variables. For $SEDA_{mn}$ we set a population size equal to 25 and present averages over 30 runs. The best fitness value reported in [13] for each function is shown in the second column whereas the third column shows a lower bound in the number of function evaluations. The next two columns contain the same information for $SEDA_{mn}$. Finally, the last column is the best fitness value obtained with the very same $SEDA_{mn}$ algorithm, but with MLE

Table 8.11 Comparison of $SEDA_{mn}$ with $EMNA_{global}$ and the other algorithms reported in [13]

$F(x)$	Best Fv[a]	#Evals[a]	Best Fv[b]	#Evals[b]	Best Fv[c]
Sphere	10^{-6}	> 200000	10^{-8}	< 3000	> 1000
Rosenbrock	48.7	> 270000	48.5	< 1500	> 150
Ackley	10^{-6}	> 280000	10^{-8}	< 2500	> 1.2
Griewangk	10^{-6}	> 170000	10^{-8}	< 2500	$> 6x10^5$

[a] Best result from [13]. ($EMNA_{global}$ and others); [b] $SEDA_{mn}$ with shrinkage; [c] $SEDA_{mn}$ with MLE covariance estimation.

Table 8.12 Scaling of $SEDA_{mn}$. Averages over 20 runs. $N = 50$ and $error = 10^{-6}$

$F(x)$	$n = 50$	$n = 100$	$n = 500$
Ackley	5060 ± 96	6720 ± 103.2	11880 ± 97.7
Griewangk	5525 ± 110	8140 ± 251.4	14050 ± 282.8
Rastrigin	6575 ± 761.4	8500 ± 1149	13540 ± 1069

covariance estimation and the maximum number of function evaluations that appears in the corresponding fifth column.

Now, and just to get an idea of the scaling of $SEDA_{mn}$, we fix the population size $N = 50$ and report in Table 8.12 the average number of function evaluations needed for getting an error of 10^{-6} in 20 runs. We show the results for 50, 100 and 500 variables and the functions: Ackley, Griewangk and Rastrigin. Note the linear dependence.

In our opinion, the experiments of this section clearly show that $SEDA_{mn}$ is a new powerful EDA algorithm that offers us a significant reduction in the number of function evaluations with respect to existing EDAs.

8.6 Summary and Conclusions

We have presented a collection of methods that utilize the probabilistic models built by the EDAs for enhancing the efficiency of these algorithms. These methods make EDAs more suitable for expensive black-box optimization. Our main goal was to send two messages to those scientists in the expensive optimization research community who are interested in experimenting with EDAs: 1) EDAs have an enormous potential to become a powerful tool for this kind of problems. 2) There are many research opportunities for improving current EDAs most of which depend on our ability to use the huge and increasing set of methods of modern statistical sciences and machine learning.

We have shown how the regularization of the parameters of the probabilistic models used in EDAs can yield dramatic improvements in efficiency. We presented the design of the Linear Entropic Mutation, which is a mutation operator that do not destroy the learned distributions and is based on global information. This technique, increases the entropy of the probabilistic models and produces higher percentage of success rates and fewer function evaluations.

Of particular interest is the proposal of a new class of EDAs based on shrinkage estimation, SEDAs, which is capable of an efficient and accurate model building with small populations. We presented a brief introduction of the SEDAs by means of a detailed discussion of one of its members: $SEDA_{mn}$ (an algorithm that uses the multivariate normal distribution). We have presented numerical simulations with popular benchmark functions that show impressive results for the new algorithm.

Another key contribution of the chapter is the development of a new surrogate fitness model based on the search distributions. With this method the evolution starts in the fitness landscape, switches to the log-probability landscape of the model and then backtracks to continue in the original landscape if the optimum is not found. We have presented numerical results that show a reduction in the number of fitness evaluations of about 45%.

For the sake of completeness the chapter reviews other techniques for improving the sampling efficiency of EDAs. In all cases the theoretical presentation is accompanied by numerical simulations that support the main claims of the chapter.

At this point it is worth noting that the main goal of the chapter is similar to that of the work [38]. Both highlight the role of the search distributions as an important source of information for developing EDAs-efficiency-enhancement strategies. The authors of that chapter presented a classification of these strategies into four categories: hibridization, parallelization, time continuation/utilization and evaluation-relaxation. However, their main contribution was the provision of examples of two principled efficiency-enhancement techniques (1) An evaluation-relaxation scheme where they build an endogenous fitness-estimate model using the probabilistic models built by the Bayesian optimization algorithm, and (2) a time-continuation scheme where they develop a scalable mutation operator in the extended compact GA.

Our work is different from [38] in several aspects. We just mention four of them.

1. We include a new category into the classification: *Regularization*.
2. Our PE (evaluation-relaxation) scheme is different. The evolution in log-probability landscapes with backtracking is introduced. The computation of the fitness surrogate is straightforward and inexpensive.
3. Linear Entropic Mutation is presented as a global mutation operator. This is in contrast with other approaches that rely on local information.
4. The new class of Shrinkage EDAs, and one of its members $SEDA_{mn}$ are introduced.

Acknowledgements. We would like to thank Marta Soto and Julio Madera for our previous joint work. It was reviewed in Sect. 8.3 and Sect. 8.4.1.

We also thank the editors and the anonymous referees for their useful comments and recommendations.

Appendix

B-Functions: A Random Class of Benchmark Functions

The random Boltzmann function model was introduced in [32] and later extended in [33] and investigated in [30], where it is called B-function model. The model allows the explicit codification of probabilistic information, which is very convenient for the study of Estimation of Distribution Algorithms.

The following definition formally introduces the B-functions.

Definition 4. *(B-functions). Assume $\eta > 0$. Let $p(x)$ be any unimodal probability mass function and let x_{mpc} be its most probable configuration (mpc). The parametric function*

$$BF_{p,\eta}(x) = \frac{1}{\eta} log\left(\frac{p(x_{mpc})}{p(x)}\right), \tag{8.5}$$

called here B-function, is an additively decomposable unimodal non-negative function with minimum at x_{mpc}.

The above definition can be modified to the deal with multimodal and real valued problems. It also says that whenever we have a distribution we can construct a B-function. For example, the Alarm B-function is built by using as $p(x)$ the famous ALARM Bayesian network [1]. We also have shown elsewhere how to build such a distribution given collections of certain types of probabilistic constraints.

The following properties tell us why B-functions are an excellent benchmark for evolutionary optimization.

- The minimum of a B-function is always zero, thus the stopping criterion of the optimization algorithm is easy to implement.
- The computation of the most probable configuration (discrete variables), which is necessary for the definition of the function, can be accomplished in polynomial time for a large class of distributions [28, 45].
- The random generation of B-function instances (graph and parameters) can be accomplished efficiently [32].
- A naming convention to facilitate referencing of some B-function instances and subclasses can be easily implemented. Alternatively, less standard B-functions instances and subclasses can be distributed as files.
- It is straightforward to control problem size, structural and parametric complexity to test scalability.
- There is no need to construct functions by concatenating small subfunctions.

In [33] we introduced a naming mechanism (and a program available from the author) to facilitate working with and referencing to certain subclasses of B-functions. We show it here for the case of boolean polytree B-functions.

$$BF2Bn_2sn_3 - d_1d_2d_3d_4[n_4] \tag{8.6}$$

The above notation stands for a function with n_2 boolean variables. The dependence structure is given by a polytree (restricted Bayesian network) with maximum number of parents equal to n_3. The digits $d_1, \ldots, d_4$ have the following meaning. The mutual information of any pair of adjacent variables in the dependency graph of the function lies on the interval $0.1 * [d_1, d_2]$. The digits d_3 and d_4 constraint the univariate entropies. In fact, the univariate probabilities lie in the interval $[p_{min}, p_{max}] = 0.1 * [d_3, d_4] + 0.05$ or in $[1 - p_{max}, 1 - p_{min}]$. Finally, the optional parameter n_4, which is a natural number not exceding $\left(10^9 - 1\right)$, is a random seed that

determines a unique B-function instance. In other words, if n_4 is given it determines a random function instance, otherwise the expression denotes a random subclass.

To clarify the above statements, consider the following examples of B-functions definitions.

BF2B30s4-1245. the random subclass of binary polytree functions that have 30 variables, each with a maximum of four parents. Besides, each pair of adjacent variables has mutual information in the interval $[0.1, 0.2]$ and the univariate probabilities are bounded by $[0.45, 0.55]$.

BF2B30s4-124595012929. this is an instance of the above class.

BF2B100s0-1301. the graph has no edges. The fields containing the mutual information bounds are senseless, thus their values are ignored.

References

1. Beinlich, I., Chavez, H.R., Cooper, G.: The ALARM Monitoring System: a Case Study with two Probabilistic Inference Techniques for Belief Networks. In: Artificial Intelligence in Medical Care, pp. 247–256 (1989)
2. Blanco, R., Lozano, J.A.: Empirical comparison of Estimation of Distribution Algorithms in combinatorial optimization. In: Larrañaga, P., Lozano, J.A. (eds.) Estimation of Distribution Algorithms. A New Tool for Evolutionar y Computation, Kluwer Academic Publishers, Dordrecht (2002)
3. Brown, L.E., Tsamardinos, I., Aliferis, C.F.: A comparison of novel and state-of-the-art polynomial bayesian network learning algorithms. In: AAAI, pp. 739–745 (2005)
4. Chiba, K., Jeong, S., Shigeru, O., Morino, H.: Data mining for multidisciplinary design space of regional-jet wing. In: Proceedings the 2005 IEEE Congress on Evolutionary Computation CEC 2005, pp. 2333–23340 (2005)
5. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. In: Analyzing deception in trap functions. Foundations of Genetic Algorithms, vol. 2, pp. 93–108 (1993)
6. Eby, D., Averill, R.C., Punch, W.F.I., Goodman, E.D.: Evaluation of injection island ga performance on flywheel design optimization. In: Proceedings of the Third Conference on Adaptive Computing in Design and Manufacturing (1998)
7. Groppo, T.: Learning bayesian networks skeleton: A comparison between TPDA and PMMS algorithm. Master's thesis, Universite Claude Bernard Lyon I (2006)
8. Ireland, C.T., Kullback, S.: Contingency Tables with Given Marginals. Biometrika 55, 179–188 (1968)
9. Jaynes, E.T.: Information Theory and Statistical Mechanics. Physics Review 6, 620–643 (1957)
10. Jiroušek, R., Přeučil, S.: On the Effective Implementation of the Iterative Proportional Fitting Procedure. Comput. Statistics and Data Analysis 19, 177–189 (1995)
11. Kim, H.S., Cho, S.B.: An efficient genetic algorithm with less fitness evaluation by clustering. In: Proceedings of 2001 IEEE Conference on Evolutionary Computation, pp. 887–894 (2001)
12. Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2002)

13. Larrañaga, P., Lozano, J.A., Miqueles, T., E-Bengoetxea: Experimental Results in Function Optimization with EDAs in Continuos Domain. In: Larrañaga, P., Lozano, J.A. (eds.) Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2002)
14. Lauritzen, S.L.: Graphical Models. Oxford Press (1996)
15. Ledoit, O., Wolf, M.: Improved estimation of the covariance matrix of stock returns with an application to portfolio selection. J. Empir. Finance 10, 603–621 (2003)
16. Lewis, P.M.: Approximating Probability Distributions to Reduce Storage Requirements. Information and Control 2, 214–225 (1959)
17. Liang, K.H., Yao, X., Newton, C.: Evolutionary search of approximated n-dimensional landscapes. International Journal of Knowledge-Based Intelligent Engineering Systems 4(3), 172–183 (2000)
18. Lozano, J., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.): Towards a new Evolutionary Computation. Advances on Estimation of Distribution Algorithms. Studies in Fuzziness and Soft Computing. Springer, Heidelberg (2006)
19. Madera, J.: Hacia una Generación Eficiente de Algoritmos Evolutivos con Estimación de Distribuciones: Pruebas de (In)dependencia +Paralelismo. PhD thesis, Instituto de Cibernética, Matemática y Física, La Habana (in Spanish). Adviser: A. Ochoa (2009)
20. Madera, J., Ochoa, A.: Un EDA basado en aprendizaje Max-Min con escalador de colinas. Tech. Rep. 396, ICIMAF (2006)
21. Madera, J., Ochoa, A.: Algoritmos Evolutivos con Estimación de Distribuciones Bayesianas basados en pruebas de (in)dependencia. Tech. Rep. 474, ICIMAF (2008)
22. Mahnig, T., Mühlenbein, H.: Comparing the Adaptive Boltzmann Selection Schedule SDS to Truncation Selection. In: Third International Symposium on Adaptive Systems, ISAS 2001, Evolutionary Computation and Probabilistic Graphical Models, La Habana, pp. 121–128 (2001)
23. Meyer, C.H.: Korrektes Schließen bei Unvollständiger Information. PhD thesis, Fernuniversität Hagen (1998) (in German)
24. Mühlenbein, H., Mahnig, T.: Mathematical analysis of evolutionary algorithms for optimization. In: Proceedings of the Third International Symposium on Adaptive Systems, pp. 166–185 (2001)
25. Mühlenbein, H., Mahnig, T.: Evolutionary Optimization and the Estimation of Search Distributions. Journal of Approximate Reasoning 31(3), 157–192 (2002)
26. Mühlenbein, H., Paas, G.: From Recombination of Genes to the Estimation of Distributions I. Binary Parameters. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)
27. Mühlenbein, H., Mahnig, T., Ochoa, A.: Schemata, Distributions and Graphical Models in Evolutionary Optimization. Journal of Heuristics 5(2), 213–247 (1999)
28. Nilsson, D.: An Efficient Algorithm for Finding the M most Probable Configuration in Bayesian Networks. Statistics and Computing 2, 159–173 (1998)
29. Ochoa, A.: Linear Entropic Mutation (submitted for publication) (2009)
30. Ochoa, A.: The Random Class of B-functions (unpublished) (2009)
31. Ochoa, A., Soto, M.: Partial Evaluation in Genetic Algorithms. In: IEA/AIE 1997: Proceedings of the 10th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, pp. 217–222. Goose Pond Press, Atlanta (1997)

32. Ochoa, A., Soto, M.: Linking Entropy to Estimation of Distribution Algorithms. In: Lozano, J., Larrañaga, P., Inza, I., Bengoetxea, E. (eds.) Towards a new Evolutionary Computation. Advances on Estimation of Distribution Algorithms. Studies in Fuzziness and Soft Computing, pp. 1–38. Springer, Heidelberg (2006)
33. Ochoa, A., Soto, M.: On the Performance of the Bayesian Optimization Algorithm with B-functions. Tech. Rep. 383, ICIMAF (2006)
34. Ochoa, A., Höns, R., Soto, M., Müehlenbein, H.: A Maximum Entropy Approach to Sampling in EDA-the Single Connected case. In: Sanfeliu, A., Ruiz-Shulcloper, J. (eds.) CIARP 2003. LNCS, vol. 2905, pp. 683–690. Springer, Heidelberg (2003)
35. Ong, Y.S., Nair, P.B., Keane, A.J.: Evolutionary optimization of computationally expensive problems via surrogate modeling. American Institute of Aeronautics and Astronautics Journal 41(4), 687–696 (2003)
36. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The Bayesian optimization algorithm. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., var, V.H., Jakiela, M., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference GECCO 1999, Orlando, FL, vol. 1, pp. 525–532. Morgan Kaufmann Publishers, San Francisco (1999)
37. Pelikan, M., Sastry, K., Cant-Paz, E. (eds.): Scalable Optimization via Probabilistic Modeling. Studies in Computational Intelligence, vol. 33. Springer, Heidelberg (2006)
38. Sastry, K., Pelikan, M., Goldberg, D.E.: Efficiency Enhancement of Estimation of Distribution Algorithms. In: Pelikan, M., Sastry, K., Cant-Paz, E. (eds.) Scalable Optimization via Probabilistic Modeling. Studies in Computational Intelligence, vol. 33, pp. 161–186. Springer, Heidelberg (2006)
39. Schäfer, J., Strimmer, K.: A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics. Statistical Applications in Genetics and Molecular Biology 4(1) (2005)
40. Shannon, C.E.: A Mathematical Theory of Communication. Bell System Technical Journal 27, 379–423 (1948)
41. Smith, R.E., Dike, B., Stegmann, S.: Fitness inheritance in genetic algorithms. In: Proceedings of the 1995 ACM Symposium on Applied Computing, SAC 1995, pp. 345–350 (1995)
42. Soto, M.: Un estudio sobre los algoritmos evolutivos basados en redes bayesianas simplemente conectadas y su costo de evaluación. PhD thesis, Instituto de Cibernética, Matemática y Física, La Habana (in Spanish). Adviser: A. Ochoa (2003)
43. Soto, M., Ochoa, A.: A Factorized Distribution Algorithm based on polytrees. In: Congress on Evolutionary Computation, CEC 2000, California, pp. 232–237 (2000)
44. Soto, M., Ochoa, A., Rodríguez-Ojea, L.: An Empirical Stuty on Oversampled Selection in Estimation of Distribution Algorithms. Tech. Rep. 494, ICIMAF (2008)
45. Yanover, C., Weiss, Y.: Finding the M most Probable Configurations in Arbitrary Graphical Models. In: Thrun, S., Saul, L., Schollkopf, B. (eds.) Advances in Neural Information Processing Systems, vol. 16. MIT Press, Cambridge (2004)
46. Yunpeng, C., Xiaomin, S., Peifa, J.: Probabilistic Modeling for Continuos EDA with Boltzmann Selection and Kullback-Liebler Divergence. In: Proceedings of GECCO Conference (2006)
47. Zhou, Z.Z., Ong, Y.S., Nair, P.B., Keane, A.J., Lum, K.Y.: Combining global and local surrogate models to accelerate evolutionary optimization. IEEE Transactions on Systems, Man and Cybernetics-Part C 37(1), 66–76 (2007)

Chapter 9
On Similarity-Based Surrogate Models for Expensive Single- and Multi-objective Evolutionary Optimization

L.G. Fonseca, H.J.C. Barbosa, and A.C.C. Lemonge

Abstract. In this chapter we propose a surrogate-assisted framework for expensive single- and multi-objective evolutionary optimization, under a fixed budget of computationally intensive evaluations. The framework uses similarity-based surrogate models and an individual-based model management with pre-selection. Instead of existing frameworks where the surrogates are used to improve the performance of evolutionary operators or as local search tools, here we use them to allow for an augmented number of generations to evolve solutions. The introduction of the surrogates into the evolutionary cycle is controlled by a single parameter, which is related with the number of generations performed by the evolutionary algorithm. Numerical experiments are conducted in order to assess the applicability and the performance in constrained and unconstrained, single- and multi-objective optimization problems. The results show that the present framework arises as an attractive alternative to improve the final solutions with a fixed budget of expensive evaluations.

9.1 Introduction

Several problems of interest in science and engineering are or can be advantageously formulated as optimization problems. However, modern problems have lead to the development of increasingly complex and computationally expensive simulation models. When the optimization algorithm involves the repeated use of expensive simulations to evaluate the candidate solutions, the computational cost of such

L.G. Fonseca
Natl Lab for Scientific Computing – LNCC, Petropolis RJ Brazil
e-mail: goliatt@lncc.br

H.J.C. Barbosa
Natl Lab for Scientific Computing – LNCC, Petropolis RJ Brazil
e-mail: hcbm@lncc.br

A.C.C. Lemonge
Department of Applied and Computational Mechanics,
Federal University of Juiz de Fora – UFJF, Juiz de Fora MG Brazil
e-mail: afonso.lemonge@ufjf.edu.br

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 219–248.
springerlink.com

applications can be excessively high. A trade-off between the number of calls to the expensive simulation model and the quality of the final solutions must often be established. As result, an improvement of the optimization process is necessary.

A possible solution to this problem is the use of a surrogate model, or metamodel. In this case, when evaluating candidate solutions in the optimization cycle, the computationally intensive simulation model is substituted by the surrogate model, which should be a relatively inexpensive approximation of the original model [24].

Genetic Algorithms (GAs) [21], inspired by Darwin's theory of evolution by natural selection, are powerful and versatile tools in difficult search and optimization problems. They do not require differentiability or continuity of the objective function, are less sensitive to the initialization procedures, and less prone to entrapment in local optima. However, they usually require a large number of evaluations in order to reach a satisfactory solution, and when expensive simulations are involved, that can become a serious drawback to their application.

The idea of reducing the computation time or improving the solutions performing less computationally expensive function evaluations, appeared early in the evolutionary computation literature [22]. It should be mentioned also that there are additional reasons for using surrogate models in evolutionary algorithms: (a) to reduce complexity [37], (b) to smooth the fitness landscape [61], (c) when there is no explicit fitness available, and (d) in noisy environments [25].

Several surrogate models, of varying cost and accuracy, can be found in the literature, such as polynomial models [36], artificial neural networks [17], Kriging or Gaussian processes [15], radial basis functions [30, 31], and support vector machines [27]. Of course such techniques can also be combined and used as an ensemble [32, 41].

Research in surrogate-assisted frameworks for solving problems with computationally expensive objective functions has been receiving increasing attention in the last few years [7, 14, 16, 18, 26, 43, 64].

In the evolutionary optimization context, the surrogate model is constructed from previously obtained solutions and used to evaluate new candidate solutions, avoiding expensive simulations. An interesting strategy, when a given budget of expensive evaluations is assumed, is to combine both exact and surrogate evaluations along the evolutionary process in order to allow for an extension in the number of generations, which can have a positive impact in the final result.

This chapter is focused on the use of a similarity-based surrogate model (SBSM) to assist evolutionary algorithms in solving single- and multi-objective optimization problems with a limited computational budget. Examples of similarity-based surrogate models are fitness inheritance [56], fitness imitation [24], and the nearest neighbor approximation model [4, 54].

In the surrogate-assisted optimization presented here, the individuals in the parent population (evaluated by the original function) are sequentially stored in a database, and then they are used to construct a surrogate model, based on similarity, which is used along the optimization procedure to perform extra (surrogate) evaluations, resulting in a larger number of generations.

This chapter is organized as follows. The optimization problem is described in Section 9.2. Section 9.3 presents the similarity-based surrogate models and the surrogate assisted evolutionary optimization algorithm, for single- and multi-objective optimization. The numerical experiments conducted are presented and discussed in Section 9.4, and finally the concluding remarks are given in Section 9.5.

9.2 The Optimization Problem

The optimization problems considered here can be written as

$$\begin{array}{c} \text{minimize } f_1(x), f_2(x), \ldots, f_{n_{obj}}(x) \\ \text{with } \; x = (x_1, \ldots, x_n) \in \mathscr{S} \\ \text{subject to } g_j(x) \leq 0, \quad j = 1, \ldots, n_i \\ x_i^L \leq x_i \leq x_i^U \end{array} \tag{9.1}$$

where $f_i(x)$ is the ith objective function to be minimized, n_{obj} is the number of objectives, n is the number of design variables, $\mathscr{S}$ is the search space bounded by $x^L \leq x \leq x^U$, and n_i is the number of inequality constraints. The feasible region is defined by $\mathscr{S}$ and the n_i inequality constraints $g_j(x)$.

We have multi-objective (MO) optimization when $n_{obj} \geq 2$. Single-objective (SO) optimization ($n_{obj} = 1$) is a special case of the formulation above. Also, in the absence of constraints ($n_i = 0$) we have the single- and multi-objective unconstrained optimization problems.

In MO optimization a set of solutions representing the tradeoff among the different objectives rather than an unique optimal solution is sought. This set of solutions is also known as the Pareto optimal set and these solutions are also termed noninferior, admissible, or efficient solutions [20]. The corresponding objective vectors of these solutions are termed nondominated and each objective component of any nondominated solution in the Pareto optimal set can only be improved by degrading at least one of its other objective components [58]. The concept of Pareto dominance and Pareto optimality will form the basis of solution quality. Pareto dominance is defined by

$$\begin{array}{l} x_1 \prec_P x_2 \; (x_1 \text{ Pareto-dominates } x_2) :\Leftrightarrow \\ \qquad \forall i \in \{1, \ldots, n_{obj}\} : f_i(x_1) \leq f_i(x_2) \wedge \\ \qquad \exists j \in \{1, \ldots, n_{obj}\} : f_j(x_1) < f_j(x_2). \end{array} \tag{9.2}$$

The Pareto optimal front (PFT) is the set of nondominated solutions such that $PFT = \{f_i(x^*) | \nexists f_j(x) \prec_P f_i(x^*),\ j \in \{1, \ldots, n_{obj}\}\}$.

9.3 Surrogate-Assisted Evolutionary Optimization

Surrogate modeling, or metamodeling, can be viewed as the process of capturing the essential features of a complex computer simulation (original evaluation function)

in a simpler, analytical model by providing an approximation of the input/output relation of the original model. The surrogate model should be simple, general, and keep the number of control parameters as small as possible [5]. Examples of such surrogates are the similarity-based surrogate models.

In this section we describe the similarity-based models, and the surrogate-assisted evolutionary algorithms for single- and multi-objective optimization.

9.3.1 Similarity-Based Surrogate Models (SBSMs)

Similarity-based surrogate models (SBSMs) can be classified as *lazy* learners [2] (and also memory-based learners) since that, in contrast to *eager* learning algorithms such as Neural Networks, Polynomial Response Surfaces, and Support Vector Machines, which generate a model and then discard the inputs, SBSMs simply store their inputs and defer processing until a prediction of the fitness value of a new individual is requested. Then they reply by combining their stored data (previous samples) using a similarity measure, and discard the constructed answer as well as any intermediate results.

Among the SBSMs one can find the Fitness Inheritance procedure, Fitness Imitation, and the nearest neighbors method. In the following subsections we present those approaches, and describe with details the nearest neighbor method, used here as a surrogate model.

9.3.1.1 Fitness Inheritance

The fitness inheritance procedure was first proposed by Smith et al [56], and since then has been applied in several problems [6, 12, 13, 49, 52, 63] and algorithms [38, 45]. In fitness inheritance, all the individuals in the initial population have their fitness value obtained via fitness function. Thereafter, the fitness of a fraction of the individuals in the subsequent populations is inherited from their parents. The remaining individuals are evaluated using the original fitness function (referred to as simulation model).

The inheritance procedure is described as follows. Given an individual x^h generated by evolutionary operators (crossover and mutation), from the parents x^{p_1} and x^{p_2}. The surrogate evaluation is given by:

$$\widehat{f}(x^h) = \begin{cases} f(x^{p_i}) & \text{if } d(x^h, x^{p_i}) = 0,\ i = 1 \text{ or } 2 \\ \frac{s(x^h,x^{p_1})f(x^{p_1})+s(x^h,x^{p_2})f(x^{p_2})}{s(x^h,x^{p_1})+s(x^h,x^{p_2})} & \text{otherwise} \end{cases} \tag{9.3}$$

where $s(x^h, x^{p_i})$ is the similarity between x^h and x^{p_i}. The assumption is that an offspring is similar to its parents and thus its fitness is assigned as the weighted average of the parents fitness.

In the inheritance procedure an entire simulation is replaced by a procedure with negligible computational cost, which may lead to great computational savings that grow with the rate of application of the inheritance technique and the cost of the

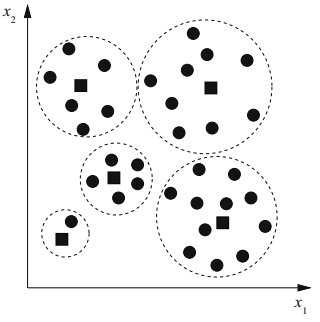


Fig. 9.1 Illustration of the Fitness Imitation procedure. The individuals inside the dotted circles belong to the same group. The representative individual, denoted by a black square, is evaluated by the exact function. The remaining individuals are evaluated by a surrogate model, their predicted fitness being calculated according to the distance to the representative individual

fitness function evaluation [8, 51]. In fact, the inheritance procedure may be orders of magnitude less expensive than the standard fitness evaluation. However, this approach introduces some noise in the search process and may adversely affect the final solution found [13].

9.3.1.2 Fitness Imitation

In Fitness Imitation [24], the individuals are clustered into several groups. Several clustering techniques can be used to perform this task [28]. Then, only the individual that represents its cluster is evaluated using the fitness function. The choice of the representative individual can be made either deterministically or randomly [35]. The fitness value of other individuals in the same cluster will be estimated from the representative individual based on a similarity measure. If a new individual to be evaluated does not belong to any cluster, it is evaluated by the original function. The term Fitness Imitation is used in contrast to Fitness Inheritance.

An illustration of the Fitness Imitation procedure is depicted in Figure 9.1. Examples of applications of this procedure can be found in [3, 28, 35].

9.3.1.3 Nearest Neighbors

The nearest neighbors surrogate model (k-NN) is a simple and transparent surrogate model where the approximations are built based on a set $\mathscr{D}$, which stores η individuals (samples).

The idea of using k-NN to assist an evolutionary algorithm was explored in [46, 47], where the aim was to reduce the number of exact function evaluations needed during the search. Here we use the surrogate to extend the generations, and to guide the search towards improved solutions.

Given an offspring x^h, the corresponding value $\widehat{f}(x^h) \approx f(x^h)$, to be assigned to x^h is

$$\widehat{f}(x^h) = \begin{cases} f(x^{\mathscr{I}_j}) & \text{if } x^h = x^{\mathscr{I}_j}, \text{ for some } j = 1,\ldots,\eta \\ \dfrac{\sum_{j=1}^{k} s(x^h,x^{\mathscr{I}_j})^u f(x^{\mathscr{I}_j})}{\sum_{j=1}^{k} s(x^h,x^{\mathscr{I}_j})^u} & \text{otherwise} \end{cases} \tag{9.4}$$

where $s(x^h,x^i)$ is a similarity measure between x^h and x^i, $\mathscr{I}_j$, $j = 1,\ldots,\eta$ is a list that stores the individuals in the set $\mathscr{D}$ most similar to x^h, k is the number of neighbors used to build the surrogate and u is set to 2.

The main advantages of the k-NN technique are that it is flexible, does not have severe restrictions, does not require any predefined functional form nor rely on any probability distribution, and the variables can be either continuous or discrete. Databases are also easy to maintain and updated when it is necessary to add or remove samples. Indeed, k-NN does not require a training procedure, and in each surrogate evaluation the database $\mathscr{D}$ must be ranked in order to determine the nearest neighbors.

The similarity measure used here is based on the Euclidean distance and it is given by

$$s(x^h,x^i) = 1 - \frac{d_E(x^h,x^i)}{d_E(x^U,x^L)}$$

where $d_E(x,y)$ is the Euclidean distance between x and y.

The nearest neighbors (and its variations) have been applied in two-dimensional interpolation [54], supervised learning [62], and recently in forest management planning [55].

9.3.2 Surrogate-Assisted Framework

Once a surrogate model has been chosen, there are many ways of introducing it into the evolutionary process. Some of them include: integrating GAs with surrogate approximations [40, 44] or landscape approximations [29], the use of surrogate-guided evolutionary operators [42], surrogate-assisted local search [33, 60], accelerating the optimization process using surrogates, pre-selection approaches [19, 39], multiple surrogates [1, 33, 50], and coevolution of fitness predictors [53].

In this chapter we introduce the surrogate models into the evolutionary cycle by means of a model management procedure [24] which, in each generation, uses in a cooperative way both surrogate and exact models, so that the evaluation of the population does not rely entirely on the surrogate model.

Maintaining a total of $N_{f,max}$ exact evaluations, surrogate model evaluations are introduced in the GA in increasing levels, by decreasing the parameter p_{sm}. The number of generations performed by the GA is given by $N_G = \frac{N_{f,max}}{p_{sm}\lambda}$. When $p_{sm} = 1$, all individuals are evaluated by the exact function, one has $N_G = N_{f,max}/\lambda$, and the standard GA is recovered. Indeed, as p_{sm} decreases, more surrogate evaluations are introduced into the evolutionary optimization process.

1: **procedure** Pre-selection (PS)
2: **if** $p_{sm} \neq 1$ **then**
3: **repeat**
4: Evaluate individual using $\widehat{f}$
5: $N_{\widehat{f}} = N_{\widehat{f}} + 1$
6: **until** all individuals in G_t evaluated
7: Rank G_t according to the surrogate model
8: **end if**
9: **repeat**
10: Evaluate individual using f
11: $N_f = N_f + 1$
12: **until** all p_{sm} best individuals in G_t evaluated

Fig. 9.2 Pre-selection (PS) management procedure. p_{sm} is the fraction of individuals evaluated by the original model, λ is the population size, f and $\widehat{f}$ are the original and surrogate functions, N_f the current number of exact evaluations and $N_{\widehat{f}}$ is the current number of surrogate evaluations

In the model management used here, only a fraction $0 < p_{sm} \leq 1$ of the population is evaluated by the time-consuming original model. We implement a pre-selection (PS) [19] strategy, where the surrogate model is used to decide which individuals will be evaluated by the original function. This procedure is described as follows: first, using evolutionary operators, λ individuals in the offspring population G_t are generated from λ parents in the parent population P_t. Then the offspring population G_t is entirely evaluated by the surrogate model and then ranked in decreasing order of quality. Based upon this rank, the $p_{sm}\lambda$ highest ranked individuals (according to the surrogate model $\widehat{f}$) are evaluated by the original model, and the remaining $\lambda - p_{sm}\lambda$ individuals in G_t maintain their objective function predicted by the surrogate model $\widehat{f}$. The PS model management procedure is shown in Figure 9.2.

In the PS model management it is not necessary that the surrogate model approximates the objective function closely. It is sufficient that the ranking of the individuals in the offspring population be similar to the ranking that would be obtained using the simulation model.

9.3.3 The Surrogate-Assisted Evolutionary Algorithm

The similarity-based surrogate-assisted GA for computationally expensive optimization problems, is shown in Figure 9.3. The developed algorithm will be referred to as SBSM-GA. The variant developed for single-objective optimization is named SBSM-SOGA while the multi-objective one is referred to as SBSM-MOGA. The differences between them are (i) the ranking procedures (line 5 and 12) and (ii) the parent population update procedure (line 13).

In the presented algorithm, we adopted the standard floating-point coding: each variable is encoded as a real number and concatenated to form a vector which is an individual in the population of candidate solutions. The following step is to randomly generate an initial population. Each individual has then one or more objective function values assigned to it and, in cases of constrained optimization, also a

1: **procedure** SBSM-GA
2: $t \leftarrow 0$; $N_f \leftarrow 0$; $N_{\widehat{f}} \leftarrow 0$; $\mathscr{D} \leftarrow \emptyset$
3: Generate an initial population P_t with λ individuals
4: Evaluate P_t using the simulation model f
5: Rank P_t in decreasing order of quality
6: Initialize the set $\mathscr{D} \leftarrow P_t$
7: **while** $N_f \leq N_{f,max}$ **do**
8: Select parents from P_t
9: Generate a population G_t from P_t
10: Apply the model management (Figure 9.2).
11: Update the set $\mathscr{D}$
12: Rank P_t in decreasing order of quality
13: Update parent population P_{t+1} from P_t and G_t
14: $t \leftarrow t+1$
15: **end while**

Fig. 9.3 Similarity-based surrogate-assisted GA (SBSM-GA). Pseudo-code for single- (SBSM-SOGA) or multi-objective (SBSM-MOGA) optimization. P_t is the parent population, G_t is the offspring population, λ is the population size, f and $\widehat{f}$ are the original and surrogate functions. $N_{f,max}$ is maximum number of exact evaluations, N_f is the current number of exact evaluations and $N_{\widehat{f}}$ is the number of surrogate evaluations

measure of constraint violation associated with it. The population is sorted in order to establish a "ranking". Individuals are then selected for reproduction in a way that better performing solutions have a higher probability of being selected. The genetic material contained in the chromosome of such "parent" individuals is then recombined and mutated, by means of crossover and mutation operators, giving rise to offspring which will form a new generation of individuals. Finally, the whole process is repeated until a termination criterion is attained.

Elitism is applied in the parent population update procedure (line 13): some individuals of the parent population are saved to the offspring population before the new parent population is created. In the single-objective version (SBSM-SOGA), the best ranked individual of the parent population P_t is copied to the offspring population G_t.

In single-objective constrained optimization ($n_i \neq 0$), we use a constraint handling technique presented in [10] to guide the search toward the (feasible) optimum. The individuals are ranked according to a pair-wise comparison procedure, where the following criteria are enforced:

1. when two feasible solutions are compared, the one with better objective function value is chosen,
2. when one feasible and one infeasible solutions are compared, the feasible solution is chosen, and
3. when two infeasible solutions are compared, the one with smaller constraint violation is chosen.

The constraint violation is given by $\sum_{j=1}^{n_i} \max(0, g_j(x))^2$.

The surrogate-assisted multi-objective GA (SBSM-MOGA) uses the operators from the Non-dominated Sorting Genetic Algorithm (NSGA-II) [11]. The multi-objective version of the algorithm differs from the single-objective version in the following aspects: (i) the ranking procedure, which uses the fast non-dominated sorting algorithm and the crowding comparison operator, and (ii) the elitism mechanism used in the parent population update procedure.

The ranking procedure that appears in lines 5 and 12 of the Figure 9.3 is replaced by the non-dominated sorting procedure [11], where the population is first partitioned by means of nondominated sorting and, then, a crowding comparison operator is employed by considering distances between individuals of the same rank. The update procedure shown in line 13 is performed over the union of the parent and offspring populations. The offspring population G_t is added to the parent population P_t and the combined population (of size 2λ) is ranked according to non-domination, then the highest ranked individuals are copied to the next generation.

The constraint handling technique for multi-objective optimization problems is based on the constraint-domination criteria [11], where feasible solutions have a better non-domination rank than any infeasible solution. A solution i is said to constraint-dominate a solution j, if any of the following conditions is true:

1. Solution i is feasible and solution j is not.
2. Solution i and solution j are both infeasible, but solution i has a smaller overall constraint violation.
3. Solution i dominates solution j

To improve the quality of the approximations in Eq. (9.4) the surrogate models are updated along the optimization process, by updating the set $\mathscr{D}$. In the SBSM-GA, all individuals exactly evaluated are recorded into the set $\mathscr{D}$, and when the maximum size η of the set is reached, the oldest individual is chosen to be replaced. As a result, one has a relatively small and updated sample set, as older individuals are discarded from $\mathscr{D}$ as the population evolves. The set size is equal to λ in the first generation (line 6 of the algorithm 9.3) and limited to η individuals along the evolutionary process.

In order to avoid convergence to false optima, and the need to re-evaluate the best solutions in each generation, after the ranking procedure (either for single- or multi-objective version), a sorting algorithm is applied in order to ensure that individuals evaluated by the original function are ranked highest in the population.

9.4 Computational Experiments

The algorithmic parameters for both SBSM-SOGA and SBSM-MOGA are summarized in Table 9.1.

We remark that, as described in Table 9.1, we have set a lower bound $p_{sm} = 1/\lambda$. For single-objective problems, we must have $p_{sm} \geq 1/\lambda = 1/40 = 0.025$, and $p_{sm} \geq 1/\lambda = 1/50 = 0.02$ for multi-objective problems. In the computational experiments of this section, we have set $p_{sm} \geq 0.05$.

Table 9.1 Algorithmic parameters setting for SBSM-GA (single- and multi-objective optimization)

Algorithmic Parameters	
Population size (λ)	Single-obj. optimization problems: $\lambda = 40$ Multi-obj. optimization problems: $\lambda = 50$
Representation	Floating-point coding: vectors of real numbers.
Operators	*Single-obj. opt. problems*: Uniform mutation, [34], Heuristic, One- and Two-point crossover, [23], Rank-based selection [57] and Elitism (best individual copied to the next generation) *Multi-obj. opt. problems*: Uniform mutation, Heuristic, One- and Two-point crossover, Rank-based selection (fast-non-dominated sorting and crowding distance [11]), and Elitism (parent and offspring population mixed and sorted in order to create the next generation)
Stop criterium	Maximum number of *exact* evaluations, given by $N_{f,max}$.
Crossover Probability (p_c)	$p_{c,heu} = 0.54$ (Heuristic), $p_{c,1p} = 0.18$ (One-point) and $p_{c,2p} = 0.18$ (Two-Point)
Mutation Rate (p_m)	$p_m = 0.02$
Database size (η)	$\eta = \{\lambda,\ 2\lambda,\ 5\lambda,\ 15\lambda\}$ or DPR=$(\eta/\lambda) = \{1,\ 2,\ 5,\ 15\}$
Database update	Replace the oldest individual. Only individuals evaluated by the original function can replace individuals in the database $\mathscr{D}$.
Surrogate Model	Nearest Neighbors (k-NN)
Number of Neighbors (k)	$k \in \{1, 2, 5, 10, 15\}$
Model Management	Individual-based Pre-Selection (PS) [19]. At each generation, the offspring population G_t is entirely evaluated by the surrogate model and ranked in decreasing order of quality. The $p_{sm}\lambda$ highest ranked individuals (according to the surrogate model $\widehat{f}$) are evaluated by the original model, and the remaining $\lambda - p_{sm}\lambda$ individuals in G_t maintain their objective function predicted by the surrogate model $\widehat{f}$.
Fraction p_{sm}	$p_{sm} \in [0.05, 1.00]$. The parameter p_{sm} defines the fraction of individuals evaluated by the original model: $p_{sm} = 1$ means the standard GA (no surrogates) with $N_G = N_{f,max}/\lambda$ generations. As we must ensure at least one individual evaluated by the original model in each generation, we have $p_{sm} \geq 1/\lambda$.
Performance Measures	The performance of the SBSM-GA is to be compared to the Standard GA ($p_{sm} = 1$). *Single-obj. opt. problems*: The value of the objective function. For constrained problems, also the number of runs leading to feasible final solutions. *Multi-obj. opt. problems*: Generational Distance [59], Maximum Spread [33] and Spacing [20].
Number of runs	50

DPR: Database size Population size Ratio, with DPR $= \frac{\text{Database size}}{\text{Population size}} = \frac{\eta}{\lambda}$

As the surrogate evaluations are introduced into the Standard GA, errors due the surrogate model evaluations are also introduced, which may adversely affect the quality of the final solutions. On the other hand, the extra surrogate evaluations allow for a longer period to search for improved solutions. There is a trade-off beetwen the noise introduced by the surrogate models and the beneficial impact in increasing the number of generations. We recall that, given a budget of $N_{f,max}$ exact evaluations, as the parameter p_{sm} decreases, the number of generations increases according to $N_G = N_{f,max}/p_{sm}\lambda$.

It is assumed that for complex real-world applications the cost of a surrogate model evaluation is negligible when compared to that of a simulation, hence total computational time will be only slightly increased due to the extra surrogate evaluations.

9.4.1 Single-Objective Optimization

In this section we show the results obtained for unconstrained and constrained problems, using the GA assisted with the k-NN surrogate model.

The single-objective minimization problems are shown in Table 9.2, and the constrained optimization problems considered are shown in the Table 9.3. For the constrained problems, the bounds for each parameter, in each function, are defined in the Table 9.4. More details of this set of constrained problems can be found in [48].

In problems with a large number of constraints (n_i), similarity-based surrogate models are computationally interesting, since they do not require a training procedure, leading to a simple and inexpensive way to estimate the constraints of the individuals in the population.

9.4.1.1 Effects of the Number of Neighbors

In this first experiment, we study the impact of increasing the number of neighbors, given a fixed database size (fixed DPR), in order to choose an appropriate neighborhood size. Under a fixed DPR=$\eta/\lambda = 2$. the experiments were conducted

Table 9.2 Single-objective minimization problems. The maximum number of simulations is $N_{f,max}$, the lower and upper bounds are respectively x^U and x^L, n is the number of design variables, and f^* is the optimal objective function value

#	Objective function	$N_{f,max}$	n	$[x^L, x^U]$	f^*
F_{01}	$\sum_{i=1}^{n} x_i^2$	1000	10	$[-5.12, 5.12]$	0
F_{02}	$\sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	1000	10	$[-100, 100]$	0
F_{03}	$\sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \frac{\cos x_i}{\sqrt{i}} + 1$	1600	10	$[-600, 600]$	0
F_{04}	$-20e^{-0.2\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}} - e^{\frac{\sum_{i=1}^{n} \cos(2\pi x_i)}{n}} + 20 + e$	1000	10	$[-32.768, 32.768]$	0
F_{05}	$\sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$	2000	10	$[-5.12, 5.12]$	0
F_{06}	$\sum_{i=1}^{n} i x_i^4 + U(0,1)$	1000	10	$[-4.28, 4.28]$	0
F_{07}	$\sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	2000	10	$[-10, 10]$	0
F_{08}	$\sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert}) - 418.982887272433n$	1000	10	$[-500, 500]$	0

Table 9.3 Constrained minimization problems – The number of design variables is n. The constraints read $g_j = g_j(x) \leq 0,\ j = 1,\ldots,n_i$

#	Objective function	Constraints	n
G_{01}	$5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$	$g_1 = 2x_1 + 2x_2 + x_{10} + x_{11} - 10$ $g_2 = 2x_1 + 2x_3 + x_{10} + x_{12} - 10$ $g_3 = 2x_3 + 2x_2 + x_{12} + x_{11} - 10$ $g_4 = -8x_1 + x_{10}$ $g_5 = -8x_2 + x_{11}$ $g_6 = -8x_3 + x_{12}$ $g_7 = -2x_4 - x_5 + x_{10}$ $g_8 = -2x_6 - x_7 + x_{11}$ $g_9 = -2x_8 - x_9 + x_{12}$	13
G_{02}	$\frac{\sum_{i=1}^{n}\cos^4(x_i) - 2\prod_{i=1}^{n}\cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} ix_i^2}}$	$g_1 = 0.75 - \prod_{i=1}^{n} x_i$ $g_2 = \sum_{i=1}^{n} x_i - 7.5n$	20
G_{04}	$5.3578547x_3^2 + 0.8356891x_1$ $+x_2 37.293239x_1 - 40792.141$	$g_1 = 85.334407 + 0.0056858x_2x_5 +$ $0.0006262x1x_4 - 0.0022053x_3x_5 - 92$ $g_2 = -85.334407 - 0.0056858x_2x_5 -$ $0.0006262x1x_4 + 0.0022053x_3x_50$ $g_3 = 80.51249 + 0.0071317x_2x_5 +$ $0.0029955x_1x_2 + 0.0021813x_3^2 - 110$ $g_4 = -80.51249 - 0.0071317x_2x_5 -$ $0.0029955x_1x_2 - 0.0021813x_3^290$ $g_5 = 9.300961 + 0.0047026x_3x_5 +$ $0.0012547x_1x_3 + 0.0019085x_3x_4 - 25$ $g_6 = -9.300961 - 0.0047026x_3x_5 -$ $0.0012547x_1x_3 - 0.0019085x_3x_420$	5
G_{06}	$(x_1 - 10)^3 + (x_2 - 20)^3$	$g_1 = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100$ $g_2 = (x_1 - 6)^2 - (x_2 - 5)^2 + 82.81$	2
G_{07}	$x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 +$ $(x_3 - 10)^2 + 4(x_4 - 5)^2 +$ $(x_5 - 3)^2 + 2(x_6 - 1)^2 +$ $5x_7^2 + 7(x_8 - 11)^2 +$ $2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$	$g_1 = -105 + 4x_1 + 5x_2 + 3x_7 + 9x_8$ $g_2 = 10x_1 - 8x_2 - 17x_7 + 2x_8$ $g_3 = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12$ $g_4 = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120$ $g_5 = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40$ $g_6 = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6$ $g_7 = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30$ $g_8 = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10}$	10
G_{08}	$\frac{\sin(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)}$	$g_1 = x_1^2 - x_2 + 1$ $g_2 = 1 - x_1 + (x_2 - 4)^2$	2
G_{09}	$(x_1 - 10)^2 + 5(x_2 - 12)^2 +$ $xqb_3^4 + 3(x_4 - 11)^2 +$ $10x_6^5 + x_6^2 + x_7^4 -$ $4x_6x_7 - 10x_6 - 8x_7$	$g_1 = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5$ $g_2 = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5$ $g_3 = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7$ $g_4 = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7$	7
G_{10}	$x_1 + x_2 + x_3$	$g_1 = -1 + 0.0025(x_4 + x_6)$ $g_2 = -1 + 0.0025(x_5 + x_7 - x_4)$ $g_3 = -1 + 0.01(x_8 - x_5)$ $g_4 = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333$ $g_5 = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4$ $g_6 = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5$	8

Table 9.4 Bound constraints for single-objective constrained optimization problems. The maximum number of simulations is $N_{f,max}$ and f^* is the optimal objective function value

Function	Bound constraints	$N_{f,max}$	f^*
G_{01}	$0 \leq x_i \leq 1\ (i = 1,\ldots,9)$, $0 \leq x_i \leq 100\ (i = 10,11,12)$, $0 \leq x_{13} \leq 1$	600	-15
G_{02}	$0 \leq x_i \leq 10\ (i = 1,\ldots,n)\ n = 20$	1200	-0.80355
G_{04}	$78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$, $27 \leq x_i \leq 45 (i = 3,4,5)$	6000	-30665.539
G_{06}	$13 \leq x_1 \leq 100$, $0 \leq x_2 \leq 100$	2400	-6961.81388
G_{07}	$-10 \leq x_i \leq 10 (i = 1,\ldots,10)$	1000	24.3062091
G_{08}	$0 \leq x_1, x_2 \leq 10$	8000	-0.095825
G_{09}	$-10 \leq x_i \leq 10 (i = 1,\ldots,7)$	800	680.6300573
G_{10}	$100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000 (i = 2,3)$, $10 \leq x_i \leq 1000 (i = 4,...,8)$	3000	7049.3307

for $k = \{1,2,5,15\}$ neighbors and the averaged fitness in 50 runs was used as performance measure.

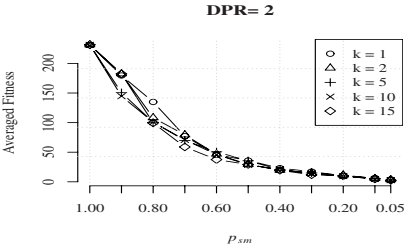
The neighborhood size affects the surrogate model in a way that small neighborhood leads to estimates very close to the data in the database $\mathscr{D}$, while a larger neighborhood tends to smooth the surrogate output, resulting in estimates close to the mean of the data in $\mathscr{D}$ [4].

The results for the SBSM-SOGA applied to the single objective optimization problems in Tables 9.2 and 9.3 for different values of p_{sm}, and using 1, 2, 5, 10, and 15 neighbors are shown in Figures 9.4 and 9.5. In Figure 9.5, for each test-problem, the average of the objective function in 50 runs is displayed. The average was calculated considering only the feasible runs, i.e. those producing a final solution which does not violate the constraints in Eq. (9.1).

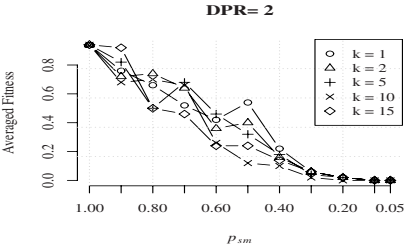
For the all unconstrained functions, except for F_{08}, as p_{sm} decreases, increasingly better results are obtained. For those functions, it is possible to use very small values of p_{sm}. In this set of experiments we set $p_{sm} = 0.05$, although we may use $p_{sm} > 1/40 = 0.025$, as described in Table 9.1. The results obtained for function F_{08}, show that improvements with respect to the Standard GA are obtained for p_{sm} values below a certain threshold value, and the maximum improvement (compared to the Standard GA) were obtained when $p_{sm} > 0.20$.

The same trend with respect to the number of neighbors and the parameter p_{sm} is observed for all unconstrained functions. We observe that the extra evaluations performed by the surrogate are beneficial to the evolutionary search, and improved results are obtained when the number of generations increases.

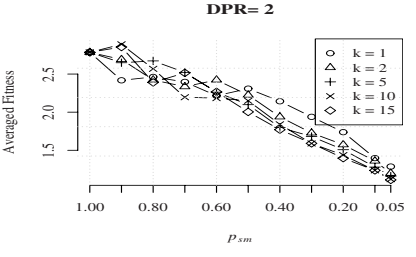
From the results obtained for function G_{08}, we can see that reducing p_{sm}, no longer improves the final results, which means that the noise introduced by the



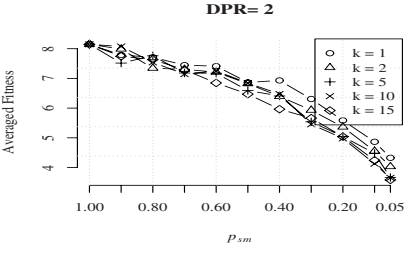
(a) F_{01}



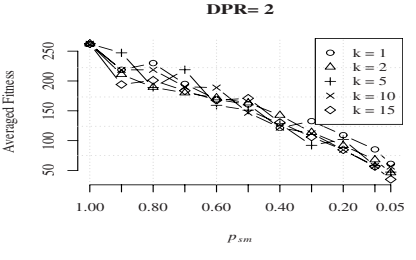
(b) F_{02}



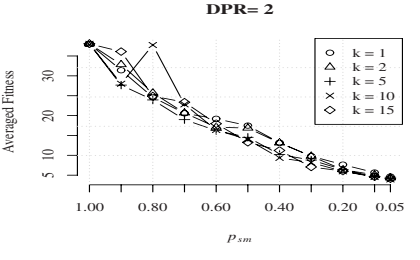
(c) F_{03}



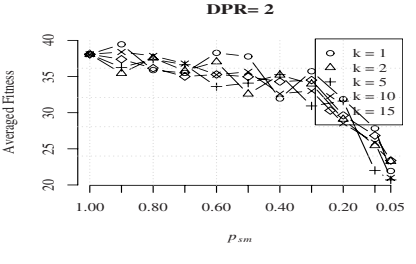
(d) F_{04}



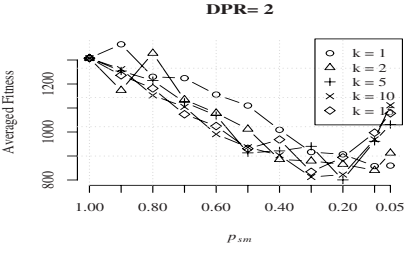
(e) F_{05}



(f) F_{06}

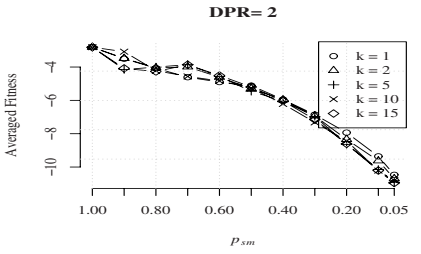


(g) F_{07}

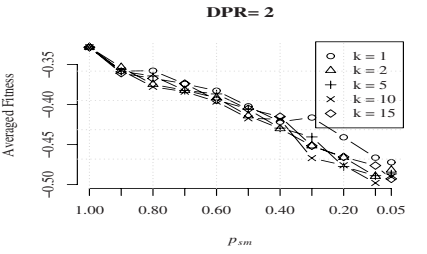


(h) F_{08}

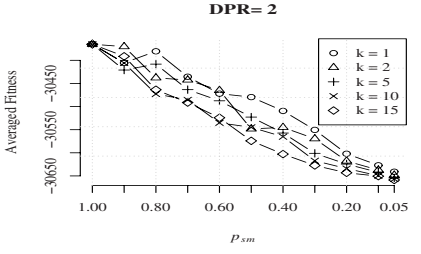
Fig. 9.4 Averaged Fitness for different values of p_{sm}, with DPR=2, using 1, 2, 5, 10, and 15 neighbors in the surrogate model shown in Eq. (9.4)



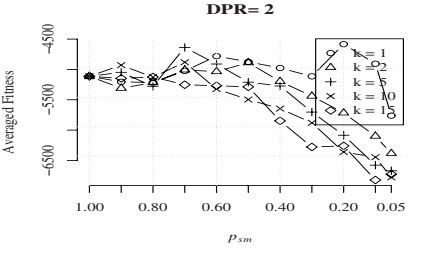
(a) G_{01}



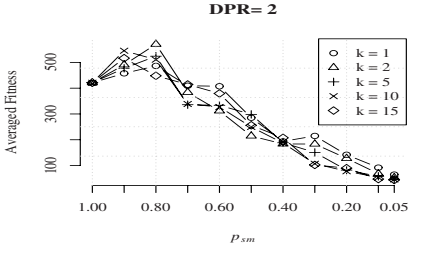
(b) G_{02}



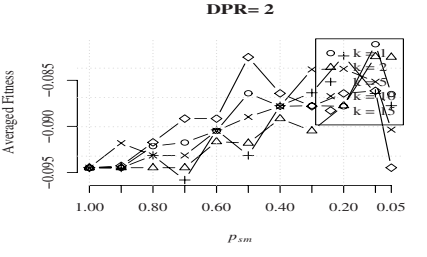
(c) G_{04}



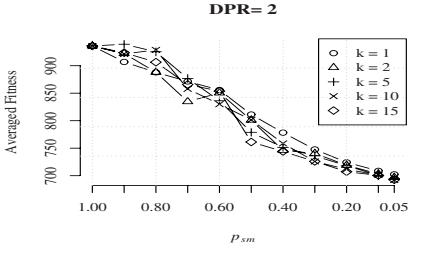
(d) G_{06}



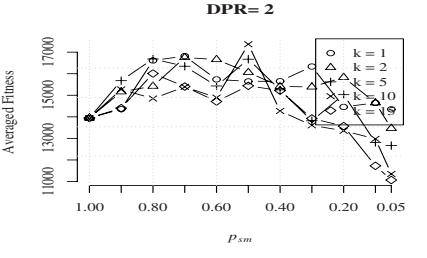
(e) G_{07}



(f) G_{08}



(g) G_{09}



(h) G_{10}

Fig. 9.5 Averaged Fitness for different values of p_{sm}, with DPR=2, using 1, 2, 5, 10, and 15 neighbors in the surrogate model shown in Eq. (9.4)

surrogate model affects the search in a negative way. Function G_{08}, corresponds to a complex landscape which could not be well approximated by the surrogate model. Although faster and simple, the k-NN surrogate model has limited capabilities to approximate complex mapping in $\Re^n$, which, as an inner-product space, allows for other calculus-based approximation. However, when the search occurs in a metric space, k-NN may be one of the few available alternatives.

As observed in function G_{08}, the constraints make the problem harder for the SBSM-SOGA, since more approximations are involved (objective functions and constraints) and the use of surrogates may lead the evolutionary process to poorer regions of the search space.

The results displayed in Figure 9.5, except for function G_{06}, and for G_{08} (where no improvements were obtained), show that the number of neighbors does not significantly affect the performance of the SBSM-SOGA for the set of functions considered here.

Table 9.5 shows the number of feasible runs for the SBSM-GA. The results were obtained using $k = 2$ neighbors and DPR=2 to build the surrogate in Eq. (9.4). We observe that the introduction of the surrogate does not affect the number of feasible runs, except in test-problem G_{06}, where a slightly decrease occurs. In G_{01} and G_{10}, the SBSM-GA increased the number of feasible runs.

Table 9.5 Constrained optimization problems – Number of runs that produce a final feasible solution with respect to the parameter p_{sm}. The results were obtained using 2 neighbors and DPR=2 to build the surrogate in Eq. (9.1)

p_{sm}	G_{01}	G_{02}	G_{04}	G_{06}	G_{07}	G_{08}	G_{09}	G_{10}
1	12	50	50	50	46	50	50	20
0.9	13	50	50	49	42	50	50	15
0.8	25	50	50	49	45	50	50	20
0.7	29	50	50	47	49	50	50	20
0.6	43	50	50	48	50	50	50	15
0.5	48	50	50	49	49	50	50	27
0.4	50	50	50	47	50	50	50	28
0.3	50	50	50	47	50	50	50	33
0.2	50	50	50	48	50	50	50	39
0.1	50	50	50	46	50	50	50	41
0.05	50	50	50	47	50	50	50	40

In frameworks that use surrogates as a local search tools or to enhance operators, the improvements are directly related to the surrogate models. In this set of experiments, the contribution of the surrogates to the evolutionary search is indirect: the surrogates allow for an extended number of generations (although with inexact evaluations), which provided the GA a longer period to evolve solutions.

9.4.1.2 Effects of the Database Size

In this section, a study of the impact of the database size on the evolutionary process is performed. Based on the experiments presented in the previous section, we set

the neighborhood size to $k = 2$ and we perform experiments for DPR=$\{1, 2, 5, 15\}$, corresponding to $\eta = \{1\lambda, 2\lambda, 5\lambda, 10\lambda, 15\lambda\}$ where $\eta = |\mathscr{D}|$.

Figures 9.6 and 9.7 display the results obtained by the SBSM-SOGA. We observe that for G_{06} larger values of η improve the results for smaller p_{sm}. The remaining test-problems are not affected by the database size. Except for G_{06}, we observe the same trend for all unconstrained and constrained functions, independent of the database size.

One can verify that the negative impact of the surrogate model persists for test problem G_{08}: the average results become worse as p_{sm} decreases, independently of the size of $\mathscr{D}$.

The results suggest that, for single-objective problems, a smaller database $\mathscr{D}$, combined with smaller values of p_{sm} are enough to improve the final solutions found by the SBSM-SOGA (when compared to the Standard GA, where $p_{sm} = 1$). However as the ruggedness/complexity of the optimization problem increases, and when constraints are involved (requiring more surrogate approximations), the performance may be not satisfactory, leading in some cases to deteriorated final solutions.

The results presented in sections 9.4.1.1 and 9.4.1.2 suggest to use a small value of the parameter p_{sm}. For SO problems, where one has no previous knowledge, we suggest as an initial trial $p_{sm} = 0.20$. Indeed, the results are indifferent to the database size, and we sugest a database size $\eta = 2\lambda$ (DPR=2).

9.4.2 *Multi-objective Optimization*

In this section we present and discuss the performance of the SBSM-MOGA when applied to constrained and unconstrained multi-objective problems.

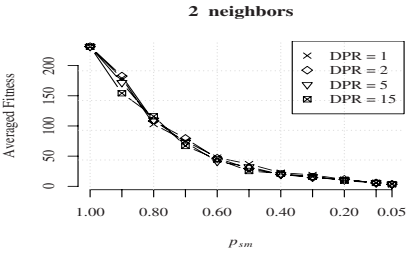
A total of 14 MO problems (8 unconstrained and 6 constrained) were collected [9] to study the impact of the surrogates into the SBSM-MOGA. Tables 9.6 and 9.7 show respectively the multi-objective unconstrained and constrained optimization problems, the bounds for each parameter, the constraints (for the constrained ones), the number of variables, and the maximum number of evaluations. Details can be found in [9].

In order to investigate the impact of the surrogate models in multi-objective optimization, we use as performance metrics the Generational Distance indicator (GD) [59], the Maximum Spread [33] and Spacing [20].

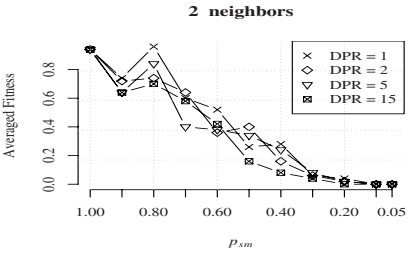
The GD indicator measures the gap between the evolved Pareto front (PFE) and the true Pareto front (PFT), given by

$$GD = \sqrt{\frac{1}{N_{PF}} \sum_{j=1}^{N_{PF}} d_j^2} \tag{9.5}$$

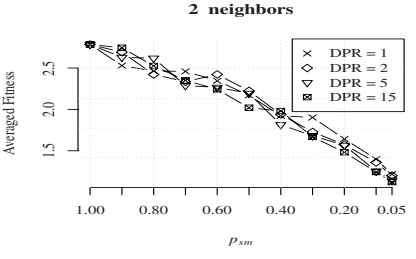
where N_{PF} is the number of individuals in PFT, d_j is the Euclidean distance (in the objective space) beetwen an individual j in PFE and its nearest individual in PFT. The generational distance in Eq. (9.5) measures the convergence to the true Pareto front, and lower values of GD are better.



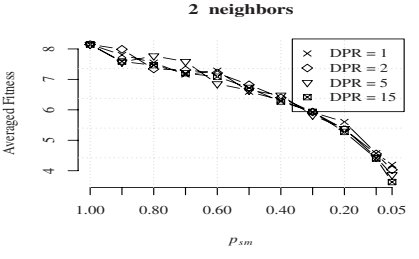
(a) F_{01}



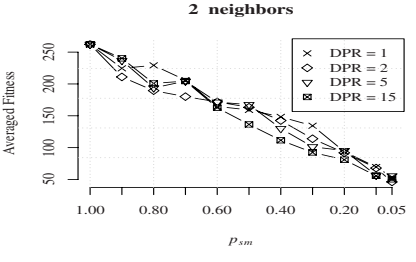
(b) F_{02}



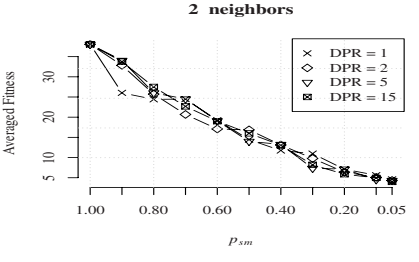
(c) F_{03}



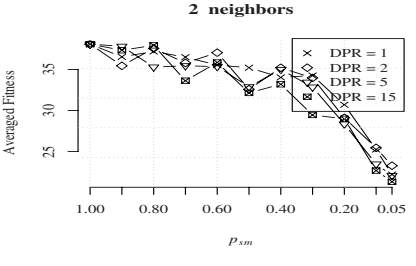
(d) F_{04}



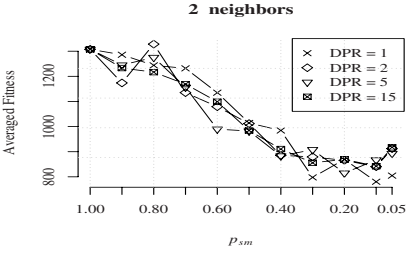
(e) F_{05}



(f) F_{06}



(g) F_{07}



(h) F_{08}

Fig. 9.6 Surrogate-assisted single-objective evolutionary unconstrained optimization

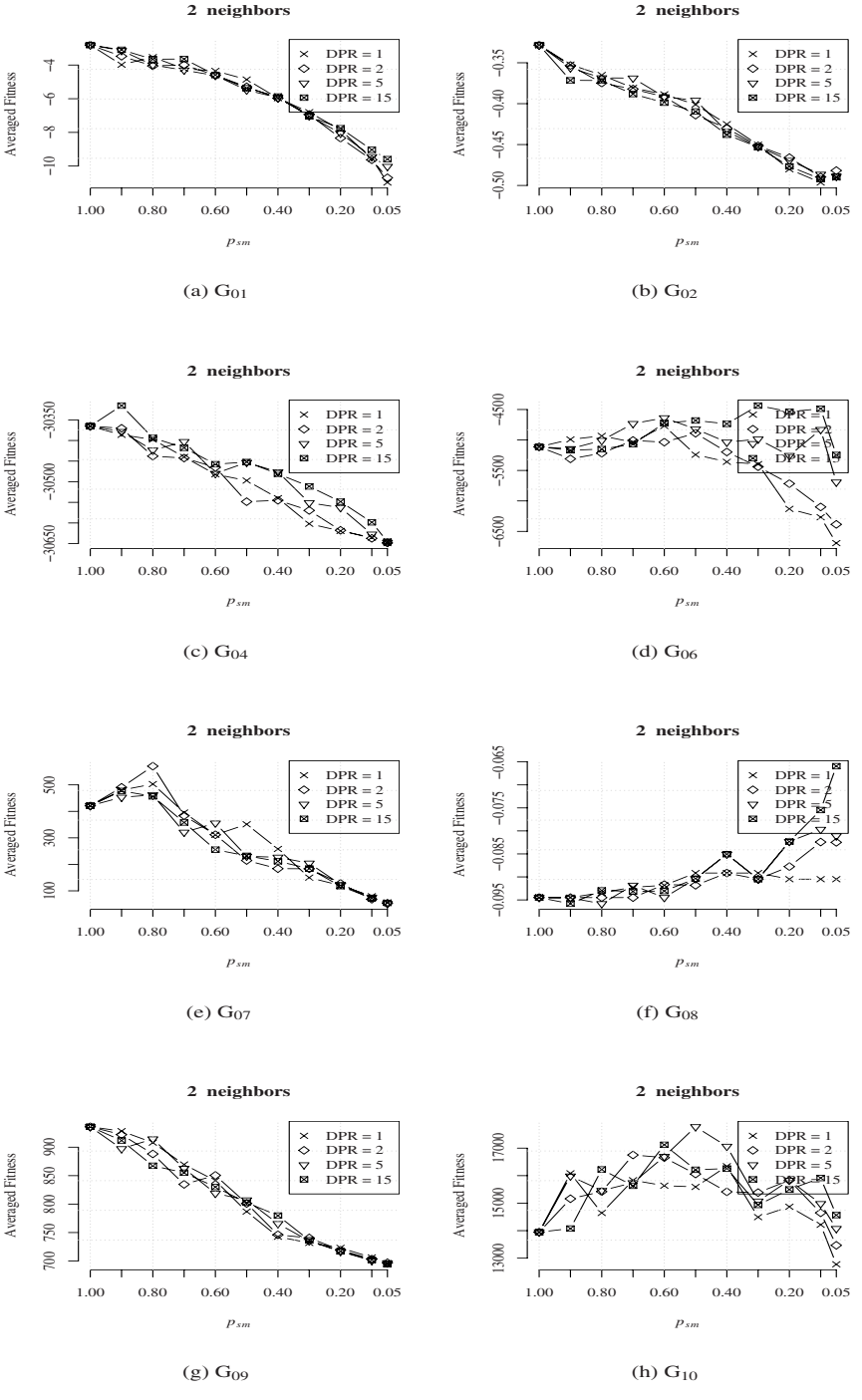


Fig. 9.7 Surrogate-assisted single-objective evolutionary constrained optimization

Table 9.6 Unconstrained multi-objective optimization problems. The maximum number of simulations is $N_{f,max}$, the lower and upper bounds are respectively x^U and x^L, and n is the number of design variables

#	Objective Functions	n	$[x^L, x^U]$	$N_{f,max}$
MF_{01}	$f_1(x) = x_1$ $f_2(x) = 1 - \sqrt{f_1(x)/g(x)}$ $g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	30	$[0,1]$	1000
MF_{02}	$f_1(x) = x_1$ $f_2(x) = g(x)\left[1 - (f_1(x)/g(x))^2\right]$ $g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	30	$[0,1]$	1000
MF_{03}	$f_1(x) = x_1$ $f_2(x) = g(x)\left[1 - \sqrt{f_1(x)/g(x)} - (f_1(x)/g(x))\sin 10\pi f_1\right]$ $g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$	30	$[0,1]$	1000
MF_{04}	$f_1(x) = x_1$ $f_2(x) = 1 - \sqrt{f_1(x)/g(x)}$ $g(x) = 1 + 10(n-1) + 9(\sum_{i=2}^{n}(x_i^2 - 10\cos 4\pi x_i)$	10	$x_1 \in [0,1]$, $x_i \in [-5,5]$ $i = 2,\ldots,10$	1000
MF_{05}	$f_1(x) = 0.5x_1x_2(1+g(x))$ $f_2(x) = 0.5(1-x_2)(1+g(x))$ $f_3(x) = 0.5(1-x_1)(1+g(x))$ $g(x) = 1000 + 100\sum_{i=3}^{n}[(x_i - 0.5)^2 - \cos 20\pi(x_i - 0.5)]$	12	$[0,1]$	2000
MF_{06}	$f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(x) = g(x)[1 - f_1(x)/g(x)^2]$ $g(x) = 1 + 9[\sum_{i=2}^{n} x_i/(n-1)]^{0.25}$	10	$[0,1]$	1000
MF_{07}	$f_1(x) = \cos\frac{\pi}{2}x_1\cos\frac{\pi}{2}x_2(1+g(x))$ $f_2(x) = \cos\frac{\pi}{2}x_1\sin\frac{\pi}{2}x_2(1+g(x))$ $f_3(x) = \sin\frac{\pi}{2}x_1(1+g(x))$ $g(x) = \sum_{i=3}^{n}(x_i - 0.5)^2$	12	$[0,1]$	1000
MF_{08}	$f_1(x) = \cos\frac{\pi}{2}x_1\cos\frac{\pi}{2}x_2(1+g(x))$ $f_2(x) = \cos\frac{\pi}{2}x_1\sin\frac{\pi}{2}x_2(1+g(x))$ $f_3(x) = \sin\frac{\pi}{2}x_1(1+g(x))$ $g(x) = 1000 + 100\sum_{i=3}^{n}[(x_i - 0.5)^2 - \cos 20\pi(x_i - 0.5)]$	12	$[0,1]$	1400

The Maximum Spread (MS) is used to measure how well the true Pareto front PFT is covered by the evolved Pareto front PFE. A larger value of MS reflects that a larger area of the PFT is covered by PFE. The MS is given as

$$MS = \sqrt{\frac{1}{n_{obj}}\sum_{i=1}^{n_{obj}}\left[\frac{\min(f_i^{max}, F_i^{max}) - \max(f_i^{min}, F_i^{min})}{F_i^{max} - F_i^{min}}\right]^2} \tag{9.6}$$

where f_i^{max} and f_i^{min} are the maximum and minimum of the i^{th} objective in the evolved Pareto front, respectively, and F_i^{max} and F_i^{min} the maximum and minimum of the i^{th} objective in the true Pareto front, respectively.

Table 9.7 Constrained multi-objective optimization problems. The maximum number of simulations is $N_{f,max}$, the lower and upper bounds are respectively x^U and x^L, and n is the number of design variables. The constraints are $g_j = g_j(x) \leq 0,\ j = 1,\ldots,n_i$

#	Objective functions	Constraints	n	Domain	$N_{f,max}$
MG_{01}	$f_1 = -2x_1 + x_2$ $f_2 = +2x_1 + x_2$	$g_1 = -x_1 + x_2 - 1$ $g_2 = +x_1 + x_2 - 7$	2	$0 \leq x_1 \leq 5$ $0 \leq x_2 \leq 3$	1000
MG_{02}	$f_1 = (x_1 - 2)^2 + (x_2 - 1)^2 - 2$ $f_2 = 9x_1 + (x_2 - 1)^2$	$g_1 = x_1^2 + x_2^2 - 225$ $g_2 = x_1 - 3x_2 + 10$	2	$[-20, 20]$	800
MG_{03}	$f_1 = x_1$ $f_2 = x_2$	$g_1 = 1 - x_1^2 - x_2^2 +$ $0.1\cos(16\arctan\frac{x_1}{x_2})$ $g_2 = (x_1 - 0.5)^2 +$ $(x_2 - 0.5)^2 - 0.5$	2	$[0, \pi]$	4000
MG_{04}	$f_1 = -25(x_1 - 2)^2 + (x_2 - 2)^2 +$ $(x_3 - 1)^2 + (x_4 - 4)^2 +$ $(x_5 - 1)^2$ $f_2 = \sum_{i=1}^{n} x_i^2$	$g_1 = x_1 + x_2 - 2$ $g_2 = 6 - x_1 - x_2$ $g_3 = 2 - x_2 + x_1$ $g_4 = 2 - x_1 + 3 * x_2$ $g_5 = 4 - (x_3 - 3)^2 - x_4$ $g_6 = (x_5 - 3)^2 + x_6 - 4$	6	$0 \leq x_1 \leq 10$ $0 \leq x_2 \leq 10$ $1 \leq x_3 \leq 5$ $0 \leq x_4 \leq 6$ $1 \leq x_5 \leq 5$ $0 \leq x_6 \leq 10$	800
MG_{05}	$f_1 = -x_1$ $f_2 = -x_2$ $f_2 = -x_3$	$g_1 = -1 + \sum_{i=1}^{n} x_i^2$	3	$[0, 1]$	1200
MG_{06}	$f_1 = \frac{1}{10}\sum_{i=1}^{10} x_i$ $f_2 = \frac{1}{10}\sum_{i=11}^{20} x_i$ $f_3 = \frac{1}{10}\sum_{i=11}^{30} x_i$	$g_1 = 1 - f_3 - 4f_1$ $g_2 = 1 - f_3 - 4f_2$ $g_3 = 1 - 2f_3 - f_1 - f_2$	30	$[0, 1]$	2000

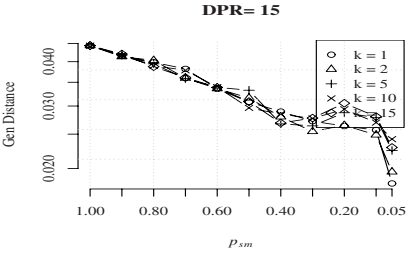
The metric of Spacing (S) shows how the nondominated solutions are distributed along the evolved Pareto front and is given as

$$S = \frac{1}{\hat{d}}\sqrt{\frac{1}{N_{PF}}\sum_{j=1}^{N_{PF}}(\hat{d} - d_j)^2}, \qquad \hat{d} = \frac{1}{N_{PF}}\sum_{k=1}^{N_{PF}} d_k \tag{9.7}$$

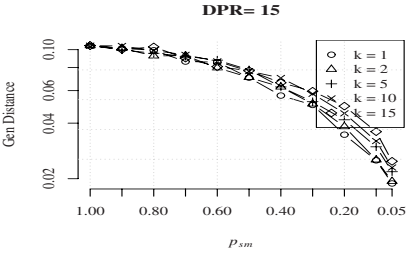
where N_{PF} is the number of individuals in PFT and d_i is the Euclidean distance (in the objective space) beetwen an individual i in the evolved Pareto front PFE and its nearest individual in the true Pareto front PFT.

9.4.2.1 Effects of the Number of Neighbors

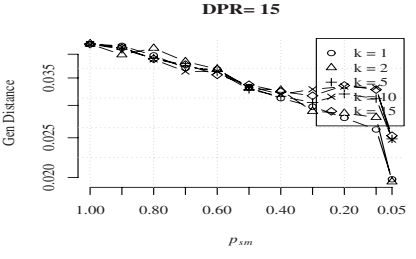
In this section we analyze the impact of the number of neighbors to the evolutionary search, given a fixed database size. According to the database replacement policy, the oldest individual is always chosen to be replaced. However, by removing solutions according to age, we may inevitably remove some important information. In order to alleviate this effect, we enlarge the training size for MO problems, and set the database size to $\eta = 15\lambda$, which corresponds to DPR=15. This value of



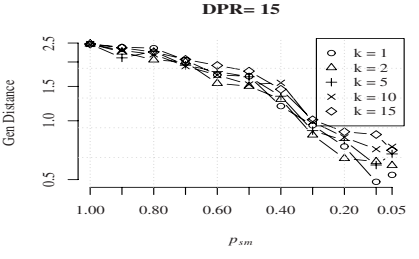
(a) MF_{01}



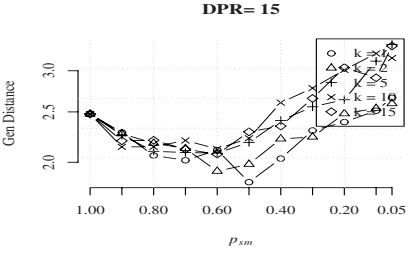
(b) MF_{02}



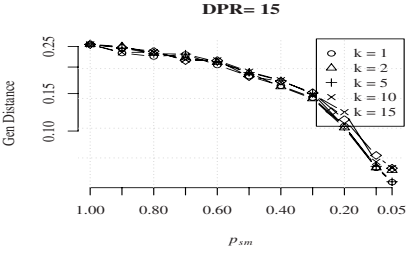
(c) MF_{03}



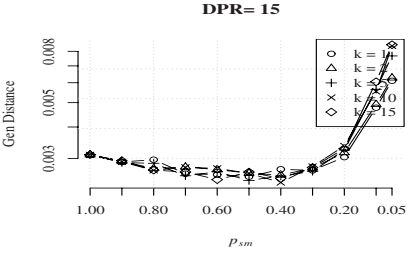
(d) MF_{04}



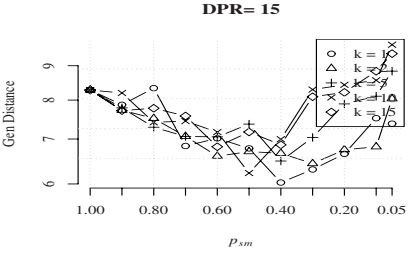
(e) MF_{05}



(f) MF_{06}

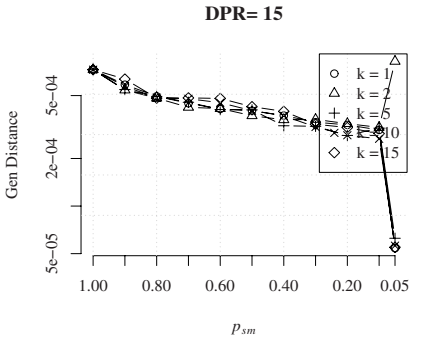


(g) MF_{07}

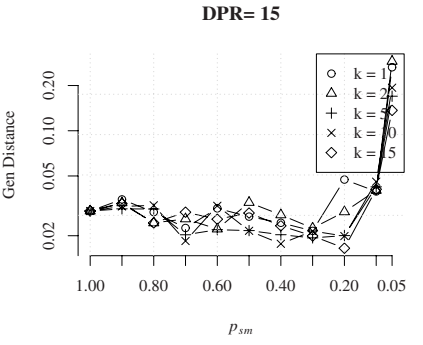


(h) MF_{08}

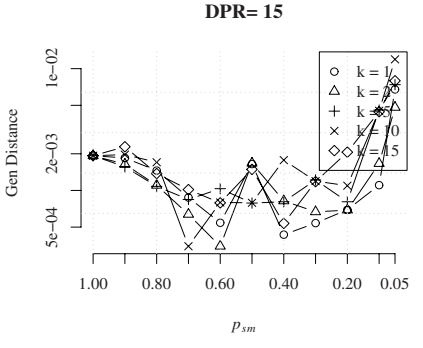
Fig. 9.8 Generational Distance (*GD*) indicator: surrogate-assisted multi-objective optimization using DPR=15 and $k = \{1, 2, 5, 10, 15\}$ neighbors



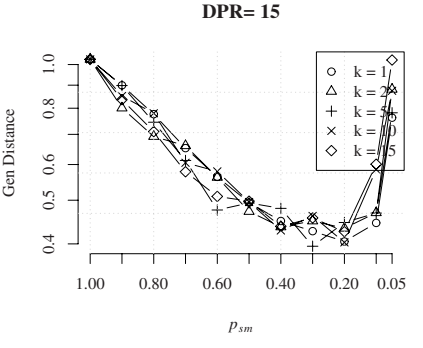
(a) MG_{01}



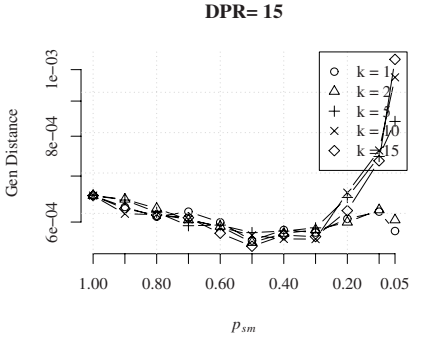
(b) MG_{02}



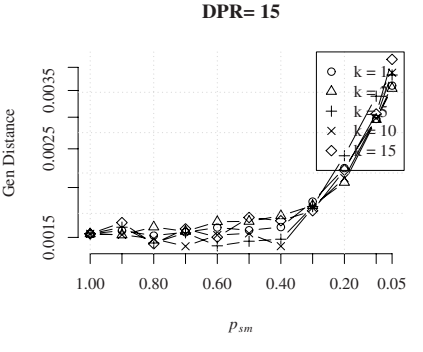
(c) MG_{03}



(d) MG_{04}



(e) MG_{05}



(f) MG_{06}

Fig. 9.9 Generational Distance (*GD*) indicator: surrogate-assisted multi-objective optimization using DPR=15 and $k = \{1, 2, 5, 10, 15\}$ neighbors

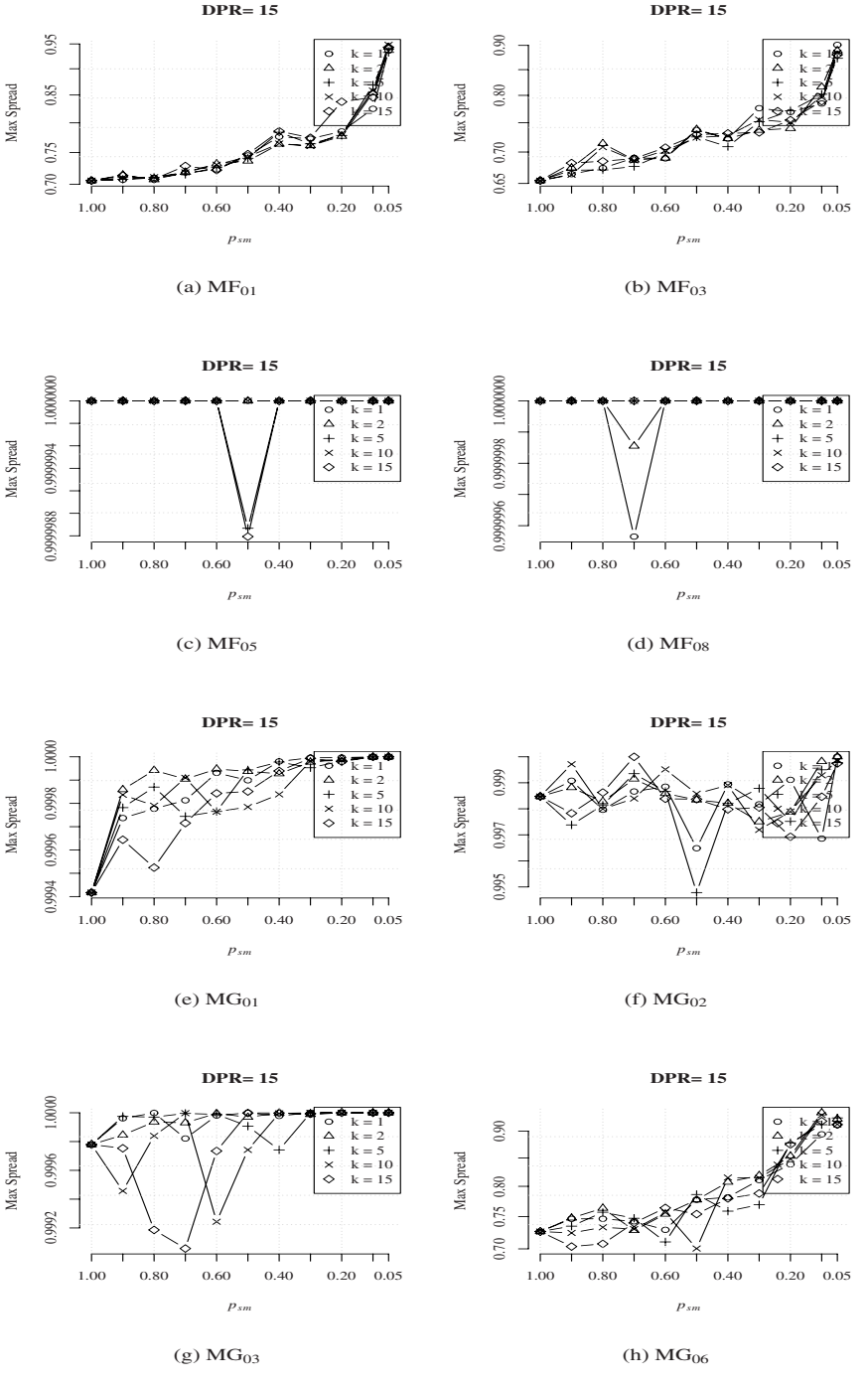
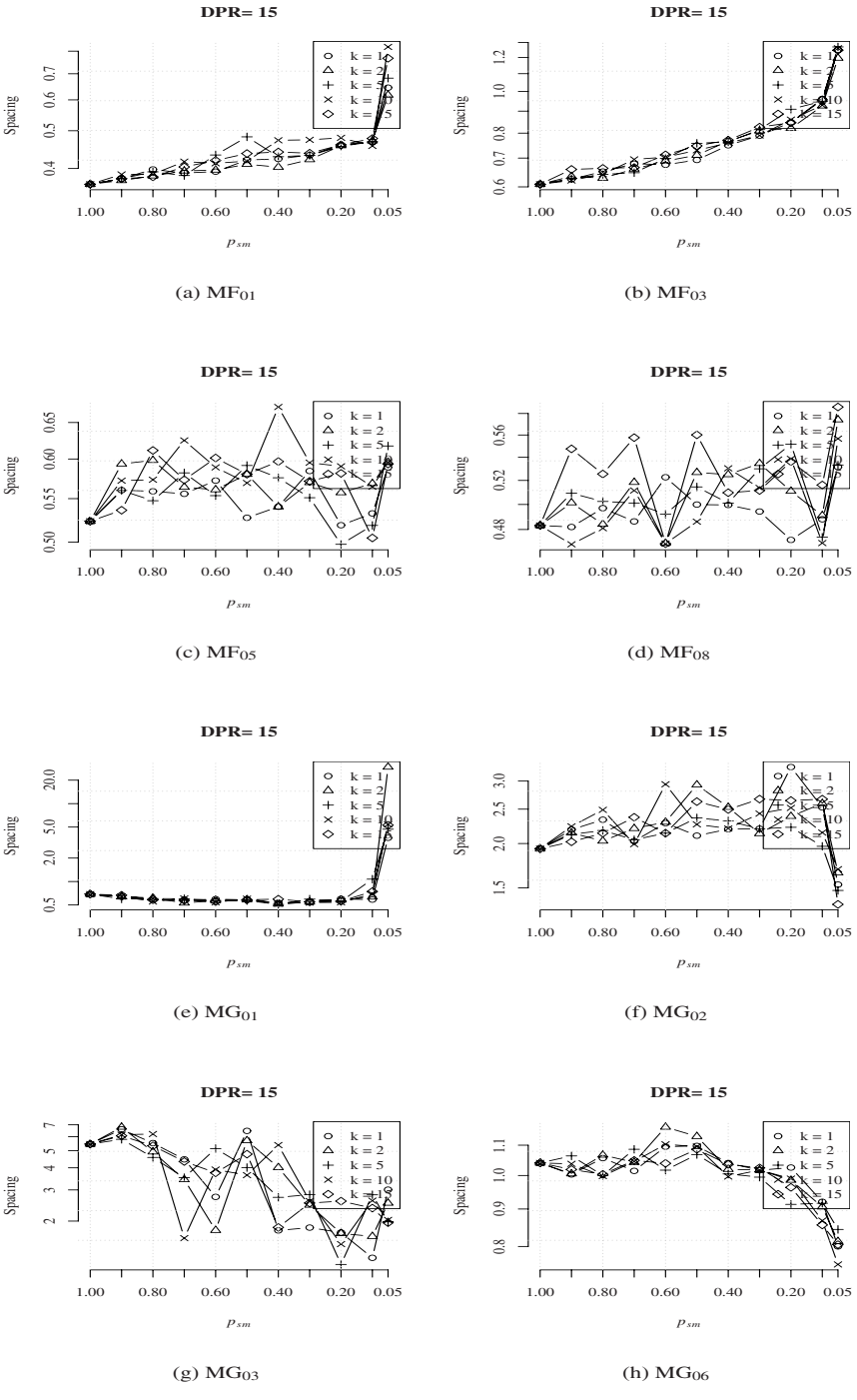


Fig. 9.10 Maximum Spread (*MS*): surrogate-assisted multi-objective optimization using DPR=15 and $k = \{1, 2, 5, 10, 15\}$ neighbors



(a) MF_{01} (b) MF_{03}

(c) MF_{05} (d) MF_{08}

(e) MG_{01} (f) MG_{02}

(g) MG_{03} (h) MG_{06}

Fig. 9.11 Spacing (S): surrogate-assisted multi-objective optimization using DPR=15 and $k = \{1, 2, 5, 10, 15\}$ neighbors

DPR results in storing information of at least 15 past generations, considering only individuals evaluated by the simulation model are stored in $\mathscr{D}$.

Figures 9.8 and 9.9 show the Generational Distance (*GD*) values, calculated for the (final) population at the end of the evolutionary process.

We observe that the SBSM-MOGA produced better results (compared to the Standard GA) depending on the values of the parameter p_{sm}. Except for the test-problems MF_{05}, MF_{07}, and MF_{08}, we observe that lower values of p_{sm} allow for final solutions closer to the true Pareto front. Also, the performance does not vary significantly as we change the number of neighbors used in the surrogate model. For function MG_{06} we observe that the surrogate model is not able to consistently help the GA in searching for improved solutions.

Figures 9.10 and 9.11 show the values of the Maximum Spread (*MS*) and Spacing (*S*) metrics, respectively, for a group of functions from those shown in Tables 9.6 and 9.7. From the results presented in Figure 9.10, we observe that, independently of the number of neighbors, smaller values of p_{sm} improve the performance of the SBSM-MOGA in the *MS* metric for MF_{01}, MF_{03}, MG_{01} and MG_{06}, and for MF_{05}, MF_{08}, MG_{02}, MG_{03} the *MS* is slightly affected when decreasing the parameter p_{sm}. Considering the Spacing metric, decreasing p_{sm} consistently improves the solutions in test-problems MF_{01}, MF_{03}, and MG_{01}.

9.5 Concluding Remarks

In this chapter we have proposed the introduction of a similarity-based surrogate model into a real-coded GA to assist the optimization of single- and multi-objective, constrained and unconstrained optimization problems, under a fixed computational budget.

We used the nearest neighbor approximation as a surrogate model, which is integrated into the evolutionary cycle by means of an individual-based evolution control where the surrogate is used to select individuals to be evaluated by the exact function according to a single parameter p_{sm}.

Instead of existing frameworks where the surrogates are used to improve the performance of evolutionary operators or as local search tools, here we use them to allow for an augmented number of generations to evolve solutions.

The tests performed so far support the following general conclusions:

Single-objective optimization: The augmented number of generations leads to improved solutions, when compared to the standard GA with the same number of expensive evaluations. Also, the number of neighbors does not affect in a significant way the final results, and a uniform trend is observed for unconstrained and constrained problems, as the parameter p_{sm} decreases. Also, the final results are not affected by the database size, which stores individuals previously evaluated by the simulation model.

Multi-objective optimization: For the set of multi-objective unconstrained optimization problems considered, small values of the parameter p_{sm} help to achieve

a better convergence to the true Pareto front, according to the performance metrics, and the results are not significantly affected by the number of neighbors used.

In the nearest neighbor approximation model no training procedure is required and the prediction involves finding the nearest neighbors in an archive of previously evaluated individuals. Under a fixed number of expensive simulations, the cost of the surrogate-assisted procedure is only slightly increased due to the negligible computational cost of the extra surrogate evaluations as the cost of the expensive simulation increases.

The framework presented here seems to be a simple and effective way to tackle single- and multi-objective unconstrained or constrained expensive optimization problems. Additionally, the proposed framework can be easily extended to other population-based metaheuristics, such as Differential Evolution, Ant Colony Optimization and Particle Swarm Optimization.

References

1. Acar, E., Rais-Rohani, M.: Ensemble of metamodels with optimized weight factors. Struct. Multidisc. Optim. 37(3), 279–294 (2009)
2. Aha, D.W.: Editorial. Artif. Intell. Rev. 11(1-5), 1–6 (1997); special issue on lazy learning
3. Akbarzadeh-T, M.R., Davarynejad, M., Pariz, N.: Adaptive fuzzy fitness granulation for evolutionary optimization. International Journal of Approximate Reasoning 49(3), 523 (2008)
4. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician 46(3), 175–185 (1992)
5. Blanning, R.W.: The source and uses of sensivity information. Interfaces 4(4), 32–38 (1974)
6. Bui, L.T., Abbass, H.A., Essam, D.: Fitness inheritance for noisy evolutionary multi-objective optimization. In: GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 779–785. ACM, New York (2005)
7. Bull, L.: On model-based evolutionary computation. Soft Computing 3(2), 76–82 (1999)
8. Chen, J.H., Goldberg, D.E., Ho, S.Y., Sastry, K.: Fitness inheritance in multi-objective optimization. In: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 319–326. Morgan Kaufmann Publishers Inc., San Francisco (2002)
9. Coello, C.A.C., Lamont, G.B., Veldhuizen, D.A.V.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, Norwell (2002)
10. Deb, K.: An Efficient Constraint Handling Method for Genetic Algorithms. Computer Methods in Applied Mechanics and Engineering 186(2/4), 311–338 (2000)
11. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
12. Ducheyne, E., De Baets, B., de Wulf, R.: Is fitness inheritance useful for real-world applications? In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 31–42. Springer, Heidelberg (2003)
13. Ducheyne, E., Baets, B.D., Wulf, R.D.: Fitness inheritance in multiple objective evolutionary algorithms: A test bench and real-world evaluation. Applied Soft Computing 8(1), 337–349 (2007)

14. El-Beltagy, M., Nair, P., Keane, A.: Metamodeling techniques for evolutionary optimization of computationally expensive problems: promises and limitations. In: Proceedings of Genetic and Evolutionary Conference, pp. 196–203. Morgan Kaufmann, Orlando (1999)
15. Emmerich, M., Giannakoglou, K., Naujoks, B.: Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. Evolutionary Computation 10(4), 421–439 (2006)
16. Emmerich, M.T.M.: Single- and multi-objective evolutionary design optimization assisted by gaussian random field metamodels. PhD thesis, Technische Universitaet Dortmund (2005)
17. Ferrari, S., Stengel, R.F.: Smooth function approximation using neural networks. IEEE Transactions on Neural Networks 16(1), 24–38 (2005)
18. Forrester, A.I., Keane, A.J.: Recent advances in surrogate-based optimization. Progress in Aerospace Sciences 45, 50–79 (2009)
19. Giannakoglou, K.C.: Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. Progress in Aerospace Sciences 38(1), 43–76 (2002)
20. Goh, C.K., Tan, K.C.: A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. IEEE Transactions on Evolutionary Computation 13(1), 103–127 (2009)
21. Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Co., Reading (1989)
22. Grefenstette, J., Fitzpatrick, J.: Genetic search with approximate fitness evaluations. In: Proceedings of the International Conference on Genetic Algorithms and Their Applications, pp. 112–120 (1985)
23. Herrera, F., Lozano, M., Verdegay, J.L.: Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. Artificial Intelligence Review 12(4), 265–319 (1998)
24. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing Journal 9(1), 3–12 (2005)
25. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments-a survey. IEEE Transactions on Evolutionary Computation 9(3), 303–317 (2005)
26. Jin, Y., Olhofer, M., Sendhoff, B.: A framework for evolutionary optimization with approximate fitness functions. IEEE Transactions on Evolutionary Computation 6(5), 481–494 (2002)
27. Kecman, V.: Learning and soft computing: support vector machines, neural networks, and fuzzy logic models. Complex adaptive systems. MIT Press, Cambridge (2001)
28. Kim, H.S., Cho, S.B.: An efficient genetic algorithm with less fitness evaluation by clustering. In: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 2, pp. 887–894 (2001)
29. Knowles, J.: Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation 10(1), 50–66 (2006)
30. Kybic, J., Blu, T., Unser, M.: Generalized sampling; a variational approach – Part I: Theory. IEEE Transactions on Signal Processing 50(8), 1965–1976 (2002)
31. Kybic, J., Blu, T., Unser, M.: Generalized sampling; a variational approach – Part II: Applications. IEEE Transactions on Signal Processing 50(8), 1977–1985 (2002)

32. Lim, D., Ong, Y., Jin, Y., Sendhoff, B.: A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation. In: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 1288–1295. ACM Press, New York (2007)
33. Lim, D., Jin, Y., Ong, Y.S., Sendhoff, B.: Generalizing surrogate-assisted evolutionary computation. IEEE Transactions on Evolutionary Computation (2008) (in press)
34. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, 3rd edn. Springer, Heidelberg (1996)
35. Mota, F., Gomide, F.: Fuzzy clustering in fitness estimation models for genetic algorithms and applications. In: IEEE International Conference on Fuzzy Systems, pp. 1388–1395 (2006) ISBN: 0-7803-9488-7
36. Myers, R.H., Montgomery, D.C.: Response Surface Methodology – Process and Product Optimization Using Designed Experiments. Wiley Series in Probability and Statistics. John Wiley & Sons Inc., New York (2002)
37. Ong, Y., Nair, P., Keane, A.: Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA Journal 41(4), 687–696 (2003)
38. Pilato, C., Tumeo, A., Palermo, G., Ferrandi, F., Lanzi, P.L., Sciuto, D.: Improving evolutionary exploration to area-time optimization of FPGA designs. Journal of Systems Architecture 54(11), 1046 (2008)
39. Praveen, C., Duvigneau, R.: Low cost PSO using metamodels and inexact pre-evaluation: Application to aerodynamic shape design. Computer Methods in Applied Mechanics and Engineering 198(9-12), 1087–1096 (2009)
40. Queipo, N., Arévalo, C., Pintos, S.: The integration of design of experiments, surrogate modeling, and optimization for thermoscience research. Engineering with Computers 20, 309–315 (2005)
41. Queipo, N.V., Haftka, R.T., Shyy, W., Goela, T., Vaidyanathana, R., Tucker, P.K.: Surrogate-based analysis and optimization. Progress in Aerospace Sciences 41(1), 1–28 (2005)
42. Rasheed, K., Vattam, S., Ni, X.: Comparison of methods for using reduced models to speed up design optimization. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 1180–1187. Morgan Kaufmann, New York (2002)
43. Rasheed, K., Ni, X., Vattam, S.: Comparison of methods for developing dynamic reduced models for design optimization. Soft Computing Journal 9, 29–37 (2005)
44. Regis, R.G., Shoemaker, C.A.: Local function approximation in evolutionary algorithms for the optimization of costly functions. IEEE Trans. Evolutionary Computation 8(5), 490–505 (2004)
45. Reyes-Sierra, M., Coello, C.A.C.: A study of fitness inheritance and approximation techniques for multi-objective particle swarm optimization. In: The 2005 IEEE Congress on Evolutionary Computation, vol. 1, pp. 65–72 (2005)
46. Runarsson, T.: Approximate evolution strategy using stochastic ranking. In: Yen, G.G., Wang, L., Bonissone, P., Lucas, S.M. (eds.) IEEE World Congress on Computational Intelligence, Vancouver, Canada (2006)
47. Runarsson, T.P.: Constrained Evolutionary Optimization by Approximate Ranking and Surrogate Models. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 401–410. Springer, Heidelberg (2004)
48. Runarsson, T.P., Yao, X.: Stochastic Ranking for Constrained Evolutionary Optimization. IEEE Transactions on Evolutionary Computation 4(3), 284–294 (2000)

49. Salami, M., Hendtlass, T.: A fast evaluation strategy for evolutionary algorithms. Applied Soft Computing 2, 156–173 (2003)
50. Sanchez, E., Pintos, S., Queipo, N.: Toward an optimal ensemble of kernel-based approximations with engineering applications. Structural and Multidisciplinary Optimization, 1–15 (2007)
51. Sastry, K., Goldberg, D.E., Pelikan, M.: Don't evaluate, inherit. Tech. Rep. IlliGAL Report No. 2001013, Illinois Genetic Algorithms Laboratory (IlliGAL), Department of General Engineering, University of Illinois at Urbana-Champaign (2001)
52. Sastry, K., Pelikan, M., Goldberg, D.E.: Efficiency enhancement of genetic algorithms via building-block-wise fitness estimation. In: Congress on Evolutionary Computation, CEC 2004, pp. 720–727 (2004)
53. Schmidt, M., Lipson, H.: Coevolution of fitness predictors. IEEE Transactions on Evolutionary Computation 12(6), 736–749 (2008)
54. Shepard, D.: A two-dimensional interpolation function for irregularly-spaced data. In: Proceedings of the 1968 23rd ACM National Conference, pp. 517–524. ACM Press, New York (1968)
55. Sironen, S., Kangas, A., Maltamo, M., Kalliovirta, J.: Localization of growth estimates using non-parametric imputation methods. Forest Ecology and Management 256, 674–684 (2008)
56. Smith, R.E., Dike, B.A., Stegmann, S.A.: Fitness inheritance in genetic algorithms. In: SAC 1995: Proceedings of the 1995 ACM symposium on Applied computing, pp. 345–350. ACM Press, New York (1995)
57. Sokolov, A., Whitley, D., Barreto, A.M.S.: A note on the variance of rank-based selection strategies for genetic algorithms and genetic programming. Genetic Programming and Evolvable Machines 8(3), 221–237 (2007)
58. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary Computation 2(3), 221–248 (1994)
59. Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary computation and convergence to a pareto front. In: Koza, J.R. (ed.) Late Breaking Papers at the Genetic Programming 1998 Conference, Stanford University Bookstore, University of Wisconsin, Madison, Wisconsin, USA, Stanford, CA, USA (1998)
60. Wanner, E.F., Guimaraes, F.G., Takahashi, R.H.C., Lowther, D.A., Ramirez, J.A.: Multiobjective memetic algorithms with quadratic approximation-based local search for expensive optimization in electromagnetics. IEEE Transactions on Magnetics 44(6), 1126–1129 (2008)
61. Yang, D., Flockton, S.J.: Evolutionary algorithms with a coarse-to-fine function smoothing. In: IEEE International Conference on Evolutionary Computation, vol. 2, pp. 657–662 (1995)
62. Zhang, J., Yim, Y.S., Yang, J.: Intelligent selection of instances for prediction functions in lazy learning algorithms. Artif. Intell. Rev. 11(1-5), 175–191 (1997)
63. Zheng, X., Julstrom, B.A., Cheng, W.: Design of vector quantization codebooks using a genetic algorithm. In: Proceedings of 1997 IEEE International Conference on Evolutionary Computation, Piacataway, NJ, pp. 525–530 (1997)
64. Zhou, Z., Ong, Y.S., Nair, P.B.: Hierarchical surrogate-assisted evolutionary optimization framework. In: Congress on Evolutionary Computation, pp. 1586–1593. IEEE, Los Alamitos (2004)

Chapter 10
Multi-objective Model Predictive Control Using Computational Intelligence

Hirotaka Nakayama, Yeboon Yun, and Masakazu Shirakawa

Abstract. When function forms in mathematical models can not be given explicitly in terms of design variables, the values of functions are usually given by numerical/real experiments. Since those experiments are often expensive, it is important to develop techniques for finding a solution with as less number of experiments as possible. To this end, the model predictive optimization methods aim to find an optimal solution in parallel with predicting the function forms in mathematical models. Successive approximate optimization or metamodeling are of the same terminology. So far, several kinds of methods have been developed for this purpose. Among them, response surface method, design of experiments, Kriging method, active learning methods and methods using computational intelligence are well known. However, the subject of those methods is mainly static optimization. For dynamic optimization problems, the model predictive control has been developed along a similar idea to the above. This chapter discusses multi-objective model predictive control problems and proposes a method using computational intelligence such as support vector regression.

Keywords: multi-objective optimization, satisficing trade-off method, model predictive control, support vector regression.

Hirotaka Nakayama
Konan University, 8-9-1 Okamoto, Higashinada, Kobe 658-8501, Japan
e-mail: nakayama@konan-u.ac.jp

Yeboon Yun
Kagawa University, 2217-20 Hayashicho, Takamatsu 761-0396, Japan
e-mail: yun@eng.kagawa-u.ac.jp

Masakazu Shirakawa
Toshiba Corporation, 2-4 Suehiro-cho, Tsurumi-ku, Yokohama 230-0045, Japan
e-mail: masakazu1.shirakawa@toshiba.co.jp

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 249–264.
springerlink.com

10.1 Introduction

In many practical problems such as engineering design, function forms in mathematical models can not be given explicitly in terms of design variables, but the values of functions are usually given by numerical/real experiments. Since those experiments are often expensive, it is important to develop techniques for finding a solution with as less number of experiments as possible. Model predictive optimization (or sequential approximate optimization: SAO depending on literatures) has been developed to this aim extensively in recent years [13, 18, 19, 22].

In this chapter, we consider model predictive optimization problems under a dynamic environment with multiple objectives. For prediction of function forms, we apply some techniques of computational intelligence such as support vector regression or radial basis function networks. For optimization with multiple objectives, the satisficing trade-off method, which was developed by one of authors in '80s, is applied along with some meta-heuristic optimization method such as genetic algorithms. It will be shown how model prediction using computational intelligence combined with an interactive multi-objective optimization technique works well for multi-objective model predictive control problems.

10.2 Meta-modeling Using Computational Intelligence

Letting the design variables x be a vector of $\mathbb{R}^n$, optimization problems can be formulated as

$$\underset{x}{\text{minimize}} \qquad f(x) \qquad \text{over} \qquad x \in X \subset \mathbb{R}^n,$$

where the constraint set X may be represented by $g_i(x) \leqq 0,\ i = 1,\ldots,m$. The identification of f and X (or $g_i,\ i = 1,\ldots,m$) is called "modeling". We assume that those functions exist due to some physical rule, although their explicit function forms can not be known in terms of design variables x. Those situations are common in particular in engineering design problems. Under the circumstance, we try to get approximate functions $\hat{f}$ (and $\hat{g}_i,\ i = 1,\ldots,m$, if necessary). The approximation of objective/constraint functions based on several observations is called "*metamodeling*" in the sense of making a model of the model.

Now, our aim is to construct a good metamodel in the sense that

i) we can obtain an approximate optimal solution $\hat{x}^*$ through the metamodel with the property

$$|\hat{f}(\hat{x}^*) - f(x^*)| \leqq \varepsilon_1,$$

where $\hat{x}^*$ and x^* minimize $\hat{f}$ and f, respectively, and ε_1 is a given small positive number,

ii) the total number of observations is as small as possible,

iii) the metamodel $\hat{f}$ approximates well f entirely, if possible. Namely

$$\|\hat{f} - f\| \leqq \varepsilon_2,$$

where ε_2 is a given small positive number.

If our aim is merely to find the optimal solution minimizing $f(x)$, the above requirement iii) is not necessary, but the metamodel $\hat{f}$ approximates f sufficiently well at least in a neighborhood of the optimal solution x^*. Depending on practical problems, however, one may want to see the global behavior of the model f.

For metamodeling, several kinds of methods for regression can be available. Among them, Response Surface Method (RSM), Design of Experiments (DOE) and Kriging method are well known as methods for getting a well predicted model with less number of experiments. On the other hand, methods using computational intelligence such as Radial Basis Function Networks (RBFN) and Support Vector Regression (SVR) have been widely recognized to be effective for general nonlinear models.

Support Vector Regression (SVR) means the regression by *Support Vector Machine* (SVM). SVM was originally developed for pattern classification and later extended to regression (Vapnik *et al.* [3, 29]), and now is widely recognized as a powerful machine learning technique. Recently, Nakayama-Yun [21] claimed that several variants of SVM are possible from linear classifiers using goal programming, which was researched extensively in 1980's [8]. Consider several kinds of objectives in goal programming using both the exterior deviation and the interior deviation:

(i) minimize the maximum exterior deviation (decrease errors as much as possible),

(ii) maximize the minimum interior deviation (i.e., maximize the margin),

(iii) maximize the weighted sum of interior deviation,

(iv) minimize the weighted sum of exterior deviation.

Introducing the objective (iv) above leads to the soft margin SVM with slack variables (or, exterior deviations) which allow classification errors to some extent. Taking into account the objectives (ii) and (iv) above, we have $\nu-$SVM which was originally proposed from a different viewpoint by Schölkopf *et al.* [25]. On the other hand, using the objectives (i) and (ii) we have $\mu - \nu-$SVM [21], which seems attractive since they provide less support vectors than other SVM models.

Those SVM models can be applied to regression by introducing ε-insensitive loss function by Vapnik [29]. Denote the given training data set by (x_i, y_i), $i = 1, \ldots, \ell$. Suppose that the regression function f on the feature space $Z = \boldsymbol{\Phi}(x)$, where the map $\boldsymbol{\Phi}$ is defined implicitly by using the kernel function $K(x, x') = z^T z = \boldsymbol{\Phi}(x)^T \boldsymbol{\Phi}(x')$, is expressed by

$$f(z) = \sum_{i=1}^{\ell} w_i z_i + b,$$

and the linear ε insensitive loss function is defined by

$$L^{\varepsilon}(z,y,f) = |y - f(z)|_{\varepsilon} = \max(0, |y - f(z)| - \varepsilon).$$

For a given insensitivity parameter ε, $C-$SVR considering exterior deviations ξ_i, $i = 1,\ldots,\ell$ is formulated as follows:

$$\begin{aligned} \underset{w,b,\xi,\acute{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|w\|_2^2 + C\left(\frac{1}{\ell}\sum_{i=1}^{\ell}(\xi_i + \acute{\xi}_i)\right) && (C-\text{SVR})_P \\ \text{subject to} \quad & \left(w^T z_i + b\right) - y_i \leqq \varepsilon + \xi_i, \ \ i = 1,\ldots,\ell, \\ & y_i - \left(w^T z_i + b\right) \leqq \varepsilon + \acute{\xi}_i, \ \ i = 1,\ldots,\ell, \\ & \xi_i, \ \ \acute{\xi}_i \geqq 0, \end{aligned}$$

where C is a trade-off parameter between the norm of w and ξ_i ($\acute{\xi}_i$).

The dual formulation to the problem $(C-\text{SVR})_P$ using the kernel function $K(x,x') = z^T z = \Phi(x)^T \Phi(x')$ is given by

$$\begin{aligned} \underset{\alpha,\acute{\alpha}}{\text{maximize}} \quad & -\frac{1}{2}\sum_{i,j=1}^{\ell}(\acute{\alpha}_i - \alpha_i)(\acute{\alpha}_j - \alpha_j)K(x_i,x_j) && (C-\text{SVR}) \\ & + \sum_{i=1}^{\ell}(\acute{\alpha}_i - \alpha_i)y_i - \varepsilon\sum_{i=1}^{\ell}(\acute{\alpha}_i + \alpha_i) \\ \text{subject to} \quad & \sum_{i=1}^{\ell}(\acute{\alpha}_i - \alpha_i) = 0, \\ & 0 \leqq \acute{\alpha}_i \leqq \frac{C}{\ell}, \ \ 0 \leqq \alpha_i \leqq \frac{C}{\ell}, \ \ i = 1,\ldots,\ell. \end{aligned}$$

In order to decide the insensitivity parameter ε automatically, Schölkopf and Smola proposed $\nu-$SVR which is formulated by the following [25]:

$$\begin{aligned} \underset{w,b,\varepsilon,\xi,\acute{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|w\|_2^2 + C\left(\nu\varepsilon + \frac{1}{\ell}\sum_{i=1}^{\ell}(\xi_i + \acute{\xi}_i)\right) && (\nu-\text{SVR})_P \\ \text{subject to} \quad & \left(w^T z_i + b\right) - y_i \leqq \varepsilon + \xi_i, \ \ i = 1,\ldots,\ell, \\ & y_i - \left(w^T z_i + b\right) \leqq \varepsilon + \acute{\xi}_i, \ \ i = 1,\ldots,\ell, \\ & \varepsilon, \ \ \xi_i, \ \ \acute{\xi}_i \geqq 0, \end{aligned}$$

where C and $0 < \nu \leqq 1$ are trade-off parameters between the norm of w and ε and ξ_i ($\acute{\xi}_i$).

The dual formulation to the problem $(\nu-\mathrm{SVR})_P$ is given by

$$\begin{aligned}
\underset{\alpha,\acute{\alpha}}{\text{maximize}} \quad & -\frac{1}{2}\sum_{i,j=1}^{\ell}(\acute{\alpha}_i-\alpha_i)(\acute{\alpha}_j-\alpha_j)K(x_i,x_j) \qquad (\nu-\mathrm{SVR})\\
& +\sum_{i=1}^{\ell}(\acute{\alpha}_i-\alpha_i)y_i\\
\text{subject to} \quad & \sum_{i=1}^{\ell}(\acute{\alpha}_i-\alpha_i)=0,\\
& \sum_{i=1}^{\ell}(\acute{\alpha}_i+\alpha_i)\leqq C\nu,\\
& 0\leqq\acute{\alpha}_i\leqq\frac{C}{\ell},\ \ 0\leqq\alpha_i\leqq\frac{C}{\ell},\ \ i=1,\ldots,\ell.
\end{aligned}$$

In a similar fashion to classification, $C-\mathrm{SVR}$ can be extended to $\mu-\mathrm{SVR}$ aiming to minimize the maximum exterior deviation ξ as follows:

For a given insensitivity parameter ε,

$$\begin{aligned}
\underset{w,b,\xi,\acute{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|w\|_2^2+\mu(\xi+\acute{\xi}) \qquad (\mu-\mathrm{SVR})_P\\
\text{subject to} \quad & \left(w^Tz_i+b\right)-y_i\leqq\varepsilon+\xi,\ \ i=1,\ldots,\ell,\\
& y_i-\left(w^Tz_i+b\right)\leqq\varepsilon+\acute{\xi},\ \ i=1,\ldots,\ell,\\
& \xi,\ \acute{\xi}\geqq 0,
\end{aligned}$$

where μ is a trade-off parameter between the norm of w and ξ $(\acute{\xi})$.

The dual formulation to the problem $(\mu-\mathrm{SVR})_P$ is given by

$$\begin{aligned}
\underset{\alpha,\acute{\alpha}}{\text{maximize}} \quad & -\frac{1}{2}\sum_{i,j=1}^{\ell}(\acute{\alpha}_i-\alpha_i)(\acute{\alpha}_j-\alpha_j)K(x_i,x_j) \qquad (\mu-\mathrm{SVR})\\
& +\sum_{i=1}^{\ell}(\acute{\alpha}_i-\alpha_i)y_i-\varepsilon\sum_{i=1}^{\ell}(\acute{\alpha}_i+\alpha_i)\\
\text{subject to} \quad & \sum_{i=1}^{\ell}(\acute{\alpha}_i-\alpha_i)=0,\\
& \sum_{i=1}^{\ell}\acute{\alpha}_i\leqq\mu,\ \ \sum_{i=1}^{\ell}\alpha_i\leqq\mu,\\
& \acute{\alpha}_i\geqq 0,\ \ \alpha_i\geqq 0,\ \ i=1,\ldots,\ell.
\end{aligned}$$

Combining $\mu-\mathrm{SVR}$ and $\nu-\mathrm{SVR}$, we can derive another formulation which may be defined as $\mu-\nu-\mathrm{SVR}$:

$$\begin{aligned} \underset{w,b,\varepsilon,\xi,\acute{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|w\|_2^2 + \nu\varepsilon + \mu(\xi + \acute{\xi}) \qquad (\mu - \nu\text{−SVR})_P \\ \text{subject to} \quad & \left(w^T z_i + b\right) - y_i \leqq \varepsilon + \xi, \ \ i = 1, \ldots, \ell, \\ & y_i - \left(w^T z_i + b\right) \leqq \varepsilon + \acute{\xi}, \ \ i = 1, \ldots, \ell, \\ & \varepsilon, \ \ \xi, \ \ \acute{\xi} \geqq 0, \end{aligned}$$

where ξ ($\acute{\xi}$) denotes the maximum outer deviation from the ε band, and ν and μ are trade-off parameters between the norm of w and ε and ξ ($\acute{\xi}$) respectively.

In this formulation, however, at least either ε or ξ (or $\acute{\xi}$) vanishes at the solution according to $\nu \geqq 2\mu$ or $\nu \leqq 2\mu$. Therefore, $\mu - \nu$−SVR may be reduced to the following formulation simply called ν_ε−SVR (or similarly μ_ξ−SVR replacing ε by ξ ($\acute{\xi}$) and $\nu\varepsilon$ by $\mu(\xi + \acute{\xi})$):

$$\begin{aligned} \underset{w,b,\varepsilon}{\text{minimize}} \quad & \frac{1}{2}\|w\|_2^2 + \nu\varepsilon \qquad (\nu_\varepsilon - \text{SVR})_P \\ \text{subject to} \quad & \left(w^T z_i + b\right) - y_i \leqq \varepsilon, \ \ i = 1, \ldots, \ell, \\ & y_i - \left(w^T z_i + b\right) \leqq \varepsilon, \ \ i = 1, \ldots, \ell, \\ & \varepsilon \geqq 0, \end{aligned}$$

where ν is a trade-off parameter between the norm of w and ε.

This formulation can be regarded as a kind of Tchebyshev approximation in the feature space.

The dual formulation of $(\nu_\varepsilon - \text{SVR})_P$ is given by

$$\begin{aligned} \underset{\alpha,\acute{\alpha}}{\text{maximize}} \quad & -\frac{1}{2}\sum_{i,j=1}^{\ell}(\acute{\alpha}_i - \alpha_i)(\acute{\alpha}_j - \alpha_j)K(x_i, x_j) \qquad (\nu_\varepsilon\text{−SVR}) \\ & + \sum_{i=1}^{\ell}(\acute{\alpha}_i - \alpha_i)y_i \\ \text{subject to} \quad & \sum_{i=1}^{\ell}(\acute{\alpha}_i - \alpha_i) = 0, \\ & \sum_{i=1}^{\ell}(\acute{\alpha}_i + \alpha_i) \leqq \nu, \\ & \acute{\alpha}_i \geqq 0, \ \ \alpha_i \geqq 0, \ \ i = 1, \ldots, \ell. \end{aligned}$$

It has been observed that μ−SVR and ν_ε−SVR provide the least number of support vectors while keeping a reasonable error rate, compared with C−SVR and ν−SVR. That is, μ−SVR and ν_ε−SVR is promising for sparse approximation which means the computation is less expensive. The fact that μ−SVR and ν_ε−SVR yields good function approximation with reasonable accuracy and with less support vectors, is important in practice in engineering design.

10.3 Aspiration Level Approach to Interactive Multi-objective Optimization

Suppose that we have several (usually conflicting) objective functions $f_1(x)$, $\ldots, f_r(x)$ to be minimized. In such a multi-objective optimization, in general, there is no solution minimizing all objective functions simultaneously. A *Pareto solution* $\hat{x}$ is defined as a solution for which in order to improve some objectives we have to sacrifice some of other objectives (see Fig. 10.1).

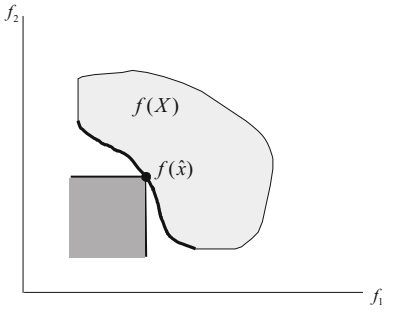


Fig. 10.1 Pareto solution in the objective space

Since there may be many Pareto solutions in practice, the final decision should be made among them taking the total balance over all criteria into account. This is a problem of value judgment of decision maker (DM). The balancing over criteria is usually called *trade-off*. The set of Pareto values (namely, the values of objective functions corresponding to Pareto solutions) is called *Pareto frontier*. If we can visualize the Pareto frontier, DM can easily make his trade-off analysis on the basis of the shown Pareto frontier. In recent years, studies aimed at generating Pareto frontier have been developed with a help of meta-heuristic algorithms such as evolutionary algorithms and particle swarm optimization (see, for example, [1, 2, 5, 11, 12]).

On the other hand, it is not so easy to understand the trade-off relation of Pareto frontier with more than 3 dimensions. Since 1970's, *interactive multi-objective programming* techniques have been developed in order to overcome this difficulty: those methods search a solution in an interactive way with DM while making trade-off analysis on the basis of DM's value judgment (see, for example, [1]). Among them, the aspiration level approach is now recognized to be effective in many practical problems. As one of aspiration level approaches, one of authors proposed the *satisficing trade-off method* [20].

Suppose that we have objective functions $f(x) := (f_1(x), \ldots, f_r(x))^T$ to be minimized over $x \in X \subset \mathbb{R}^n$. In the satisficing trade-off method, the aspiration level at the k-th iteration $\overline{f}^k$ is modified as follows:

$$\overline{f}^{k+1} = T \circ P(\overline{f}^k).$$

Here, the operator P selects the Pareto solution nearest in some sense to the given aspiration level $\overline{f}^k$. The operator T is the trade-off operator which changes the k-th aspiration level $\overline{f}^k$ if DM does not compromise with the shown solution $P(\overline{f}^k)$. Of course, since $P(\overline{f}^k)$ is a Pareto solution, there exists no feasible solution which makes all criteria better than $P(\overline{f}^k)$, and thus DM has to trade-off among criteria if he wants to improve some of criteria. Based on this trade-off, a new aspiration level is decided as $T \circ P(\overline{f}^k)$. Similar process is continued until DM obtains an agreeable solution.

The operation which gives a Pareto solution $P(\overline{f}^k)$ nearest to $\overline{f}^k$ is performed by some auxiliary scalar optimization:

$$\underset{x}{\text{minimize}} \qquad \max_{1 \leqq i \leqq r} \left\{ \omega_i \left(f_i(x) - \overline{f}_i \right) \right\} + \alpha \sum_{i=1}^{r} \omega_i f_i(x),$$

where α is usually set a sufficiently small positive number, say 10^{-6}.

The weight ω_i is usually given as follows: Let f_i^* be an ideal value which is usually given in such a way that $f_i^* < \min\{f_i(x) \mid x \in X\}$. For this circumstance, we set

$$\omega_i^k = \frac{1}{\overline{f}_i^k - f_i^*}.$$

With this weight, Fig. 10.2 illustrates the operation P (namely, finding the nearest Pareto solution to the given aspiration level) and T (namely, revising the aspiration level).

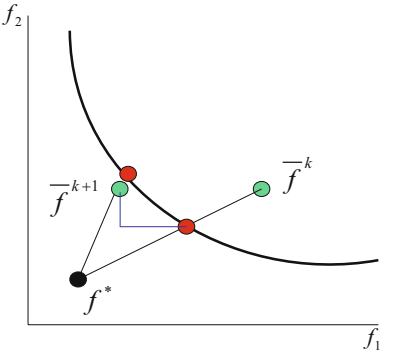


Fig. 10.2 Satisificing Trade-off Method

For more details of nonlinear interactive methods for multiobjective optimization, the readers should refer Branke *et al.* [1], Gal *et al.* [9], Miettinen [15], Sawaragi *et al.* [24], Steuer [27], Wierzbicki *et al.* [30], etc. In particular, Nakayama [17] emphasizes why the weighting method using linearly weighted sum of objective functions does not work so well.

10.4 Multi-objective Model Predictive Control

Consider dynamic optimization problems in this section. Along optimal control theory, let $u(t),\ x(t)$ denote the control (input) vector and the state vector at the time t, respectively, and T the terminal time. The theory assumes the mathematical model as follows:

$$\begin{aligned} \underset{u,x}{\text{minimize}} \quad & J = \phi[x(T)] + \int_0^T F(x(t),u(t),t)dt \\ \text{subject to} \quad & \dot{x} = f(x(t),u(t),t),\ \ x(0) = x_0. \end{aligned} \tag{10.1}$$

If the function form in the above model is explicitly given, then we can apply some techniques on the basis of optimal control theory. However, we assume that some of function forms, in particular the dynamic system equation (10.1), can not explicitly be given. Under this circumstance, we predict some of future state $x(t+1),\ldots,x(t+p_1)$ for given $u(t+1),\ldots,u(t+p_2)$, where the prediction period p_1 and the control period p_2 are given $(p_1 \geqq p_2)$. Our aim is to decide the optimal control sequence $u(t)$ over $[0,T]$.

Suppose that our problem to be considered in this section has multiple objectives

$$J = (J_1,\ldots,J_r)^T.$$

For example, those objectives are the energy consumption, constraints of terminal state, the terminal time (T) itself and so on.

For predicting the future state, we apply a support vector regression technique, namely $\mu-$SVR which was introduced in the previous section. It has been observed that $\mu-$SVR provides less support vectors than other SVRs.

In order to get the final decision for these multi-objective problems, applying the satisficing trade-off method [20] which is an aspiration level based method, we summarize the algorithms as belows:

Step 1. Predict the model f by using $\mu-$SVR based on the past state and control $(x(k-q),x(k-q+1),\ldots,x(k),u(k-q),u(k-q+1),\ldots,u(k-1)),\ \ k=q,\ldots,t$, where q represents a depth of sampling training data and $x(0)=x_0$ (denote $\hat{f}$ as the predicted function of f).

Step 2. Decide a control $u^*(t)$ at the time t by using genetic algorithm:

(i) Generate randomly N individuals of control sequence:

$$u_j(t),\ u_j(t+1),\ldots,u_j(t+p_2-1),\ \ j=1,2,\ldots,N,$$

and set $u_j(t+i) = u_j(t+p_2-1)$ for $i \geqq p_2$, generally.

(ii) Predict the next state vector $x_j(k+1)$ for each control sequence from the present time t to the time $t+p_1$:

$$x_j(k+1) - x_j(k) := \hat{f}(x_j(k),u_j(k)),\ k=t,t+1,\ldots,t+p_1-1.$$

(iii) For $x_j = (x(0), x(1), \ldots, x(t), x_j(t+1), \ldots, x_j(t+p_1))$ and $u_j = (u(0), u(1), \ldots, u(t), u_j(t+1), \ldots, x_j(t+p_1-1))$, calculate the value of auxiliary scalar function of satisficing trade-off method: for the aspiration level $\overline{J} = (\overline{J}_1, \ldots, \overline{J}_r)^T$ given by a decision maker[2],

$$z_j = \max_{1 \leqq i \leqq r} \left\{ w_i \left(J_i(u_j, x_j) - \overline{J}_i \right) \right\} + \alpha \sum_{i=1}^{r} w_i \left(J_i(u_j, x_j) - \overline{J}_i \right),$$

where $w_i = \frac{1}{\overline{J}_i - J_i^*}$ and J_i^* is an ideal value of i-th objective function.

(vi) Evaluating the individuals of control sequence by the value of z_j, generate new individuals of control sequence through natural selection and genetic operators (for details, see [5]).

(v) Repeat (ii)–(iv) until a stop condition, for example the number of iteration, holds.

Step 3. Decide the best control sequence u^* such that $\min_{j=1,\ldots,N} z_j$, and observe the real value $x(t+1)$ using $u(t) = u^*(t)$.

Step 4. Stop if $t = T$, otherwise update $t \leftarrow t+1$ and go to Step 1.

In order to show the effectiveness of the stated method, we shall apply our method to an example of flight control of an aircraft described in [14].

Let x_1 be the angle of attack, x_2 the pitch angle, x_3 the pitch rate and x_4 the altitude. In addition, let the output be the pitch angle y_1 (rad), the altitude y_2 (m) and the altitude rate y_3 (m/sec). Suppose that the elevator angle u (rad) is just an input (controller), and the aircraft flies with a constant velocity 128.2 m/sec at the altitude 5000 m. Then the linearized dynamic equation is given by

$$\dot{x} = Ax + Bu, \quad y = Cx + Du. \tag{10.2}$$

The above equation (10.2) for the discrete time can be represented by

$$x(t+1) = Ax(t) + Bu(t), \quad y(t) = Cx(t) + Du(t), \tag{10.3}$$

where

$$A = \begin{bmatrix} -1.2822 & 0 & 0.98 & 0 \\ 0 & 0 & 1 & 0 \\ -5.4293 & 0 & -1.8366 & 0 \\ -128.2 & 128.2 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} -0.3 \\ 0 \\ -17 \\ 0 \end{bmatrix},$$

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -128.2 & 128.2 & 0 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

[2] A decision maker may change her/his aspiration level from the one at the previous time $t-1$.

We have the following structural constraints

$$|u| \leqq 0.262 \text{ rad } (15^\circ), \quad |\dot{u}| \leqq 0.524 \text{ rad/sec } (30^\circ/\text{sec}),$$

and in addition, for the comfortability of passengers we impose the following constraint

$$|y_1| \leqq 0.349 \text{ rad } (20^\circ).$$

Our aim is to get up to the target altitude H as soon as possible, and as comfortable for passengers as possible within the given time T. Therefore, our objective at the present time t is given by

$$\text{minimize} \qquad J_1 = \sum_{k=0}^{t+p_1} \left(1 - \frac{y_2(k)}{H}\right)^2$$

$$\text{minimize} \qquad J_2 = \sum_{k=0}^{t+p_1} \left(\frac{y_1(k)}{0.349}\right)^2$$

Now, we optimize the input (i.e., the elevator angle) without knowing the explicit form of dynamic equation (10.3), and set the target altitude $H = 400$ m. Both the prediction period p_1 and the control period p_2 are 5 sec and 1.5 sec, respectively, and the depth of sampling training data $q = 1$. The terminal time T is given by 20 sec, and the sampling period for discretization of dynamics is 0.5 sec. In GA, we use BLX_α, $\alpha = 0.25$ (blend crossover, BLX) which is well known as a real-coded GA [6]. The size N of individuals is 100 and the iteration number is 100. Each constraint is treated by the penalty method (other techniques to deal with constraints in GA can be referred to [2, 28]).

Case 1

Suppose that we are at the time $t = 5$, and consider two situations without/with turbulence for 5 seconds from the present time $t = 5$ (the observed altitude may be considerably different with the predicted one by turbulence). The aspiration level is given by $\overline{J}_1 = 6.0$, $\overline{J}_2 = 10.0$, and the ideal point $J_1^* = 3.0$, $J_2^* = 5.0$.

Fig. 10.3 shows the solutions by using the satisficing trade-off method with the predicted model. Here, the symbol $\triangle$ represents the aspiration level, the symbol $\diamond$ the solution without turbulence and the symbol $\circ$ the solution with turbulence. Fig. 10.4 and Fig. 10.5 show each response corresponding to the obtained solution. Compared Fig. 10.5 with Fig. 10.4, because of turbulence, there are relatively strong fluctuations in controlling the elavator angle u.

In this case, one may see that the time in which the increase of altitude attains 400 m becomes longer because the comfortability of passengers is considered relatively more important. However, since the upper bound of the altitude rate y_3 is not constrained, the pitch angle y_1 may take the value of the upper bound during the transient state. Thus, in the following case 2, we consider the case in which the upper bound of the altitude rate y_3 is 30 m/sec.

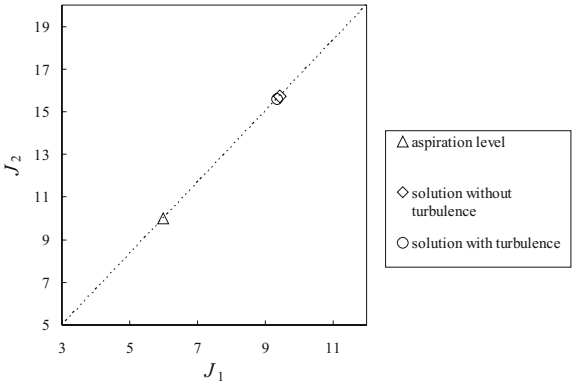


Fig. 10.3 Solution by using the aspiration level method (case 1)

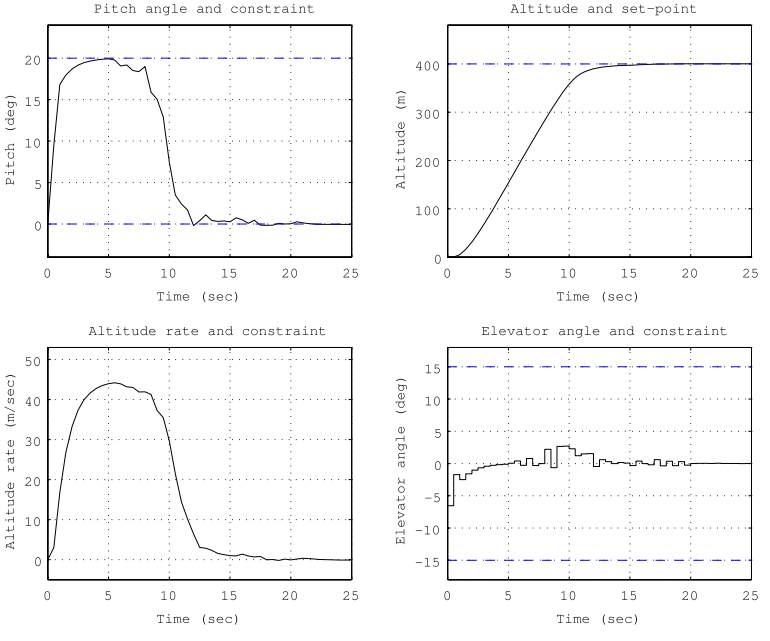


Fig. 10.4 Responses without turbulence (case 1)

Case 2

The aspiration level is given by $\overline{J}_1 = 8.0$, $\overline{J}_2 = 8.0$, and the ideal point $J_1^* = 4.0$, $J_2^* = 4.0$. We show the solutions by using the satisficing trade-off method with the predicted model in Fig. 10.6. Fig. 10.7 and Fig. 10.8 show each corresponding response to the obtained solution.

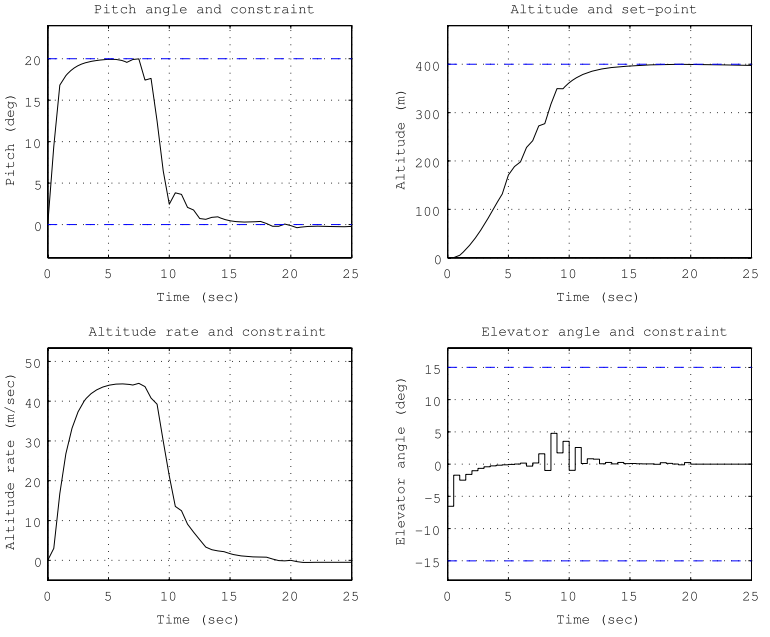


Fig. 10.5 Responses with turbulence (case 1)

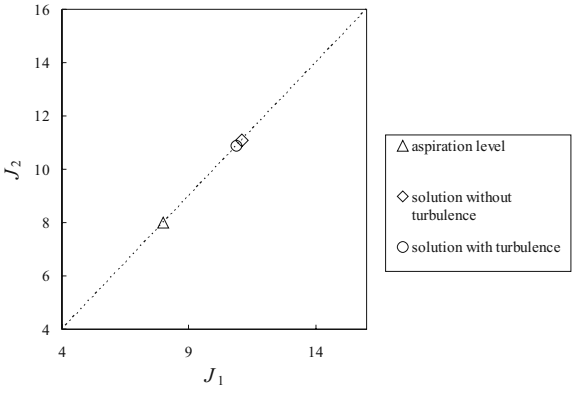


Fig. 10.6 Solution by using the aspiration level method (case 2)

Comparing the results of case 1 and case 2, it is seen in case 2 that the pitch angle y_1 during the transient state is smaller than its upper bound due to the limitation of the altitude rate y_3. As seen from Fig. 10.7 and Fig. 10.8, consequently, there are strong fluctuations in controlling the elavator angle u from $t = 10$ to $t = 15$, in order to attain the target altitude 400 m as fast as possible. Moreover, the curve of the pitch angle y_1 in Fig. 10.8 is not as smooth as the one in Fig. 10.7.

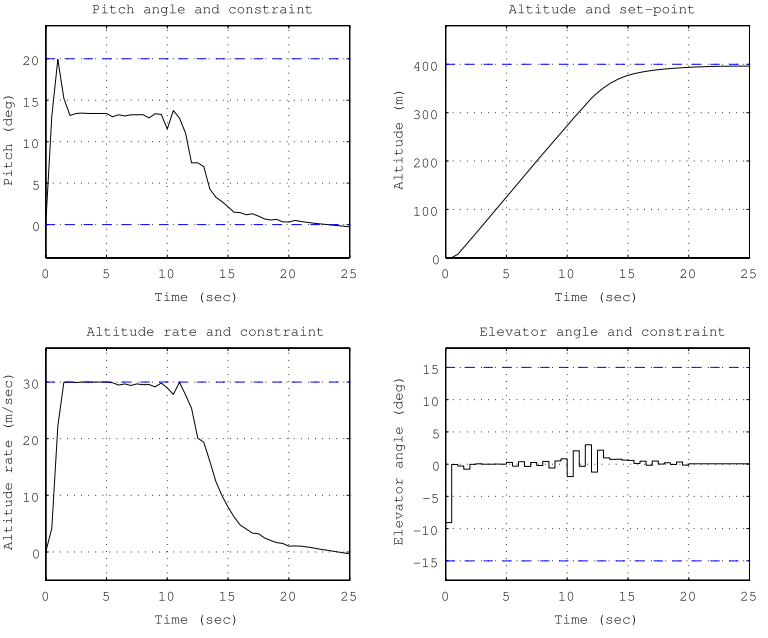


Fig. 10.7 Responses without turbulence (case 2)

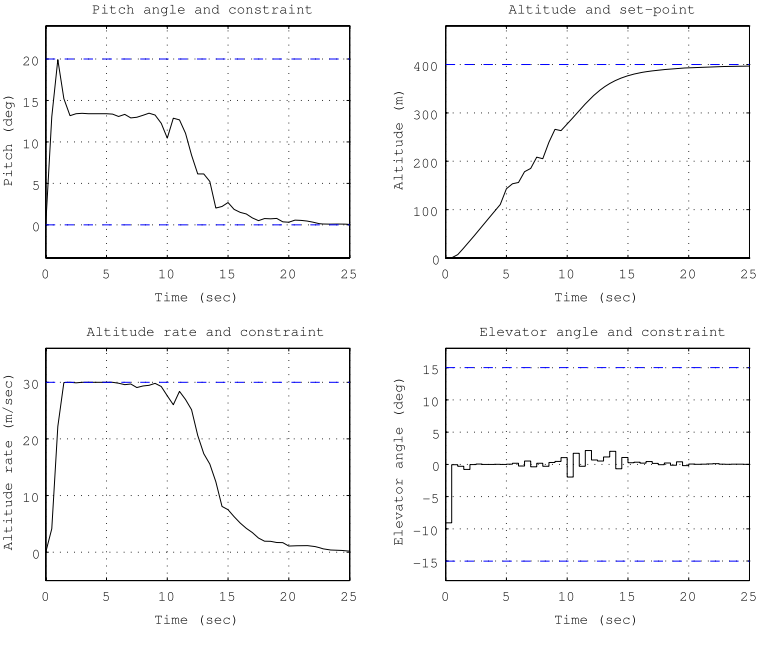


Fig. 10.8 Responses with turbulence (case 2)

10.5 Concluding Remarks

We discussed a method for multi-objective model predictive control. $\mu-$SVR is applied for model prediction, while the satisficing trade-off method for solving multi-objective optimization. It has been observed that the proposed method works well through our several experiments. A difficulty in the problems stated in this chapter is that the terminal time is variable. In this chapter, for a fixed terminal time, firstly, we find a sequence of optimal control. Next, we vary the terminal time. This procedure requires a lot of iterations for optimization. It is desirable to reduce the number of iterations for optimization from this viewpoint in the future.

Another difficulty occurs in finding optimal solutions to predicted models by using genetic algorithms (GAs): In general, GAs are not good at treating constraints. As can be seen in our numerical example, it is difficult to obtain optimal solutions for constrained optimization problems in a reasonable precision by GAs such as BLX_α used in this chapter. It seems better to apply usual gradient-based methods for treating simple constraints such as linear forms. In addition, since there have been developed some methods for treating constraints in a framework of evolutionary algorithms [2, 11, 12], comparative studies on the performance of those methods should be further research topics.

References

1. Branke, J., Deb, K., Miettinen, K., Slowinski, R.: Multiobjective Optimization Interactive and Evolutionary Approaches. Springer, Heidelberg (2008)
2. Coello, C.A.C.: Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. Computer Methods in Applied Mechanics and Engineering 191(11-12), 1245–1287 (2002)
3. Cortes, C., Vapnik, V.: Support Vector Networks. Machine Learning 20, 273–297 (1995)
4. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, Cambridge (2000)
5. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Ltd., Chichester (2001)
6. Eshelman, L.J., Schaffer, J.D.: Real-Coded Genetic Algorithms and Interval-Schemata. In: Foundations of Genetic Algorithms, vol. 2, pp. 187–202 (1993)
7. Erenguc, S.S., Koehler, G.J.: Survey of Mathematical Programming Models and Experimental Results for Linear Discriminant Analysis. Managerial and Decision Economics 11, 215–225 (1990)
8. Freed, N., Glover, F.: Simple but Powerful Goal Programming Models for Discriminant Problems. European Journal of Operational Research 7, 44–60 (1981)
9. Gal, T., Stewart, T.J., Hanne, T.: Multicriteria Decision Making – Advances in MCDM Models, Algorithms, Theory, and Applications. Kluwer Academic Publishers, Dordrecht (1999)
10. Glover, F.: Improved Linear Programming Models for Discriminant Analysis. Decision Sciences 21, 771–785 (1990)
11. Goh, C.K., Tan, K.C.: An Investigation on Noisy Environments in Evolutionary Multiobjective Optimization. IEEE Trans. Evolutionary Computation 11(3), 354–381 (2007)

12. Goh, C.K., Ong, Y.S., Tan, K.C., Teoh, E.J.: An investigation on evolutionary gradient search for multi-objective optimization. In: IEEE Congress on Evolutionary Computation, pp. 3741–3746 (2008)
13. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient Global Optimization of Expensive Black-Box Functions. J. of Global Optimization 13, 455–492 (1998)
14. Maciejowski, J.M.: Predcitive Control with constraints. Pearson Educational Limited, London (2002)
15. Miettinen, K.M.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Dordrecht (1999)
16. Myers, R.H., Montgomery, D.C.: Response Surface Methodology: Process and Product Optimization using Designed Experiments. Wiley, Chichester (1995)
17. Nakayama, H.: Aspiration Level Approach to Interactive Multi-objective Programming and its Applications. In: Pardalos, P.M., Siskos, Y., Zopounidis, C. (eds.) Advances in Multicriteria Analysis, pp. 147–174. Kluwer Academic Publishers, Dordrecht (1995)
18. Nakayama, H., Arakawa, M., Sasaki, R.: Simulation based Optimization for Unknown Objective Functions. Optimization and Engineering 3, 201–214 (2002)
19. Nakayama, H., Arakawa, M., Washino, K.: Optimization for Black-box Objective Functions. In: Pardalos, P.M., Tseveendorj, I., Enkhbat, R. (eds.) Optimization and Optimal Control, pp. 185–210. World Scientific, Singapore (2003)
20. Nakayama, H., Sawaragi, Y.: Satisficing Trade-off Method for Multi- objective Programming. In: Grauer, M., Wierzbicki, A. (eds.) Interactive Decision Analysis, pp. 113–122. Springer, Heidelberg (1984)
21. Nakayama, H., Yun, Y.: Generating Support Vector Machines using Multiobjective Optimization and Goal Programming. In: Multi-objective Mahine Learning. Studies in Computational Intelligence. Springer, Heidelberg (2006)
22. Nakayama, H., Yun, Y., Yoon, M.: Sequential Approximate Multiobjecitve Optimization using Computational Intelligence. Springer Series on Vector Optimization (to appear, 2009)
23. Radcliffe, N.J.: Forma Analysis and Random Respectful Recombination. In: Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 222–229 (1991)
24. Sawaragi, Y., Nakayama, H., Tanino, T.: Theory of Multiobjective Optimization. Academic Press, London (1985)
25. Schölkopf, B., Smola, A.J.: New Support Vector Algorithms. NeuroCOLT2 Technical report Series, NC2-TR-1998-031 (1998)
26. Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2002)
27. Steuer, R.: Mutiple Criteria Optimization: Theory, Computation, and Application. Wiley, Chichester (1986)
28. Runarsson, T.P., Yao, X.: Stochastic Ranking for Constrained Evolutionary Optimization. IEEE Transactions on Evolutionary Computation 4(3), 284–294 (2000)
29. Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, New York (1998)
30. Wierzbicki, A.P., Makowski, M., Wessels, J.: Model-based Decision Support Methodology with Environmental Applications. Kluwer Academic Publishers, Dordrecht (2000)

Chapter 11
Improving Local Convergence in Particle Swarms by Fitness Approximation Using Regression

Stefan Bird and Xiaodong Li*

Abstract. In this chapter we present a technique that helps Particle Swarm Optimisers (PSOs) locate an optimum more quickly, through fitness approximation using regression. A least-squares regression is used to estimate the shape of the local fitness landscape. From this shape, the expected location of the peak is calculated and the information given to the PSO. By guiding the PSO to the optimum, the local convergence speed can be vastly improved. We demonstrate the effectiveness of using regression on several static multimodal test functions as well as dynamic multimodal test scenarios (Moving Peaks). This chapter also extends the Moving Peaks test suite by enhancing the standard conic peak function to allow the creation of asymmetrical and additional multiple local peaks. The combination of this technique and a speciation-based PSO compares favourably to another multi-swarm PSO algorithm that has proven to be working well on the Moving peaks test functions.

Keywords: Particle Swarm Optimization, Swarm Intelligence, Optimization in Dynamic Environments, Regression Techniques, Numerical Optimization.

11.1 Introduction

Local search methods are known for their extremely fast convergence, however they are also highly susceptible to becoming trapped in the first optimum they find. Many real world problems are far too complex to be solvable by these methods. Evolutionary Algorithms (EAs) and Particle Swarm Optimisation (PSO) have been proved

Stefan Bird
School of Computer Science and IT, RMIT University, Melbourne, Australia
e-mail: stbird@seatiger.org

Xiaodong Li
School of Computer Science and IT, RMIT University, Melbourne, Australia
e-mail: xiaodong.li@rmit.edu.au

* Corresponding author.

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 265–293.
springerlink.com

to be effective search strategies. Both algorithms are able to converge on an optimum even when the catchment area occupies only a small portion of the search space. These algorithms are far less likely to become trapped in a local peak than local search, especially when combined with a diversification measure. This property comes at a cost though, as they take far more evaluations to locate an optimum than local search methods.

Combining local search with an EA (or PSO) can provide the best of both worlds, as we gain the robustness of the population-based algorithms as well as the local search's convergence speed [26]. This hybrid approach is quite common, for example [24, 34, 36]. However, using the local search requires extra fitness evaluations to be performed; when considered over the entire optimisation process, these evaluations can be very costly.

To overcome the issue associated with high computational cost, several fitness approximation techniques have been developed (see also Section 11.2.1). For example, in aerodynamic structure optimization, since simulations for computational fluid dynamics are usually very expensive, approximate models were developed [1, 16]. Fitness approximation was also used in conjunction with an evolutionary algorithm for protein structure prediction to cut down the computational cost [27]. Another interesting study in [23] shows the benefits of approximating fitness landscape using a polynomial regression model.

This chapter presents a fitness approximation technique that helps Particle Swarm Optimisers (PSOs) locate an optimum more quickly, without requiring any extra fitness evaluations. Rather than performing a local search, we use the candidate solutions already tested by a PSO to create a surface that best fits the peak. We then attempt to calculate the highest point of this surface. Provided that the local features of the fitness landscape roughly match our surface, the optimum should be very close to the computed highest point. This allows us to very quickly hone in on the actual maximum point – with each successive attempt we know more and more about the landscape, improving our estimation further.

Our early study in [7] suggested that regression was able to improve efficiency in handling some dynamic optimization functions (ie., Moving peaks scenario 2). Extending the early findings, this chapter provides several new investigations on using regression. Firstly, we identified similarities and differences between our regression method and other existing works in literature. Secondly, we included several widely used static multimodal test functions to further verify if regression is effective in improving local convergence for solving static multimodal problems in general. In addition, we also adopted a Generic Hump function (which is tunable with the number of peaks and the number of dimensions) to study especially if regression is effective in reducing the number of evaluations for high dimensional multimodal problems. Furthermore, we extended the Moving Peaks test functions to allow creation of asymmetrical and additional multiple local peaks. The effectiveness of regression on these more complex peak shapes was examined.

Although fitness approximation using regression has been developed for EAs [23], this study represents a first attempt to integrate regression with a PSO for improving local convergence. This paper is organised as follows. Section 11.2 provides

the background of this technique and the algorithms we have used to test it. A detailed explanation of the method follows in Section 11.3. Sections 11.4 and 11.5 show the experimental setup and results. Our conclusions will be presented in Section 11.6, as well as some further research directions.

11.2 Background

Local search methods are typically designed to rapidly locate an optimum once its general area has been found. These methods are susceptible to becoming trapped in a local optimum, meaning that they are most effectively used once the peak's location is already approximately known. The most intuitive local search method is hill climbing. This method works by continually sampling the decision space around the best point found so far [26]. At each iteration, a point somewhere near the current best is selected and evaluated. If the new point is better than the current best it replaces it, otherwise the new point is discarded. By repeating this process many times we "climb" the peak of the initial starting point, usually chosen randomly. One variant of hill climbing is gradient ascent, which works by using the derivative of the fitness function to guide the search direction [14]. The next point chosen to search is one that is close to the last point evaluated, but in the direction of the steepest ascent. However, this technique can only be used with fitness functions where the gradient can be computed.

To improve the local convergence of a PSO on a multimodal fitness landscape we are incorporating a method that approximates the fitness landscape using regression. This chapter will demonstrate that the use of regression and a convergence enhancer can dramatically improve local convergence while saving computational cost. This section provides the background for these techniques.

11.2.1 Fitness Approximation

In recent years, several studies have proposed EAs incorporating fitness approximation with an aim to improve performance while not incurring expensive computational cost [15, 23, 28]. Typically these EAs employ a *surrogate* model in place of the expensive original function evaluations. The surrogate model is used to approximate the original fitness function by using a small set of evaluated search points chosen from the EA population. The goal is to reduce the number of expensive original function evaluations while retaining an accurate approximation of the original function. The most commonly used techniques for constructing surrogate models include Kriging [31], neural networks [17], and polynomial regression [23, 32, 37]. A recent survey on fitness approximation in EAs can be found in [15].

One particularly interesting EA combining fitness approximation and local search is EANA (Evolutionary Algorithms with N-dimensional Approximation) [23], where polynomial regression was used to approximate fitness landscape. The experiments

of EANA on a wide range of test functions have demonstrated the effectiveness of this approach. Nevertheless, EANA was designed to locate a single global optimum (not multiple global optima), and test functions were all static test functions. To the best of our knowledge, no EAs using fitness approximation and local search have been tested for locating multiple global optima. It is even more difficult when tracking multiple moving peaks in a dynamic environment. This paper aims to develop a PSO incorporating fitness approximation and local search methods, and evaluate the effectiveness of the hybrid PSO using multimodal test functions in both static and dynamic environments.

11.2.2 Particle Swarms

Particle Swarm Optimisation (PSO) is an evolutionary algorithm that mimics a flock of birds [18]. As birds move throughout a territory, they are all simultaneously watching for both food and predators. In addition they are monitoring the behaviour of the birds around them. A change in a neighbour's behaviour usually indicates there is some new information available, for example a food source or predator has been seen. By copying the behaviour of its neighbours each bird is able to benefit from the discovery, even before it has the information itself. In PSO, each bird is represented by a particle. It maintains its current location and velocity, as well as a memory of the best location it has seen so far, known as the *personal best*. Each particle also has a number of neighbours with whom it can share its personal best. At every timestep each particle chooses a random point between its personal best location and the fittest personal best of any of its neighbours. It then steers towards that point, but does not travel there directly. The particles have momentum, meaning that if the chosen point is in the opposite direction to where they're travelling it may take a number of timesteps to turn around. This ensures the particles thoroughly explore the area surrounding the fittest known point, and are able to jump to a better peak if one is discovered nearby.

To guarantee convergence, we have used Clerc's constriction coefficient PSO [11, 18]. This is described by Equations (11.1) and (11.2), which are run for every timestep t.

$$v_{(i,j,t+1)} = \chi(v_{(i,j,t)} + \varphi_1(p_{(i,j,t)} - x_{(i,j,t)}) + \varphi_2(p_{(g,j,t)} - x_{(i,j,t)})) \tag{11.1}$$

$$x_{(i,j,t+1)} = x_{(i,j,t)} + v_{(i,j,t+1)} \tag{11.2}$$

where:

$$\varphi_1 = c_1 r_1, \quad \varphi_2 = c_2 r_2, \quad \chi = \frac{2\kappa}{\left|2 - c - \sqrt{c^2 - 4c}\right|} \tag{11.3}$$

The current location of particle i in dimension j at time t is represented as $x_{(i,j,t)}$, with the current velocity $v_{(i,j,t)}$. φ_1 and φ_2 act as random weightings for the

personal and neighbourhood bests, represented as $p_{(i,j,t)}$ and $p_{(g,j,t)}$ respectively. c_1 and c_2 are constants, usually set at 2.05, with $c = c_1 + c_2$. r_1 and r_2 are uniform random numbers in the range $[0, 1]$. Equation (11.3) calculates χ, a constant friction on the particles that prevents them from oscillating violently around an optimum. κ is usually set at 1.

11.2.3 Speciated Particle Swarms

Most PSO algorithms are limited in that they will only converge on a single solution, even when there are many global optima. Locating several solutions is beneficial in several ways. Firstly, it provides the user with a choice. While to the algorithm it may appear that two optima are of equal fitness, in reality it may be preferable to choose one over the other. In many environments there are factors that are too complex to incorporate into the fitness function. These factors are nevertheless present and may lead an expert user to choose one solution over another. Secondly, by simultaneously locating multiple solutions we reduce the risk of premature convergence, that is where the entire population becomes trapped in a local optimum.

Speciation, also known as niching, is one way to achieve this. The population is divided into species, which are groups of particles that are close to each other in the decision space. Communication between species is either severely limited or non-existant, allowing them to each explore their local area without interference from particles on distant peaks.

We have used SPSO [21] as our base algorithm to test the regression method. To determine the effectiveness on dynamic environments we will be using Moving Peaks, a well-known dynamic test function generator (see section 11.4.2). SPSO is an ideal candidate for this function for two reasons. Not only does it perform well on dynamic multimodal functions [29], its performance is also strongly correlated with its local convergence speed [6]. This shows that by increasing the local convergence speed, we should see a marked improvement in overall performance.

In SPSO, each species is defined by a hyper-spherical area of radius r. This area is centred on the locally-fittest particle, called the species seed. Any particles within the species area are considered to belong to that species, although they are free to leave should they move away. The particles within each species are connected using the global neighbourhood topology [19]. There is no communication between particles of different species. To allocate the particles to species, they are first sorted from fittest to least-fit. The list is then iterated through; for each particle, if there is a species seed within r of its location it joins that species. Otherwise it becomes the seed particle of a new species. If a particle is within r of two or more species seeds it is allocated to the species of the fittest seed, as shown in Fig. 11.1.

In the original SPSO algorithm [21], particles were allocated to species based on their current location. To improve species stability we have used the personal best location and fitness, as was done in [5].

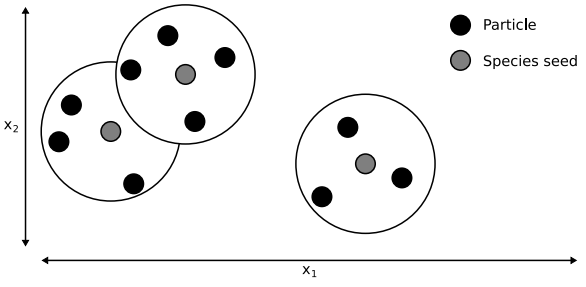


Fig. 11.1 The species seeds are the fittest particles in their area of the decision space. The middle seed is fitter than the one on the left, so its species area takes precedence where they overlap

11.2.4 Guaranteed Convergence PSO

The basic PSO model has an inefficiency, in that the velocity of the fittest particle will quickly drop to zero. This is caused by its personal and neighbourhood bests being on the same point. While this is not a big problem in the standard implementation, it can become noticeable when using speciated algorithms because the population of each species is often very low. Having an idle particle in a population of 20 is far less noticeable than in a population of 4. In order for the particle to start moving again, one of its neighbours must locate a better point. If there are only 2 or 3 particles this can take a long time, if it happens at all.

Guaranteed Convergence PSO [30] was developed to overcome this problem. Instead of travelling around like the rest of the population, the fittest particle randomly tries points within a distance d of its personal best location. GCPSO adaptively determines the value of d for each particle by tracking how many consecutive successful or unsuccessful tries there were. An attempt is considered successful if it improves on the personal best, otherwise it is a failure. If the number of consecutive successes exceeds a threshold, the algorithm searches more aggressively by doubling d. Likewise if the number of consecutive failures becomes too large d is halved so as to search in a smaller area. Combining SPSO with GCPSO means that the species seed will follow the GCPSO rules, ensuring that it never stops searching. The other particles follow the standard PSO implementation.

11.2.5 mQSO

To provide a benchmark to test SPSO's performance against, we have used one of the most effective PSO-based algorithms on Moving Peaks, mQSO [8]. mQSO modifies the standard PSO in several ways to improve performance on this function. These improvements are discussed below.

To track as many peaks as possible, mQSO divides the population into subswarms. These are equivalent to species in SPSO, except that particles are not free to join or leave a subswarm. If two subswarms become too close to each other, the weaker one will have its particles reinitialised. This prevents duplication and encourages the swarm to explore new areas. Stagnation is prevented by means of an anticonvergence measure. If all of the subswarms have converged to small areas, the weakest one is reinitialised in the same way as a duplicate subswarm. This prevents the system from wasting its resources on peaks of low fitness.

mQSO is used to increase the swarm's responsiveness to a peak movement. Rather than letting all of the particles tightly converge on the optimum, half of the particles are reserved as quantum particles. These particles do not follow the standard PSO movement equations; instead at each timestep they are placed randomly within a hypersphere of radius r_{cloud} using a uniform volume distribution. This technique is similar in some respects to GCPSO mentioned above.

11.3 Using Regression to Locate Optima

On a multimodal fitness landscape, around each peak there is a catchment area. Within this area, fitness generally improves as you get closer to the peak. If we can model the overall shape of the peak while ignoring the local features, we can calculate the highest point of that shape. Assuming that our model is reasonably accurate, the top of our shape should be close to the optimum. We can use regression to approximate the shape of the peak. Polynomial regression with least-squares approximation has been used to improve traditional optimisation methods [12, 35]. And more recently, regression was also employed to improve the performance of EAs [23, 32, 37].

In our proposed hybrid PSO using regression, we maintain a separate memory to the base algorithm, storing only the best locations and their fitnesses. If the base algorithm's memory was used, points would only remain known as long as there is an individual there. The regression needs to know the locations of the fittest points, regardless of the population's current state.

A minimum number of points is needed in order to calculate the regression – below this there will be more than one shape that fits the data. As the algorithm continues sampling the fitness landscape, the regression may keep some excess points to help reduce the effect of any local landscape features. The number of excess points e is a tunable, although robust, parameter.

By performing a linear least-squares regression on the known points and their fitnesses, we are able to estimate the peak's shape. From the regression we obtain a set of equations, one for each decision variable, that defines the shape that best fits our known points. Although more complex and flexible equations can be used if desired, for simplicity and efficiency we have used quadratic equations to represent the shape. This results in a set of simultaneous equations in the form of Equation (11.4) to be solved for $a_1, a_2, \ldots, a_{2n}$ and c, where n is the number of decision variables.

$$f(x_1,x_2,...,x_n) = a_1x_1^2 + a_2x_1 + a_3x_2^2 + a_4x_2 + \ldots + a_{(2n-1)}x_n^2 + a_{(2n)}x_n + c \tag{11.4}$$

In matrix form, the simultaneous equations look like:

$$\mathbf{A} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} x_{1,1}^2 & x_{1,1} & x_{1,2}^2 & x_{1,2} & \ldots & x_{1,n}^2 & x_{1,n} & 1 \\ x_{2,1}^2 & x_{2,1} & x_{2,2}^2 & x_{2,2} & \ldots & x_{2,n}^2 & x_{2,n} & 1 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ x_{m,1}^2 & x_{m,1} & x_{m,2}^2 & x_{m,2} & \ldots & x_{m,n}^2 & x_{m,n} & 1 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{2n} \\ c \end{bmatrix}$$

where there is an equation for each of the m known points. To solve the simultaneous equations, we manipulate the matrices as in Equation (11.5):

$$\begin{aligned} \mathbf{A} &= \mathbf{BC} \\ \mathbf{B}^+\mathbf{A} &= \mathbf{B}^+\mathbf{BC} \\ \mathbf{B}^+\mathbf{A} &= \mathbf{C} \end{aligned} \tag{11.5}$$

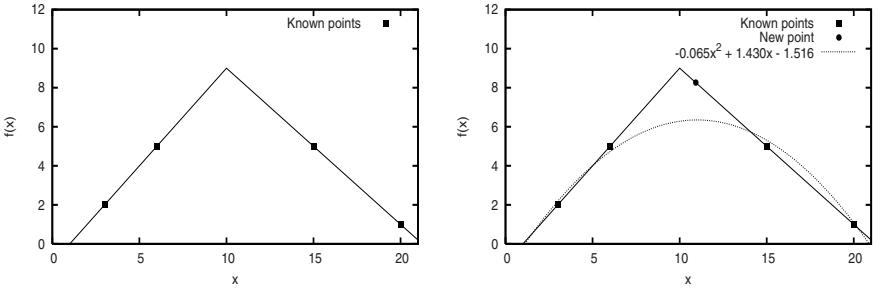
where $\mathbf{B}^+$ is the pseudoinverse of $\mathbf{B}$ [3]. If we only use the minimum number of points, $\mathbf{B}$ will be square and we can use the inverse $\mathbf{B}^{-1}$ instead. The minimum number of points required to perform an approximation is $2n+1$, according to Equation (11.4) (see also [23]). The coefficients that make our equation best match the known points are found by computing $\mathbf{C}$. We then find the turning point for the equation in each dimension $i = [1,n]$ by taking the partial derivative, as in Equation (11.6):

$$\frac{\partial}{\partial x_i} f(x_1,x_2,\ldots,x_n) = 2x_i a_{(2i-1)} + a_{(2i)} \tag{11.6}$$

The turning point in dimension i is where $\frac{\partial}{\partial x_i} f(x_1,x_2,\ldots,x_n) = 0$. To find out whether it is a maximum or minimum point, we take the second derivative. When using quadratic equations, we can simply look at the sign of $a_{(2i-1)}$; a negative number indicates that it is a maximum. If this is a maximisation problem and one of the equations has only a minimum turning point, we abort the regression and wait for better data. Similarly, if it is a minimisation problem we abort if any of the equations has no minimum turning point.

The global maximum point of the shape will be at the location of the turning point in each decision variable. Even though we were able to compute a maximum, we still need to check that it is valid. If the points do not give a good representation of the peak, for example they are all on one side, the regression will not be accurate. If the computed point is outside the expected area, or even the entire decision space, it is discarded. We will try the regression again when we have more data.

To test the calculated position, we replace the least-fit individual with a new individual at the shape's highest point. This avoids using an extra evaluation, and it is unlikely that the individual's next movement would have contributed much to the search. If the regression was successful, the new point will be used to further



(a) (b)

Fig. 11.2 a) Trying to find the highest point of the peak. We currently know the fitnesses of 4 points: 3, 6, 15 and 20. The right side of the peak is less steep than the left. b) The regression curve has a maximum at $x = 10.926$, considerably closer to the peak than any of the previously known points

refine the shape when it is next performed, hopefully improving the fitness still further. When using this technique with dynamic environments, we clear the memory whenever a peak movement is detected. This prevents the regression from being performed on stale data.

The main cost of this method is in performing the matrix inversion. Assuming the minimum number of points are used, this has a complexity of $\mathcal{O}(n^3)$. As n is dependent only on the number of decision variables and complexity of the equations used, the cost is usually quite low. The CPU cost can be further reduced by only performing the regression at certain intervals or only for the most promising peaks. In many environments, fitness evaluations are the most expensive aspect. The regression's minimal CPU overhead is usually far outweighed by the number of evaluations saved.

As an example we will try to solve a 1-dimensional triangular function, as shown in Fig. 11.2 a). Currently we know the fitnesses of 4 points:

$$\begin{aligned} f(3) &= 2 \\ f(6) &= 5 \\ f(15) &= 5 \\ f(20) &= 1 \end{aligned}$$

We place these values into $\mathbf{B}$ and $\mathbf{C}$:

$$\begin{bmatrix} 2 \\ 5 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} 3^2 & 3 & 1 \\ 6^2 & 6 & 1 \\ 15^2 & 15 & 1 \\ 20^2 & 20 & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ c \end{bmatrix}$$

Multiplying both sides by $\mathbf{B}^+$ gives:

$$\begin{bmatrix} 0.01 & -0.01 & -0.01 & 0.01 \\ -0.24 & 0.12 & 0.30 & -0.17 \\ 1.46 & 0.02 & -1.01 & 0.54 \end{bmatrix} \begin{bmatrix} 2 \\ 5 \\ 5 \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ c \end{bmatrix}$$

$$\begin{bmatrix} -0.065 \\ 1.430 \\ -1.516 \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ c \end{bmatrix}$$

This gives us the best-fitting quadratic curve, Equation (11.7).

$$f(x) = -0.065x^2 + 1.430x - 1.516 \tag{11.7}$$

To find the turning point, we differentiate it:

$$\frac{df(x)}{dx} = (-0.065)2x + 1.430 \tag{11.8}$$

Solving Equation (11.8) gives a turning point of $x = 10.926$. We know this point is the maximum because the x^2 coefficient is negative. The fitness at $x = 10.926$ is 8.2592. As can be seen from Fig. 11.2 b), this is not the location of the actual peak, however it is considerably closer than any of the points known so far.

There are similarities between EANA [23] and the technique presented here. Both methods use a polynomial regression to estimate the location of a peak. However there are also important differences, including the focus of the algorithms. EANA is designed to estimate the area of the global optimum, whereupon local search is used to refine its guess. As with all local search algorithms, premature convergence can be a problem. Our technique takes advantage of the fact that modern EAs are already very effective at locating the area of a peak. Instead we use the regression as a heuristic, guiding the base algorithm towards its goal as it explores the surrounding areas. This gives us the best of both worlds – substantially improving performance while only minimally increasing the risk of premature convergence.

Another important difference is the way the two techniques obtain an accurate model of the landscape. EANA assumes that the height of the local optima is correlated with their distance from the global optimum, and so spends evaluations searching for the local peaks. Our technique makes no such assumption, instead using more than the required number of points to compute its model. This allows it to better reflect the general trends of the landscape and discourages overfitting, without needing additional evaluations.

11.4 Experimental Setup

To determine whether performing the regression is effective, we compared the performance of SPSO and GCPSO with and without the regression. For the rest of the paper, we will use SPSO to mean SPSO + GCPSO, and rSPSO to mean SPSO +

GCPSO + regression. The regression has been implemented so as to discard any calculated solutions that are outside the species boundary, as described in Section 11.3.

For the purposes of the regression, we consider each species to be an individual subpopulation with its own memory. This means that for every timestep, there is a regression run for each species.

The regression will be tested on both static and dynamic multimodal test functions; we will describe our procedure below. For all of the tests, each species is limited to $P_{max} = 6$ particles; any excess particles are reinitialised elsewhere in the decision space. This is the same method as was used in [29] to avoid having too many individuals crowd an optimum. The success and failure thresholds for GCPSO have been set to the values recommended in [4], that is $s_c = 15$ and $f_c = 5$. Unless otherwise stated, the regression stores a maximum of $e = 10$ excess points. Each experiment was performed 50 times and the results have been averaged.

11.4.1 Static Functions

To test general performance in a static multimodal environment, we chose functions that represents several different landscape features. These functions are described below; the mathematical definitions are shown in Table 11.1.

- Inverted Branin RCOS (F1) and Himmelblau (F4) both have peaks with large catchment areas.
- Six-Hump Camel Back (F2) has two global optima with relatively large catchment areas, however there are also 4 local optima for the particles to become trapped in.
- Deb's 1st Function (F3) has 5 narrow narrow peaks; even though it is only 1 dimensional, it can be difficult to locate all of the optima.

Table 11.1 Static multimodal test functions

Function	r	Comments
Inverted Branin RCOS [9]: $F1(x,y) = -[(y - \frac{5x^2}{4\pi^2} + \frac{5x}{\pi} - 6)^2 + 10(1 - \frac{1}{8\pi})cos(x) + 10]$, where $-5 \le x \le 10$; $0 \le y \le 15$	4	3 global optima
Six-Hump Camel Back [25]: $F2(x,y) = -4[(4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (-4 + 4y^2)y^2]$, where $-1.9 \le x \le 1.9$; $-1.1 \le y \le 1.1$	1	2 global optima and 4 local optima
Deb's 1st Function [13]: $F3(x) = sin^6(5\pi x)$, where $0 \le x \le 1$	0.15	5 equally spaced global optima
Himmelblau [2]: $F4(x,y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$, where $-6 \le x,y \le 6$	3	4 global optima
Inverted Shubert 2D [20]: $F5(x,y) = -\sum_{i=1}^{5} i\cos[(i+1)x + i] \sum_{i=1}^{5} i\cos[(i+1)y + i]$, where $-10 \le x,y \le 10$	0.75	18 global optima in 9 clusters, many local optima

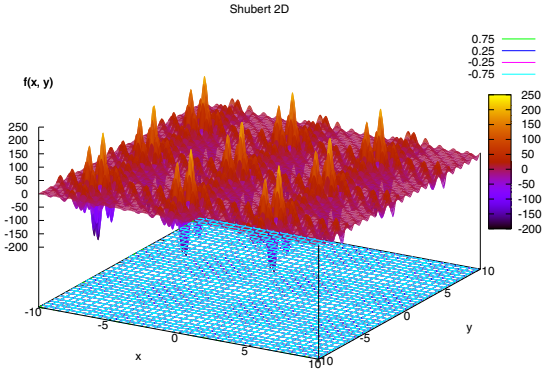


Fig. 11.3 The inverted Shubert 2D test function has 18 global optima located in 9 pairs, as well as many local optima

- Inverted Shubert 2D (F5) is the most difficult as it is highly multimodal. There are 18 global optima, all of which have extremely small catchment areas. In addition, there are numerous local optima as shown in Fig. 11.3.

A run is only considered successful if the algorithm locates all of the optima to within a fitness of $\varepsilon = 0.00001$ within 2000 timesteps.

For F1 through F4, a population size of 50 has been used. To reliably solve F5, SPSO requires at least 500 particles. With the exception of Deb's First function, all of these are two dimensional functions.

In addition to the test functions in Table 11.1, we also used a modified version of the Generic Hump Function proposed by Singh and Deb [33], to test the regression's performance in environments where there are a larger number of decision variables. The Hump function allows us to independently control the number of dimensions, the number of humps and the size and shape of those humps, making it ideal for our testing.

The Humps function consists of K humps rising from a flat surface. The height of the hump k at a given point X is shown in Equation (11.9):

$$f(x,k) = \begin{cases} h_k \left[1 - \left(\frac{d(X,k)^{\alpha_k}}{r_k}\right)\right], & \text{if } d(X,k) < r_k \\ 0, & \text{otherwise} \end{cases} \tag{11.9}$$

The distance from the centre of the hump k to X is denoted by $d(X,k)$. In Singh's testing, all of the humps have the same height, radius and shape; $h_k = 1, \alpha_k = 1$ for all k. The radius r_k was varied with the number of dimensions. The humps are placed so that they do not intersect, meaning that the distance between any two humps j and k is at least $r_j + r_k$. For our testing, we have removed the function's flat base so as to allow SPSO to more easily find the peaks, as shown in Equation (11.10):

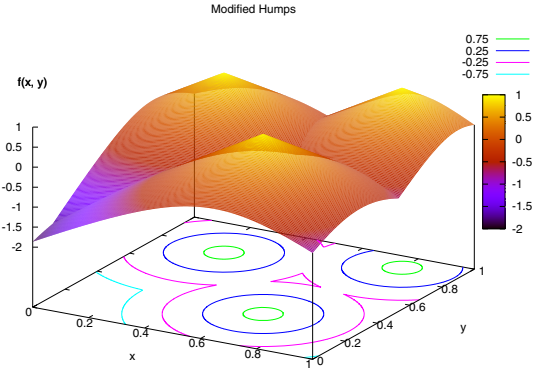


Fig. 11.4 An example of the Modified Humps landscape using 3 humps in two dimensions

$$f(x,k) = h_k \left[1 - \left(\frac{d(X,k)^{\alpha_k}}{r_k} \right) \right] \tag{11.10}$$

The height of any given point in the decision space is the height of the tallest hump at that point, as in Equation (11.11):

$$f(x) = \mathrm{Max}_{k=1}^{K} f(x,k) \tag{11.11}$$

An example of a landscape with 3 humps in two dimensions is shown in Fig. 11.4. We have tested this function with 20 peaks and 5, 10, 15 and 20 decision variables, with r_k set to 0.27, 0.37, 0.41 and 0.43 respectively. All of the tests used 300 particles and SPSO's r parameter was set to $2r_k$, that is twice the hump radius.

11.4.2 Moving Peaks

Moving Peaks is a highly configurable dynamic test function suite, and is a common benchmark for dynamic optimisation algorithms [10]. Moving Peaks is very similar to the Humps function described above, except that the peaks periodically move and change size. The peaks are usually conic, however in this paper we will be testing with other shapes as well.

The algorithm must track the peaks as they move around the fitness landscape. As the peaks move they also change height; this means that the globally optimal peak changes over time, as in Fig. 11.5. For this reason, algorithms that only track one or a few peaks tend to perform poorly; the results of a particular run depend more on the peak's average height than the algorithm's performance [6]. To achieve good performance it is critical that an algorithm tracks as many peaks as possible. It is too expensive to locate the new optimum from scratch after every landscape change. Following this rationale, it is expected that the good performance of an optimiser for

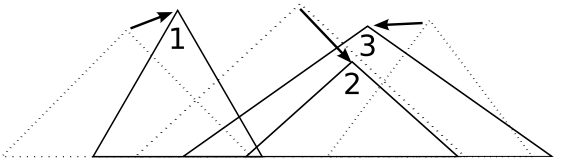


Fig. 11.5 The difficulty of tracking peaks: Peak 2 used to be the global optimum however it is now covered by Peak 3. At some point in the future Peak 2 may re-emerge and become the highest again; algorithms with insufficient population diversity are unlikely to rediscover this peak

a static multimodal environment should be transferable to the dynamic multimodal environment.

Unless otherwise stated, all Moving Peaks parameters are set as specified by scenario 2; please refer to Table 11.2 for details[1].

The most widely-used performance metric on Moving Peaks function is offline error [10]. Offline error is the difference in fitness between the best-known point and the global optimum, averaged over the entire run. All of our results on Moving Peaks will be presented in terms of offline error after 500,000 evaluations.

Both the PSO and regression algorithms need to be informed when the landscape changes. To detect these changes, at the end of each timestep we check the fitness of the top 5 species seeds. If any of the fitnesses differ from the recorded value, the particles' personal best memories and the memory used for the regression are both cleared.

To determine the robustness of the regression under different circumstances, we tested:

- The number of peaks between 1 and 200
- The severity of each peak movement, between 0 and 6
- The number of decision variables between 5 and 10

11.4.2.1 Creating Varying Peak Shapes

Extending our preliminary work in [7], we will fully analyse the regression's performance, including on more complex peak shapes, as defined in this section. We wanted to see whether the regression was still effective with other peak shapes. The standard conic shape used by Moving Peaks is produced by Equation (11.12):

$$f(x) = h - w\sqrt{\sum_{d=1}^{D}(x_d - p_d)^2} \tag{11.12}$$

[1] See also: http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/

Table 11.2 Moving Peaks Scenario 2 parameters

Parameter	Setting
Random seed	1
Dimensions	5
Peaks	10
Minimum peak height	30
Maximum peak height	70
Standard peak height	50
Minimum peak width	1
Maximum peak width	12
Standard peak width	0
Coordinate range	[0, 100]
Peak movement severity	1
Peak height severity	7
Peak width severity	1
Basis function	None
Movement correlation λ	0
Peak movement interval	5000 evaluations
Peak shape	Conic
Change stepsize	Constant

where w and h specify the width and height of the peak respectively, x_d is the location of the point in dimension d and p_d is the tip of the peak in dimension d. We have extended this equation in several ways:

- The relationship between height and distance from the peak's centre in dimension d is now controlled by α_d. $\alpha_d = 1$ will produce the conic shape used by the standard Moving Peaks scenarios. Using $\alpha_d > 1$ will produce a mound shape, with steepness increasing as the α_d gets bigger. Setting $\alpha_d \in (0, 1)$ produces a spike shape, as depicted in Fig. 11.6.
- To create asymmetric peaks, for each decision variable we divide the peak into 2 halves, left and right. The value of α_d used for the left and right halves are denoted α_{d0} and α_{d1} respectively. Fig. 11.7 shows an asymmetric peak.
- Local optima have been added by superimposing a cosine wave over the peak. The amplitude and frequency of the wave are specified by β_d and γ_d respectively. All γ values used in this paper are in radians. Fig. 11.8 shows a cosine wave superimposed on Fig. 11.7. By adjusting β_d and γ_d we can specify the number and severity of local peaks in variable d.

The new peak function is defined in Equation (11.13). The standard conic peak shape can be achieved by setting $\alpha_d = 1, \beta_d = 0, \gamma_d = 0$ for all d.

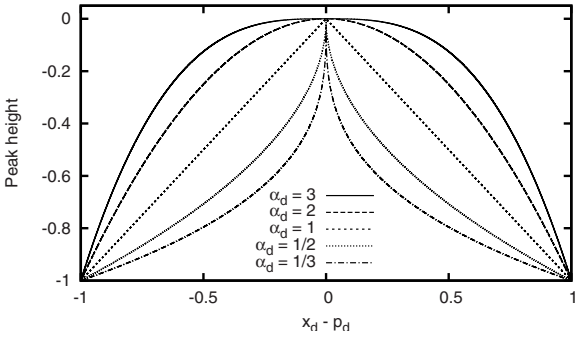


Fig. 11.6 Five peaks have been overlaid, showing how α_d determines the peak's shape

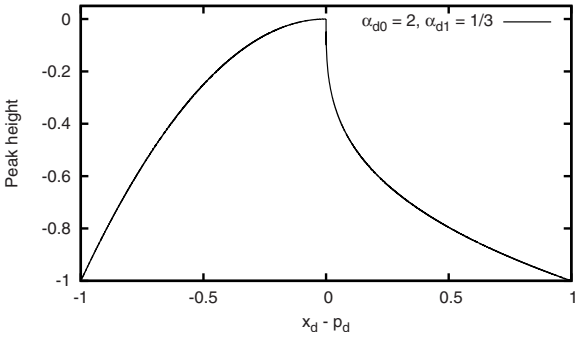


Fig. 11.7 An example of an asymmetric peak. The values of α_d are 2 and $\frac{1}{3}$ for the left and right halves respectively

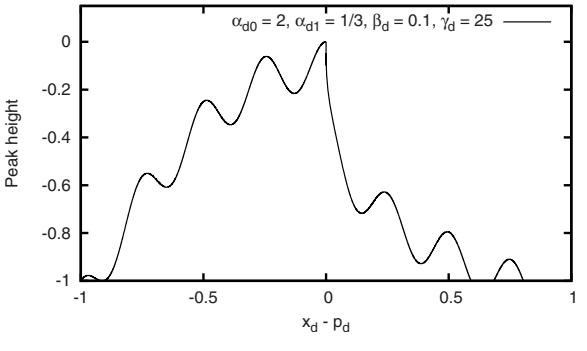


Fig. 11.8 Superimposing the cosine wave over the peak in Fig. 11.7

$$f(x) = h - w\sqrt{\sum_{d=1}^{D}\left(|x_d - p_d|^{\alpha_d} + u_d\right)^2} \tag{11.13}$$

where

$$\alpha_d = \begin{cases} \alpha_{d0} & \text{if } x_d \quad p_d \\ \alpha_{d1} & \text{otherwise} \end{cases} \quad u_d = \beta_d\left(cos\left[\gamma_d\left(x_d - p_d\right)\right] - 1\right) \quad \alpha_{d0}\ \alpha_{d1} \in (0\ \infty)\ \ \beta_d\ \gamma_d \in [0\ \infty)$$

To determine the regression's performance on varying peak types, the following tests were carried out:

- Symmetric peaks where all α_{d0} and α_{d1} values are equal. We tested with the following setting: $\alpha_d \in \left\{\frac{1}{3}, \frac{1}{2}, 1, 2, 3\right\}, \beta = \gamma = 0$.
- Asymmetric peaks where all α_d values are randomly generated within the range $[\frac{1}{3}, 3]$, and $\beta = \gamma = 0$.
- Conic peaks with a superimposed wave: $\alpha_d = 1, Max_\beta \in [2, 10], Max_\gamma \in [20, 100]$.
- Asymmetric peaks with a superimposed wave: $\alpha_{d0}, \alpha_{d1} \in [\frac{1}{3}, 3]$, $Max_\beta \in [2, 10]$, $Max_\gamma \in [20, 100]$.

Max_β indicates that each peak's values for β_d are randomly chosen in the range $[0, Max_\beta]$. Max_γ indicates the same thing for γ.

We also tested the regression's sensitivity to e, both on the standard scenario 2 problem and with the modified peak function, using $\alpha_d \in [\frac{1}{3}, 3], Max_\beta = 10, Max_\gamma = 100$. The latter is designed to show whether using a larger e value improves performance on functions with many local optima.

Finally we compared our results to mQSO, one of the best performing algorithms on Moving Peaks scenario 2. We have compared our results to the $10\,(5 + 5^q)$ configuration that Blackwell showed to be optimal on this problem. All experiments were run with 100 particles and with SPSO's r parameter set to 30. These were the settings used in [22].

11.5 Results

This section is divided into two main parts. We will first look at the regression's performance on static multimodal functions. Secondly we will analyse its behaviour on the Moving Peaks test suite.

11.5.1 Static Functions

Even with static environments, it is still very important to reduce the number of evaluations. Each evaluation costs CPU time, often well in excess of the time used by the optimiser itself. In this part we will report on the regression's performance on static problems, both in low dimensional and high dimensional environments. The

Table 11.3 Regression performance on low dimensional static functions. Mean, standard error and improvement over SPSO are shown

Function	rSPSO	SPSO	Improvement
F1	6254.54 (±298 59)	9552.86 (±216 98)	35%
F2	1489.63 (±53 90)	8934.58 (±304 97)	83%
F3	5306.60 (±225 23)	7613.16 (±271 40)	30%
F4	4963.74 (±277 75)	11069.68 (±299 91)	55%
F5	50511.46 (±794 94)	164360.00 (±2912 89)	69%

Table 11.4 Regression performance on the high dimensional Humps function. d is the number of decision variables

d	rSPSO	SPSO	Improvement
5	191902.50 (±5353 82)	292472.57 (±12638 88)	34%
10	256795.42 (±2506 12)	356665.10 (±5146 87)	28%
15	277200.14 (±2038 94)	404640.00 (±1161 45)	31%
20	297978.00 (±1427 11)	462036.00 (±1216 41)	36%

third part of this section will investigate how the number of excess points affects performance.

11.5.1.1 Low Dimensional Landscapes

Table 11.3 shows that using the regression dramatically increased performance on all of the low dimensional functions we tested. The largest improvement was on Branin RCOS, where the number of evaluations required was reduced by more than 80%. Even on Deb's First Function (F3), the regression still reduced the number of evaluations by nearly a third. This is a significant improvement.

Inverted Shubert 2D (depicted in Fig. 11.3) was by far the hardest 2 dimensional function we tested, normally requiring 160,000 evaluations for SPSO to locate all of the optima. Adding the regression reduced this to just 50,000. We suspect that one of the reasons the regression performed so well is that the peak tips resemble a parabola, allowing it to be accurately modelled.

11.5.1.2 High Dimensional Landscapes

The number of dimensions does not appear to affect the regression's effectiveness. Table 11.4 shows that using the regression reduced the number of evaluations by about 30%. This becomes especially signficant as the number of dimensions

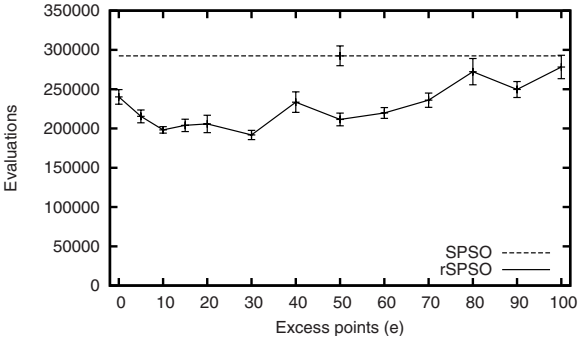


Fig. 11.9 Number of evaluations needed for Humps 5D with different numbers of excess points. The vertical bars show one standard error

increases; for the 20 dimensional function using the regression saved over 160,000 evaluations.

11.5.1.3 Sensitivity to *e*

As Fig. 11.9 shows, the best performance was obtained by setting *e* to between 10 and 30. Although it still beat SPSO, the regression performed relatively poorly when excess points were not kept. The Humps function's peaks are conic, making them difficult to model when only using the minimum number of points. The excess points help to define the surface better, improving the regression's guess.

Using too many points also decreased performance. The regression does not perform any weighting – it tries to match all of the points regardless of their fitness. By storing too much data, we allow the memory to become polluted with distant and poor quality points. Instead of just modelling the tip, the peak's overall shape is matched. We are then unable to accurately determine the tip's location as our model is not specific to that area.

We recommend that *e* be set to 10 for all problems. This value represents a good tradeoff; it provides excellent performance without creating too much CPU or memory overhead. We have also tested this on the Moving Peaks problem and found the same ideal value. This is discussed in the next section.

11.5.2 Moving Peaks

Reducing the number of evaluations is critical when working with dynamic environments. If the environment is changing every 2 minutes, an algorithm that takes 3 minutes to find an adequate solution is useless. By adding the regression we are able to significantly reduce the number of evaluations needed; this section details the performance on Moving Peaks under different situations.

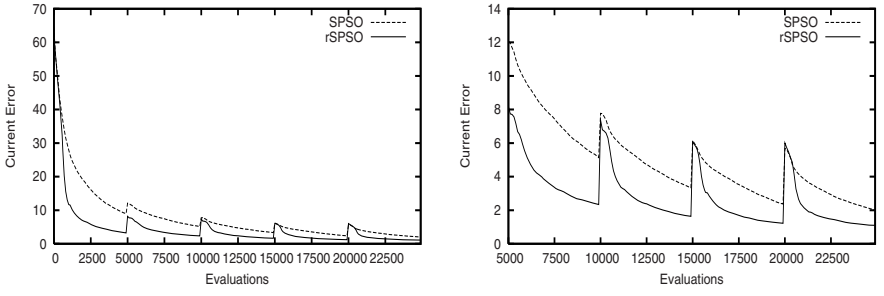


Fig. 11.10 Current error over time on Moving Peaks scenario 2; a) showing the effect of adding the regression; b) the same as a) but without the first 5000 evaluations

11.5.2.1 Increasing Convergence Speed

Fig. 11.10 a) compares the current error over time for SPSO and rSPSO. Current error is the difference in fitness between the best known point and the global optimum. Offline error is calculated by averaging the current error over an entire run.

In scenario 2, the peaks move every 5000 evaluations, as can be seen by the upwards jumps in the graphs. The regression is inactive for at least the first 200 evaluations after a peak movement. As the population size is 100, this represents only two timesteps; improvements here are mostly due to the fortunate placement and existing momentum of the particles. The regression cannot be used yet because there are insufficient points for it to be computed. For a 5-dimensional function, 11 points are needed. Since each species is limited to 6 particles, at least two timesteps needed to collect the required data.

Fig. 11.10 b) is the same as a), but without the first 5000 evaluations. This gives a better indication of the regression's performance helping to track the peaks. After a peak movement the PSO requires 100 evaluations to re-evaluate the particles, one for each individual. Thus the indicated error immediately after a change is not representative of the individuals' true fitnesses.

At 300 evaluations after a peak movement the regression's effect becomes obvious. The curve for rSPSO drops quickly as the algorithm hones in on the optimum. After a landscape change, a normal PSO must wait for the particles to accelerate towards the new peak, then wait for them to converge again once the peak has been located. The combination of GCPSO and the regression reduces the time spent doing this: the regression moves the worst particle close to the peak while GCPSO's rules set the velocity to 0 and force it to explore the local area.

At 1200 evaluations after a peak movement rSPSO has already achieved the same error as SPSO does after 4900 evaluations. Even at this point, the curve has a fairly steep gradient; substantial improvements are being made each timestep. The graph plateaus around 3000 evaluations; the error at this point is very small. The curve does not converge to 0 because SPSO is usually unable to maintain species on all of

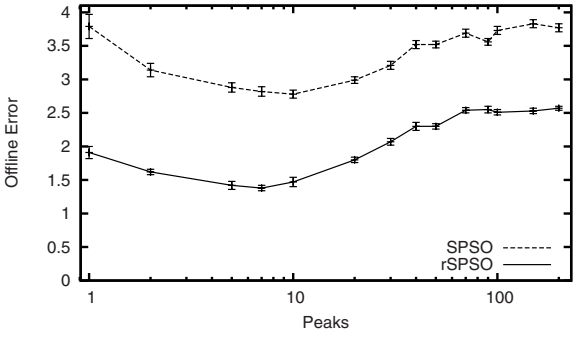


Fig. 11.11 Adding the regression substantially reduced offline error

the optima [6]; the residual error is from the times that the algorithm is not tracking the highest peak.

Please note that to ensure a smooth curve for Fig. 11.10, we have performed 1000 runs for each of the algorithms. All other results presented in this paper are based on experiments of 50 runs.

11.5.2.2 Number of Peaks

Adding the regression reduced offline error by between 1 and 1.5, as shown in Fig. 11.11. The settings of SPSO ($r = 30, P_{max} = 6$) are optimised for 10 peaks, which explains the sweet spot at that point. Below this, there are too many particles for the number of peaks. Since only 6 particles are allowed on each peak, when there are too few peaks most of the particles are continually reinitialised. This wastes evaluations, resulting in a larger offline error.

At the other end of the graph it becomes impossible for the algorithm to track all of the peaks. Having neither enough particles nor a small enough r value, the algorithm must rely on the particles to jump between peaks, hopefully to the globally optimal one. The increased error is caused by the times the algorithm is unable to discover the best peak.

11.5.2.3 Peak Movement Severity

The peak movement severity controls how far the peak moves each time. Larger severities increase the time needed to re-find the peak. The further the peaks move, the faster the particles will be travelling when they reach it. In a standard PSO they will then take longer to slow down and reconverge. Using the regression in combination with GCPSO helps this process; whenever the regression's guess is successful, the worst particle will become the species seed as the regression's guess was better than any of the existing solutions. Since the species seed follows the GCPSO movement rules, it immediately loses the momentum it previously had.

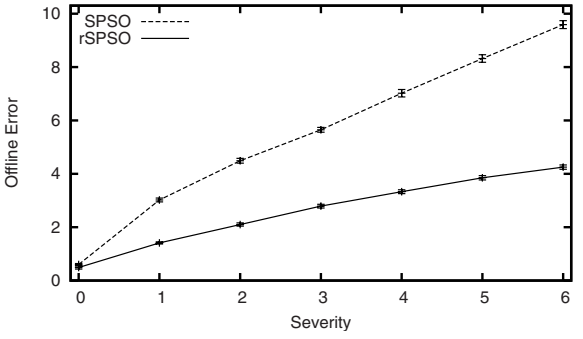


Fig. 11.12 The benefit of the regression increases with severity

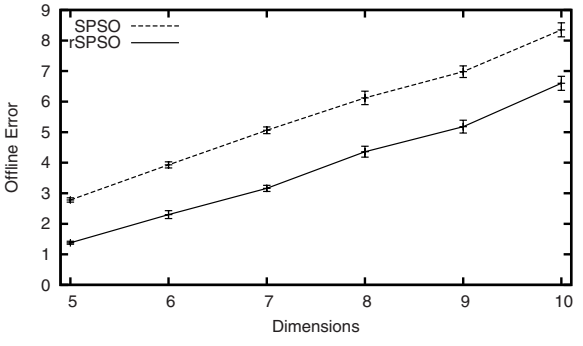


Fig. 11.13 The regression reduced the offline error by around 1.5 for all of the dimension values tested

As this happens to successive particles, the average velocity is quickly reduced, allowing the particles to reconverge.

Fig. 11.12 shows how SPSO's error increases linearly with severity. rSPSO's error also increases, but at a much lower rate. At a severity of 0 the peaks do not move at all, they only change height. In this situation the error achieved depends almost entirely on being able to track all of the peaks. As the peaks are not moving, once the optima have been initially located the regression is not needed to track them, thus the performances of SPSO and rSPSO are very similar.

11.5.2.4 Dimensionality

As with the results shown in Table 11.4 for the Hump function, performance improvement provided by the regression on the Moving Peaks is fairly constant. The error increases linearly as the number of dimensions increases, however the difference between rSPSO and SPSO remains about 1.5, as shown in Fig. 11.13.

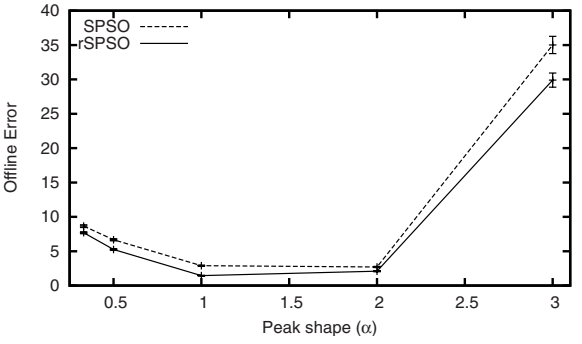


Fig. 11.14 Offline error for different peak shapes

11.5.2.5 Peak Shape

For this experiment, all of the α_d values have been set identically, making the shapes tested symmetrical. A one dimensional peak for each value of α_d is depicted in Fig. 11.6.

The most obvious feature of Fig. 11.14 is that outside the range $\alpha_d \in [1,2]$ there is a large increase in offline error.

For $\alpha_d > 2$, the areas away from the optima are exceptionally steep. Performance depends almost entirely on how quickly the algorithm can locate the general area of the tip, rather than its exact location. Guiding the PSO's search, the regression is able to reduce the offline error from 35 to 30.

When $\alpha_d < 1$, the penalty for being far away from the peak's tip is relatively small, however to achieve a small error it is extremely important to precisely locate the optimum. The difference between a point 0.1 units away from the peak and 0.2 units away can be substantial. Again, using the regression reduced the offline error by about 1. This is quite impressive as the actual peak shape is the opposite of the regression's model – the fitness landscape does not fit a parabola at all. This result suggests the parabolic model works well even on difficult peak shapes, and that more elaborate models are generally unnecessary. Further testing would be required to confirm this though.

11.5.2.6 Asymmetric Peaks

In the real world, many fitness landscapes have asymmetrical peaks. The peak may be very steep on one or more sides, or be at the edge of the feasible region. For these experiments, we have used random peak shapes. The α_d values for each side of every peak in each dimension are chosen randomly within a specified range. For example, in a run with only two peaks, the following values may be chosen:

$$\alpha_{00} = 0.40, \alpha_{01} = 0.94, \alpha_{10} = 2.64, \alpha_{11} = 0.58$$

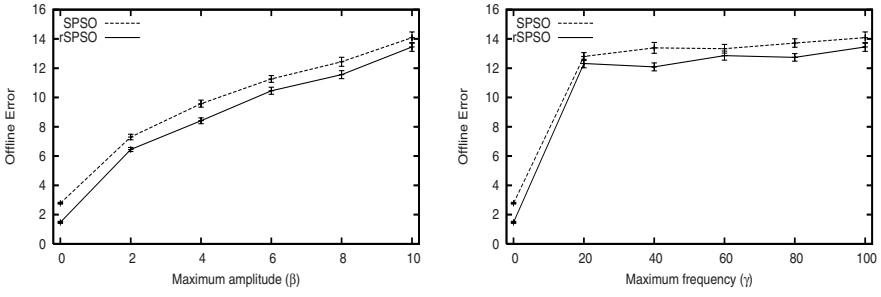
Table 11.5 Regression performance on asymmetric peaks

α_d	rSPSO	SPSO	Improvement
$[0.33, 3]$	1.82 ($\pm$0.07)	2.59 ($\pm$0.09)	30%
$[0.4, 2.5]$	1.70 ($\pm$0.07)	2.46 ($\pm$0.07)	31%
$[0.5, 2]$	1.66 ($\pm$0.06)	2.67 ($\pm$0.07)	38%
$[0.67, 1.5]$	1.58 ($\pm$0.05)	2.66 ($\pm$0.09)	41%
$[1, 1]$	1.45 ($\pm$0.05)	2.90 ($\pm$0.08)	50%

As Table 11.5 shows, the regression is most effective when the range of α_d is small. The regression works better on peaks that are roughly symmetrical as they more closely match the parabolic shape used. However even when highly asymmetrical peaks are created, the regression still achieved a 30% improvement over SPSO. This shows again that the regression's guesses are still accurate enough to aid the optimisation, even when it cannot closely model the peak.

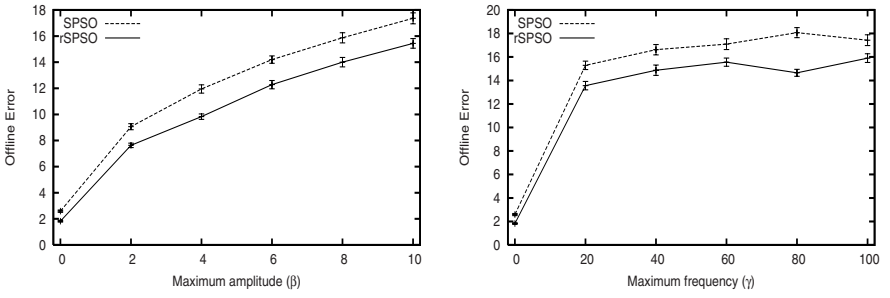
11.5.2.7 Adding Local Optima

By comparing Fig. 11.15 a) and b) we can see that the wave's amplitude had a much greater effect on performance than its frequency. The flatness of Fig. 11.15 b) suggests that the swarm is able to jump from peak to peak with relative ease. The amplitude's effect is far greater because each new candidate solution is just as likely to be at the bottom of the wave as the top. On average each new point will be halfway down a wave, increasing the overall error incurred. The local optima decrease the accuracy of the regression's model, reducing its effectiveness. Even so, it still managed to reduce the offline error by around 1 in all of the runs.



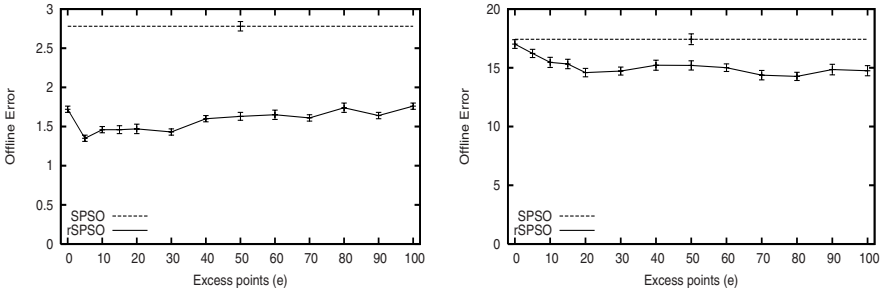
(a) (b)

Fig. 11.15 Performance on symmetric peaks for different values of: a) Max_β ($\alpha_d = 1, Max_\gamma = 100$); b) different values of Max_γ ($\alpha_d = 1, Max_\beta = 10$)



(a) (b)

Fig. 11.16 Performance on asymmetric peaks for different values of: a) Max_β ($\alpha_d \in [0.33, 3]$, $Max_\gamma = 100$); b) Max_γ ($\alpha_d \in [0.33, 3]$, $Max_\beta = 10$)



(a) (b)

Fig. 11.17 a) Offline error for Moving Peaks Scenario 2 for different values of e. b) Offline error for asymmetric peaks with waves for different values of e ($\alpha_d \in [0.33, 3], Max_\beta = 10, Max_\gamma = 100$)

11.5.2.8 Asymmetric Peaks with Local Optima

This is the most challenging landscape for the PSO; we are creating peaks that look similar to the one shown in Fig. 11.8. In all of the experiments the regression reduced the offline error by between 1.5 and 2. As can be seen by comparing Fig. 11.16 a) and b) with the results for the asymmetric peaks and local optima individually, the local optima are the primary cause of the large offline error - the asymmetric peaks are not a significant component. As would be expected, the results here are very similar to the results for the local optima tests.

11.5.2.9 Sensitivity to e

On scenario 2 the value of e does not greatly affect the regression's performance. Low, nonzero values provided slightly better results for the same reason as before,

Table 11.6 Comparing against mQSO: Severity

s	mQSO	rSPSO	SPSO
0	1.18 (±0.07)	0.49 (±0.06)	0.60 (±0.04)
1	1.75 (±0.06)	1.41 (±0.04)	3.02 (±0.07)
2	2.40 (±0.06)	2.10 (±0.06)	4.49 (±0.09)
3	3.00 (±0.06)	2.79 (±0.07)	5.65 (±0.09)
4	3.59 (±0.10)	3.33 (±0.07)	7.02 (±0.14)
5	4.24 (±0.10)	3.85 (±0.08)	8.32 (±0.14)
6	4.79 (±0.10)	4.25 (±0.08)	9.59 (±0.15)

Table 11.7 Comparing against mQSO: Number of peaks

Peaks	mQSO	rSPSO	SPSO
1	5.07 (±0.17)	1.91 (±0.09)	3.79 (±0.18)
2	3.47 (±0.23)	1.62 (±0.04)	3.14 (±0.10)
5	1.81 (±0.07)	1.42 (±0.06)	2.88 (±0.07)
7	1.77 (±0.07)	1.38 (±0.04)	2.82 (±0.07)
10	1.80 (±0.06)	1.47 (±0.07)	2.78 (±0.06)
20	2.42 (±0.07)	1.80 (±0.04)	2.99 (±0.05)
30	2.48 (±0.07)	2.07 (±0.05)	3.21 (±0.06)
40	2.55 (±0.07)	2.30 (±0.06)	3.52 (±0.06)
50	2.50 (±0.06)	2.30 (±0.04)	3.52 (±0.05)
100	2.36 (±0.04)	2.51 (±0.04)	3.73 (±0.06)
200	2.26 (±0.03)	2.57 (±0.03)	3.77 (±0.06)

by allowing the regression to concentrate on the best information. In all cases the regression outperformed SPSO by a significant margin, as shown in Fig. 11.17 a).

When optimising the most complex function, Moving Peaks with $\alpha_d \in [0.33, 3]$, $\beta = 10, \gamma = 100$, the value of e played a larger role in performance (Fig. 11.17 b)). As would be expected, increasing e slightly helped the regression ignore the local optima. Values larger than 20 gave no extra advantage however, showing that even for difficult problems only a few excess points are needed.

11.5.2.10 Comparing to mQSO

In Table 11.6 we compare the performance against mQSO (with the anticonvergence measure) for differing movement severities. As can be seen, rSPSO exceeds mQSO's performance for all of the severities tested. This is even more impressive considering that for most of the experiments SPSO had far worse performance

than mQSO. It should also be noted that both mQSO and SPSO have been tuned for this benchmark - the parameters chosen by Blackwell were optimised for each severity setting. SPSO's r has been set to the standard value for the Moving Peaks benchmark. The only parameter specifically related to the regression, e, requires very little tuning. The value of 10 was shown to be either optimal or near-optimal for all of the tests we conducted.

Table 11.7 compares mQSO's performance against both SPSO and rSPSO for differing numbers of peaks. It can be seen that rSPSO is highly competitive with mQSO; offering better performance for all but the 100 and 200 peak runs. It should be remembered that the regression can be added to most numerical optimisation algorithms; it is highly likely that it could be used to improve mQSO's performance even further.

11.6 Conclusion

In this chapter we have presented a technique to incorporate regression into a PSO algorithm, in order to improve local convergence. We have provided experimental studies and analysis of results using several multimodal test functions with varying difficulty. We have also extended the Moving Peaks test suite with more complex peak shapes, and carried out experimental studies on these newly defined test functions. Our results show that the performance of the regression-based SPSO (rSPSO) compares favorably against two existing multimodal PSOs (SPSO and mQSO). In particular, adding the regression significantly improved the performance of an existing multimodal PSO algorithm (SPSO) on a range of fitness landscapes in both static and dynamic environments. By using the existing population members, the regression technique does not require any additional evaluations, but only a modest amount of memory and CPU time depending on the number of decision variables and excess points.

As a future research direction it may be worthwhile exploring other deterministic techniques that could be combined with a PSO. Currently much of the available information is thrown away as the population moves each generation. By retaining and analysing this data, it is likely that further improvements in performance can be found.

References

1. Barthelemy, J.F.: Approximation concepts for optimimum structural design - a review. Structural Optimization 5, 129–144 (1993)
2. Beasley, D., Bull, D., Martin, R.: A sequential niche technique for multimodal function optimization. Evolutionary Computation 1(2), 101–125 (1993)
3. Ben-Israel, A., Greville, T.: Generalized Inverses: Theory and Applications. Wiley-Interscience, New York (1974)
4. van den Bergh, F., Engelbrecht, A.: A new locally convergent particle swarm optimiser. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 3, pp. 96–101. IEEE Press, Hammamet (2002)

5. Bird, S., Li, X.: Enhancing the robustness of a speciation-based PSO. In: Proceedings of the Congress on Evolutionary Computation, pp. 843–850. IEEE Press, Vancouver (2006)
6. Bird, S., Li, X.: Informative performance metrics for dynamic optimisation problems. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 18–25. ACM Press, London (2007)
7. Bird, S., Li, X.: Using regression to improve local convergence. In: Proceedings of the Congress on Evolutionary Computation, pp. 1555–1562. IEEE Press, Singapore (2007)
8. Blackwell, T., Branke, J.: Multi-swarm optimization in dynamic environments. Applications of Evolutionary Computing, 489–500 (2004)
9. Branin, F.: Widely convergent method for finding multiple solutions of simultaneous nonlinear equations. IBM Journal of Research and Development 16(5), 504–522 (1972)
10. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the Congress on Evolutionary Computation, vol. 3, pp. 1875–1882. IEEE Press, Washington (1999)
11. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6, 58–73 (2002)
12. Conn, A., Toint, P.: An algorithm using quadratic interpolation for unconstrained derivative free optimization. Nonlinear Optimization and Applications, 27–47 (1996)
13. Deb, K., Goldberg, D.: An investigation of niche and species formation in genetic function optimization. In: Proceedings of the International Conference on Genetic Algorithms, pp. 42–50. Morgan Kaufmann, San Francisco (1989)
14. Gottfried, B.: Introduction to Optimization Theory. Prentice-Hall, Englewood Cliffs (1973)
15. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing-A Fusion of Foundations, Methodologies and Applications 9(1), 3–12 (2005)
16. Jin, Y., Olhofer, M., Sendhoff, B.: Managing approximate models in evolutionary aerodynamic design optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, vol. 1, pp. 592–599 (2001)
17. Jin, Y., Olhofer, M., Sendhoff, B.: A framework for evolutionary optimization with approximate fitness functions. IEEE Transactions on Evolutionary Computation 6(5), 481–494 (2002)
18. Kennedy, J., Eberhart, R.: Swarm intelligence. Morgan Kaufmann, San Francisco (2001)
19. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of the Congress on Evolutionary Computation, pp. 1671–1676. IEEE Press, Honolulu (2002)
20. Li, J., Balazs, M., Parks, G., Clarkson, P.: A species conserving genetic algorithm for multimodal function optimization. Evolutionary Computation 10(3), 207–234 (2002)
21. Li, X.: Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 105–116. Springer, Heidelberg (2004)
22. Li, X., Branke, J., Blackwell, T.: Particle swarm with speciation and adaptation in a dynamic environment. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 51–58. ACM Press, Seattle (2006)
23. Liang, K.H., Yao, X., Newton, C.: Evolutionary search of approximated n-dimensional landscapes. International Journal of Knowledge-Based Intelligent Engineering Systems 4(3), 172–183 (2000)

24. Mahinthakumar, G., Sayeed, M.: Hybrid genetic algorithm - local search methods for solving groundwater source identification inverse problems. Journal of Water Resources Planning and Management 131(1), 45–57 (2005)
25. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, New York (1996)
26. Michalewicz, Z., Fogel, D.: How to Solve It: Modern Heuristics. Springer, Berlin (2000)
27. Neumaier, A.: Molecular modeling of protein and mathematical prediction of protein structures. SIAM Review 39(3), 407–460 (2002)
28. Ong, Y., Nair, P., Keane, A.: Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA Journal 41(4), 687–696 (2003)
29. Parrott, D., Li, X.: Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Transactions on Evolutionary Computation 10(4), 440–458 (2006)
30. Peer, E., van den Bergh, F., Engelbrecht, A.: Using neighbourhoods with the guaranteed convergence PSO. In: Proceedings of the IEEE Swarm Intelligence Symposium, pp. 235–242. IEEE Press, Indianapolis (2003)
31. Ratle, A.: Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 87–96. Springer, Heidelberg (1998)
32. Runarsson, T.: Ordinal Regression in Evolutionary Computation. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 1048–1057. Springer, Heidelberg (2006)
33. Singh, G., Deb, K.: Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1305–1312. ACM Press, Seattle (2006)
34. Vesterstrom, J., Riget, J., Krink, T.: Division of labor in particle swarm optimisation. In: Proceedings of the Congress on Evolutionary Computation, vol. 2, pp. 1570–1575. IEEE Press, Honolulu (2002)
35. Winfield, D.: Function minimization by interpolation in a data table. Journal of the Institute of Mathematics and its Applications 12, 339–347 (1973)
36. Yin, P., Yu, S., Wang, P., Wang, Y.: A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems. Computer Standards & Interfaces 28(4), 441–450 (2006)
37. Zhou, Z., Ong, Y., Nguyen, M., Lim, D.: A study on polynomial regression and Gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm. In: Proceedings of the 2005 Congress on Evolutionary Computation, vol. 3, pp. 2832–2839. IEEE Press, Los Alamitos (2005)

Part II
Techniques for High-Dimensional Problems

Chapter 12
Differential Evolution with Scale Factor Local Search for Large Scale Problems

Andrea Caponio, Anna V. Kononova, and Ferrante Neri

Abstract. This chapter proposes a novel algorithm for handling high dimensionality in large scale problems. The proposed algorithm, here indicated with Differential Evolution for Large Scale problems (DELS) is a Differential Evolution (DE) based Memetic Algorithm with self-adaptive control parameters and automatic population size reduction, which employs within its framework a variation operator local search. The local search algorithm is applied to the scale factor in order to generate high quality solutions. Due to its structure, the computational cost of the local search is independent on the number of dimensions characterizing the problem and thus is a suitable component for large scale problems. The proposed algorithm has been compared with a standard DE and two other modern DE based metaheuristics for a varied set of test problems. Numerical results show that the DELS is an efficient and robust algorithm for highly multivariate optimization, and the employment of the local search to the scale factor is beneficial in order to detect solutions with a high quality, convergence speed and algorithmic robustness.

12.1 Introduction

Computationally expensive optimization problems can be classified into two categories: problems which require a long calculation time for each objective function evaluation and problems which require a very high amount of objective function evaluations for detecting a reasonably good candidate solution. The problems belonging to the latter category, which are the focus of this chapter, are usually

Andrea Caponio
Department of Electrotechnics and Electronics, Technical University of Bari, Italy
e-mail: caponio@deemail.poliba.it

Anna V. Kononova
Centre for CFD, School of Process, Environmental and Materials Engineering, University of Leeds, LS2 9JT, UK
e-mail: pmak@leeds.ac.uk

Ferrante Neri
Department of Mathematical Information Technology, University of Jyvaškylä, FI-40014 Jyväskylä, Finland
e-mail: ferrante.neri@jyu.fi

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 297–323.
springerlink.com

characterized by a vast decision space which is strictly related to the high dimensionality of the function. Optimization problems characterized by a high number of variables are also known as large scale optimization problems, or briefly Large Scale Problems (LSPs).

The detection of an efficient solver for LSPs can be a very valuable achievement in applied science and engineering since in many applications a high number of design variables may be of interest for an accurate problem description. For example, in structural optimization an accurate description of complex spatial objects might require the formulation of a LSP; similarly such a situation also occurs in scheduling problems, see [19]. Another important example of a class of real-world LSPs is the inverse problem chemical kinetics studied in [13] and [14].

Unfortunately, when an exact method cannot be applied, LSPs can turn out to be very difficult to solve. As a matter of fact, due to high dimensionality, algorithms which perform a neighborhood search (e.g. Hooke-Jeeves Algorithm) might require an unreasonably high number of fitness evaluations at each step of the search while population based algorithms are likely to either prematurely converge to suboptimal solutions, or stagnate due to an inability to generate new promising search directions. In other words, many metaheuristics that perform well for problems characterized by a low dimensionality, e.g. Evolutionary Algorithms (EAs), can often fail to find good near optimal solutions to high-dimensional problems. The deterioration in the algorithmic performance, as the dimensionality of the search space increases is commonly known as a "curse of dimensionality", see [38].

Since the employment of optimization algorithms can lead to a prohibitively high computational cost of the optimization run without the detection of a satisfactory result, it is crucially important to detect an algorithmic solution that allows good results by performing a relatively low amount of objective function evaluations. In the literature various studies have been carried out and several algorithmic solutions have been proposed. In [15], a modified Ant Colony Optimizer (ACO) has been proposed for large scale optimization problems. Some other papers propose a technique, namely cooperative coevolution, originally defined in [27] and subsequently developed in other works, see e.g. [17] and [36]. The concept of the cooperative coevolution is to decompose a LSP in a set of low dimension problems which can be separately solved and then recombined in order to compose the solution of the original problem. It is obvious that if the objective function (fitness function) is separable then the problem decomposition can be trivial while for nonseparable functions the problem decomposition can turn out to be a very difficult task. However, some techniques for performing the decomposition of nonseparable functions have been developed, see [26]. Recently, cooperative coevolution procedures have been successfully integrated within Differential Evolution (DE) frameworks for solving LSPs, see [35], [39], [41], [24] and [40].

It should be remarked that a standard DE can be inefficient for solving LSPs, see [6]. However, DE framework, thanks to its simple structure and flexibility, can be easily modified and become an efficient solver of high dimensional problems. Besides the examples of DE integrating cooperative coevolution, other DE based algorithms for LSPs have been proposed. In [30] the opposition based technique is proposed for

handling the high dimensionality. This technique consists of generating extra points, that are symmetric to those belonging to the original population, see details in [31]. In [22] a Memetic Algorithm (MA) (see for the definitions e.g. [20], [12], and [25]) which integrates a simplex crossover within the DE framework has been proposed in order to solve LSPs, see also [23]. In [5], on the basis of the studies carried out in [1], [4], and [2], a DE for LSPs has proposed. The algorithm proposed in [5] performs a probabilistic update of the control parameter of DE variation operators and a progressive size reduction of the population size. Although the theoretical justifications of the success of this algorithm are not fully clear, the proposed approach seems to be extremely promising for various problems. In [21], a memetic algorithm which hybridizes the self-adaptive DE described in [2] and a local search applied to the scale factor in order to generate candidate solutions with a high performance has been proposed. Since the local search on the scale factor (or scale factor local search) is independent on the dimensionality of the problem, the resulting memetic algorithm offered a good performance for relatively large scale problems, see [21].

This chapter proposes a novel memetic algorithm which integrates the potential of the scale factor local search within the self-adaptive DE with automatic reduction of the population size in order to guarantee a high performance, in terms of convergence speed and solution detection, for large scale problems.

The rest of this chapter is organized in the following way. Section 12.2 describes the algorithmic components characterizing the proposed algorithm. Section 12.3 shows the numerical results and highlights the performance of the proposed algorithm with respect to a standard DE and two modern DE variants. Section 12.4 gives the conclusion of this work.

12.2 Differential Evolution for Large Scale Problems

This section describes the algorithmic components composing the proposed algorithm, namely Differential Evolution for Large Scale Problems (DELS), and their combination.

In order to clarify notation used throughout this article we refer to the minimization problem of an objective function $f(x)$, where x is a vector of n design variables in a decision space D.

12.2.1 Differential Evolution

According to its original definition given in [37], the DE consists of the following steps. An initial sampling of S_{pop} individuals is executed pseudo-randomly with a uniform distribution function within the decision space D. At each generation, for each individual x_i of the S_{pop} available, three individuals x_r, x_s and x_t are randomly extracted from the population. According to DE logic, a provisional offspring x'_{off} is generated by mutation as:

$$x'_{off} = x_t + F(x_r - x_s) \tag{12.1}$$

where $F \in [0, 1+[$ is a scale factor which controls the length of the exploration vector $(x_r - x_s)$ and thus determines how far from point x_i the offspring should be

generated. With $F \in [0, 1+[$, it is here meant that the scale factor should be a positive value which cannot be much greater than 1, see [28]. While there is no theoretical upper limit for F, effective values are rarely greater than 1.0. The mutation scheme shown in eq. (12.1) is also known as DE/rand/1. Although the DE/rand/1 mutation have been employed in this chapter, it's important to mention that other variants of the mutation rule have been proposed in literature, see [29]:

DE/best/1: $x'_{off} = x_{best} + F(x_s - x_t)$
DE/cur-to-best/1: $x'_{off} = x_i + F(x_{best} - x_i) + F(x_s - x_t)$
DE/best/2: $x'_{off} = x_{best} + F(x_s - x_t) + F(x_u - x_v)$
DE/rand/2: $x'_{off} = x_r + F(x_s - x_t) + F(x_u - x_v)$

where x_{best} is the solution with the best performance among the individuals of the population, x_u and x_v are two additional randomly selected individuals.

Then, to increase exploration, each gene of the new individual x'_{off} is switched with the corresponding gene of x_i with a uniform probability $CR \in [0, 1]$ and the final offspring x_{off} is generated:

$$x_{off,j} = \begin{cases} x_{i,j} & \text{if} \quad rand(0,1) < CR \\ x'_{off,j} & otherwise \end{cases} \tag{12.2}$$

where $rand(0,1)$ is a random number between 0 and 1; j is the index of the gene under examination, from 1 to n, n the length of each individual.

```
generate S_pop individuals of the initial population randomly;
while budget condition
  for i = 1 : S_pop
    compute f(x_i);
  end-for
  for i = 1 : S_pop
    **mutation**
    select three individuals x_r, x_s, and x_t;
    compute x'_off = x_t + F(x_r - x_s);
    **crossover**
    x_off = x'_off;
    for j = 1 : n
      generate rand(0,1);
      if rand(0,1) < CR
        x_off,j = x_i,j;
      end-if
    end-for
    **selection**
    if f(x_off) ≤ f(x_i)
      x_i = x_off;
    end-if
  end-for
end-while
```

Fig. 12.1 DE pseudocode

The resulting offspring x_{off} is evaluated and, according to a steady-state strategy, it replaces x_i if and only if $f(x_{off}) \leq f(x_i)$; otherwise no replacement occurs. For the sake of clarity, the pseudo-code highlighting working principles of the DE is shown in Fig. 12.1.

12.2.2 Self-Adaptive Control Parameter Update

The standard DE described in Subsection 12.2.1 has been modified according to the work proposed in [2] and its enhancement proposed in [41]. When the initial population is generated, two extra values between 0 and 1 are also generated per each individual. These values represent F and CR related to the individual under analysis. Each individual is thus composed (in a self-adaptive logic) of its genotype and its control parameters:

$$x_i = \langle x_{i,1}, x_{i,2}, \ldots, x_{i,j}, \ldots x_{i,n}, F_i, CR_i \rangle . \tag{12.3}$$

In accordance with a self-adaptive logic, see e.g. [32], the variation operations are preceded by the parameter update. More specifically when, at each generation, the i^{th} individual x_i is taken into account and three other individuals are extracted randomly, its parameters F_i and CR_i are updated according to the following scheme:

$$F_i = \begin{cases} F_l + F_u rand_1, \text{ if } rand_2 < \tau_1 \\ F_i, \qquad \text{otherwise} \end{cases} \tag{12.4}$$

$$F_i = -F_i \text{ if } rand_5 < \tau_3 \text{ AND } f(x_s) > f(x_t) \tag{12.5}$$

$$CR_i = \begin{cases} rand_3, \text{ if } rand_4 < \tau_2 \\ CR_i, \qquad \text{otherwise} \end{cases} \tag{12.6}$$

where $rand_j$, $j \in \{1,2,3,4,5,6\}$, are uniform random values between 0 and 1; τ_1, τ_2, and τ_3 are constant values which represent the probabilities that parameters are updated, F_l and F_u are constant values which represent the minimum value that F_i could take and the maximum variable contribution to F_i, respectively. The sign inversion in the scale factor described in eq. (12.5) can be seen as the exploitation of a crude approximation of the gradient information in order to generate offspring along the most promising search directions. The newly calculated values of F_i and CR_i are then used for generating the offspring. Mutation, crossover, and selection scheme are performed as shown in Subsection 12.2.1 for a standard DE.

12.2.3 Population Size Reduction

Taking into account the studies given in [1] and [3] a population size reduction has been integrated within the proposed algorithm. This algorithmic component requires that initial population size S^1_{pop}, total budget T_b, in terms of fitness evaluations, and number of stages N_s, i.e. the number of population sizes, employed during the algorithm's run are prearranged.

Thus, the total budget of the algorithm is divided into N_s periods, each period being characterized by a population size value S^k_{pop} (for $k = 1$ we obtain the initial population size). Each period is composed of N^k_g generations which are calculated in the following way:

$$N^k_g = \left\lfloor \frac{T_b}{N_s S^k_{pop}} \right\rfloor + r_k \tag{12.7}$$

where r_k is a constant non-negative value which takes a positive value when T_b is not divisible by N_s. In this case r_k extra generations are performed. The population reduction is simply carried out by halving the population size at the beginning of the new stage, see [1]. In other words, for $k = 1,2,...,N_s - 1$, $S^{k+1}_{pop} = \frac{S^k_{pop}}{2}$.

In this way, the population size is progressively reduced during the optimization process until the final budget is reached. The concept behind this strategy can be explained as the satisfaction of the necessity of focusing the search in progressively smaller search spaces in order to inhibit the DE stagnation in the environment with high dimensionality. During the early stages of the optimization process, the search requires a highly explorative search rule, i.e. a large population size, in order to explore a large portion of the decision space. During the optimization, the search space is progressively narrowed by decreasing the population size and thus exploiting the promising search directions previously detected. Although the number of stages and the population size values remain arbitrary issues defined by the algorithmic designer, the idea seems to lead to a fairly robust algorithmic behavior for the setting proposed in [41] and seems to be very promising for LSPs, as highlighted in [1].

The last topic to be clarified is the selection rule employed every time a population reduction occurs. At the end of each stage, i.e. at each N^k_g generation for $k = 2,3,...,N_s$, the population is divided into two sub-populations on the basis of the position index i of the individuals. Each sub-population encounters $\frac{S^k_{pop}}{2}$ individuals. Thus, the first sub-population is composed of the candidate solutions $x_1, x_2, ..., x_{\frac{S_{popk}}{2}}$ while the second sub-population is composed of the candidate solutions $x_{\frac{S_{popk}}{2}+1}, x_{\frac{S_{popk}}{2}+2}, ..., x_{S^k_{pop}}$. The selection occurs by applying the one-to-one spawning, typical of the DE logic, to the two sub-populations analogous to the selection among parent and offspring individuals in a standard DE scheme. In other words, the individuals x_i and $x_{\frac{S_{popk}}{2}+i}$ are pairwise compared, for $i = 1,2,..\frac{S^k_{pop}}{2}$, and the individuals having the most promising fitness value are retained for the subsequent generation.

For the sake of clarity, it should be remarked that in order to guarantee a proper functioning of the population reduction mechanism, populations should never undergo sorting of any kind.

12.2.4 Scale Factor Local Search

At each generation, with a certain probability p_{ls}, the individual with the best performance undergoes local search while its offspring is generated. However, the local

search is not applied to all the coordinates of the individual but on its scale factor F_i. The main idea is that the update of the scale factor and thus generation of the offspring is, with a certain probability, controlled in order to guarantee a high quality solution which can take on a key role in subsequent generations, see also [18].

Local search in the scale factor space can be seen as the minimization over the variable F_i of fitness function f in the direction given by x_r and x_s and modified by the crossover. More specifically, at first the scale factor local search determines those genes which are undergoing crossover by means of the standard criterion explained in eq. (12.2), then it attempts to find the scale factor value which guarantees an offspring with the best performance. Thus, for given values of x_t, x_r, x_s, and the set of design variables to be swapped during the crossover operation, the scale factor local search attempts to solve the following minimization problem:

$$\min_{F_i} f(F_i) \quad \text{in} \quad [-1,1]. \tag{12.8}$$

For sake of clarity, the procedure describing the fitness function is shown in Fig. 12.2.

insert F_i;
compute $x'_{off} = x_t + F_i(x_r - x_s)$;
perform the swapping of the genes
and generate in a crossover fashion x_{off};
compute $f(F_i) = f(x_{off})$;
return $f(F_i)$;

Fig. 12.2 Local search fitness function, $f(F_i)$ pseudocode

It must be remarked that each scale factor F_i corresponds to an offspring solution x_{off} during the local search and thus the proposed local search can be seen as an instrument for detecting solutions with a high quality over the directions suggested by a DE scheme. At the end of the local search process, newly generated design variables $x_{i,j}$ with corresponding scale factor F_i within the candidate solution x_i, see eq. (12.3), compose the offspring solution. In addition, it is fundamental to observe that negative values of F_i are admitted up to -1. The meaning of the negative scale factor is obviously, in this context, the inversion of the search direction. If this search in the negative direction succeeds, the corresponding positive value (the absolute value $|F_i|$) is assigned to the offspring solution which has been generated by a local search. In order to perform this minimization, a simple uni-dimensional hill-climb (see any book on optimization, e.g. [33]) local search has been employed.

The algorithm uses the current value of F_i as a starting point and is composed of an exploratory move and a decisional move. The exploratory move samples $F_i - h$ and $F_i + h$ where h is a step size. The decisional move computes the $\min\{f(F_i - h), f(F_i), f(F_i + h)\}$ and selects the corresponding point as the center of the next exploratory move. If the center of the new exploratory move is still F_i,

the step size h is halved. The local search is stopped when a budget condition is exceeded. For the sake of completeness the pseudo-code of the Scale Factor Hill-Climb (SFHC) is shown in Fig. 12.3.

```
insert F_i;
initialize h;
while budget condition
   compute f(F_i − h), f(F_i), and f(F_i + h);
   select the point with the best performance F_i^*;
   if F_i^* == F_i
      h = h/2;
   end-if
   F_i = F_i^*;
end-while
```

Fig. 12.3 SFHC pseudocode

It should be remarked that the SFHC is a local search algorithm characterized by a steepest descent pivot rule, see [12], i.e. an algorithm which explores the whole neighborhood of the candidate solution before making a decision on the search direction. This property makes, in general, the local search accurate and thus relatively computationally expensive. The computational cost of the search in one dimension cannot, in any case, be very high.

It is interesting to visualize the functioning of this local searcher in terms of generation of an offspring within a DE for a multi-dimensional problem. Given that the scale factor is related to the modulus of a moving vector $(x_r - x_s)$ in the generation of the preliminary offspring, the SFHC in operation can be seen as a pulsing vector in a multi-dimensional space which tunes to the best offspring and then generates this offspring.

Regarding the probability for the local search activation, the initial value of p_{ls} is set equal to 1. Subsequently, p_{ls} is progressively halved every time the population size is reduced. The main idea is that at the beginning of the optimization process the global search due to large population size is balanced by the application of the SFHC on the individual with the best performance and therefore, offers promising search directions to the algorithm. During the subsequent stages of the optimization process, the DELS tends, due to the population reduction, to focus its search in a smaller portion of the decision space. In these conditions a lower intensity of the local search is needed. It should be remarked that the reduction in the probability p_{ls} means, on one hand, a reduction in the local search with respect to the occurrence within the generations and, on the other hand, a constant employment of global and local components in terms of fitness evaluations. In other words, the reduction rule related to p_{ls} assures that the ratio between the amount of fitness evaluations devoted to the local search with respect to the ones devoted global search is kept constant throughout the entire optimization process.

Table 12.1 Test Problems

Test Problem	Analytic Expression	Decision Space
Ackley	$-20+e+20\exp\left(-\frac{0.2}{n}\sqrt{\sum_{i=1}^{n}x_i^2}\right)$ $-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi\cdot x_i)x_i\right)$	$[-1,1]^n$
Ellipsoid	$\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	$[-65.536,65.536]^n$
Griewangk	$\frac{\|\|x\|\|^2}{4000}-\prod_{i=0}^{n}\cos\frac{x_i}{\sqrt{i}}+1$	$[-600,600]^n$
Michalewicz	$-\sum_{i=1}^{n}\sin x_i\left(\sin\left(\frac{i\cdot x_i^2}{\pi}\right)\right)^{20}$	$[0,\pi]^n$
Parallel Axis	$\sum_{i=1}^{n}i\cdot x_i^2$	$[-5.12,5.12]^n$
Rastrigin	$10n+\sum_{i=0}^{n}\left(x_i^2-10\cos(2\pi x_i)\right)$	$[-5.12,5.12]^n$
Rosenbrock	$\sum_{i=1}^{n-1}\left(\left(x_{n+1}-x_i^2\right)^2+(1-x)^2\right)$	$[-2.048,2.048]^n$
Schwefel	$\sum_{i=1}^{n}x_i\sin\left(\sqrt{\|x_i\|}\right)$	$[-500,500]^n$
Tirronen	$3\exp\left(-\frac{\|\|x\|\|^2}{10n}\right)-10\exp\left(-8\|\|x\|\|^2\right)$ $+\frac{2.5}{n}\sum_{i=1}^{n}\cos\left(5x_i(1+i \mod 2)\cos\left(\|\|x\|\|\right)\right)$	$[-10,5]^n$

For the sake of clarity, the pseudocode highlighting the working principle of the DELS integrating the scale factor local search is given in Fig. 12.4.

12.3 Numerical Results

The DELS has been tested on a set of benchmark problems. The problems considered in this study are listed in Table 12.1. The amount of variables is indicated with n.

In addition to the problems listed in Table 12.1, one rotated problem for each test problem has been added to the benchmark set. The rotated problems are obtained by means of multiplication of the vector of variables to a randomly generated orthogonal rotation matrix.

The performance of the DELS has been compared with the performance obtained by the self-adaptive Differential Evolution with dynamic population size and inversion of the scale factor (jDEdynNP-F) proposed in [1] for LSPs, the Differential Evolution with Self-adaptive Control Parameters proposed in [2] and with a standard DE. The algorithms involved in this study ad their parameter settings are listed below.

1. The DE has been run with $F = 0.7$, and $CR = 0.7$ in accordance to the suggestions given in [42].
2. Regarding the SACPDE, the constant values in formulas (12.4) and (12.6) have been set, respectively, $F_l = 0.1$, $F_u = 0.9$, $\tau_1 = 0.1$, and $\tau_2 = 0.1$ as suggested in [2].
3. Regarding the jDEdynNP-F, it employs the scale factor inversion shown in formula (12.5) with $\tau_3 = 0.75$, as suggested in [1] and the population size reduction described in Subsection 12.2.3 with $N_s = 4$ in accordance with the results in [41]. The control parameters vary with the same rules given in formulas 12.4 and 12.6 and the related parameter setting (F_l, F_u, τ_1 and τ_2) has been performed as for the SACPDE.

generate S_{pop} individuals of the initial population with related parameters randomly;
while budget condition
  for $i = 1 : S_{pop}$
    compute $f(x_i)$;
  end-for
  for $i = 1 : S_{pop}$
  generate $rand_j$ for $j = 1$ to 5;
  randomly select three individuals x_r, x_s, and x_t;
  if x_i is the best AND $rand_6 < p_{ls}$
    Scale Factor Hill-Climb
    randomly select genes undergoing crossover;
    apply SFHC using F_i as the starting point;
    save the resulting offspring x_{off};
  else
    F_i update

$$F_i = \begin{cases} F_l + F_u rand_1, \text{ if } rand_2 < \tau_1 \\ F_i, \qquad \text{otherwise} \end{cases};$$

    if $rand_5 < \tau_3$ AND $f(x_s) > f(x_t)$
      $F_i = -F_i$;
    end-if
    CR_i update
    generate $rand_3$ and $rand_4$;

$$CR_i = \begin{cases} rand_3, \text{ if } rand_4 < \tau_2 \\ CR_i, \qquad \text{otherwise} \end{cases}$$

    mutation
    compute $x'_{off} = x_t + F_i(x_r - x_s)$;
    crossover
    $x_{off} = x'_{off}$;
    for $j = 1 : n$
      generate $rand(0,1)$;
      if $rand(0,1) < CR_i$
        $x_{off,j} = x_{i,j}$;
      end-if
    end-for
  end-if
  selection
  if $f(x_{off}) \leq f(x_i)$
    $x_i = x_{off}$;
  end-if
  end-for
  population size and local search probability reduction
  if $N_g == \left\lfloor \frac{T_b}{N_s S_{pop}} \right\rfloor + r_k$
    for $i = 1 : \frac{S_{pop}}{2}$
      if $f\left(x_{\frac{S_{pop}}{2}+i}\right) \leq f(x_i)$
        $x_i = x_{\frac{S_{pop}}{2}+i}$;
      end-if
    end-for
    halve S_{pop} and p_{ls};
    update N_g;
  end-if
end-while

Fig. 12.4 DELS pseudocode

4. The proposed DELS has the same parameter setting of the jDEdynNP-F. In addition, the SFHC has been run with a budget of 40 fitness evaluations and an initial step size $h = 0.1$.

The experiments have been performed for $n = 100$, $n = 500$, and $n = 1000$. The total budget for all the algorithms has been set equal to 1.5×10^5, 3×10^6, and 6×10^6 fitness evaluations, respectively. Regarding jDEdynNP-F and DELS, the initial population size S^1_{pop} is set equal to n. Regarding DE and SACPDE the population size S_{pop} has been set in order to keep constant the amount of generations for all the algorithms considered in this study. More specifically, DE and SACPDE have been run with a population size $S_{pop} = 27$ for the 100 dimension case, $S_{pop} = 134$ for the 500 dimension case, and $S_{pop} = 267$ for the 1000 dimension case.

For each test problem, each algorithm performed the optimization process on 30 independent runs.

Regarding the rotated test problems, in order to perform a fair comparison and an analysis on the robustness, a rotation matrix has been generated for each problem and for each run. Then, all the algorithms considered in this study have been run with the same set of rotated problems.

12.3.1 Results in 100 Dimensions

Table 12.2 shows the final average detected results by each algorithm ± the standard deviations for the 100 dimension case.

Table 12.2 Average final fitness values ± standard deviations in 100 dimensions

Test Problem	DE	SACPDE	jDEdynNP-F	DELS
Ackley	9.41E+00±1.26E+00	4.54E+00±1.14E+00	8.87E-05±8.97E-05	**3.21E-05±4.56E-05**
Rotated Ackley	7.87E+00±1.67E+00	5.83E+00±1.50E+00	6.49E-01±6.04E-01	**3.57E-01±5.77E-01**
Ellipsoid	**1.02E+03±4.20E+02**	3.81E+03±1.87E+03	4.37E+03±9.65E+02	3.91E+03±9.11E+02
Rotated Ellipsoid	**8.86E+02±3.90E+02**	3.80E+03±1.32E+03	4.07E+03±7.58E+02	3.98E+03±9.97E+02
Griewangk	1.02E-01±1.74E-01	2.76E-01±6.66E-01	**1.21E-02±2.12E-02**	2.18E-02±4.05E-02
Rotated Griewangk	2.24E-02±3.31E-02	**7.45E-4±2.34E-03**	6.85E-03±8.11E-03	2.45E-03±2.42E-03
Michalewicz	-4.60E+01±5.86E+00	-8.99E+01±1.13E+00	-8.86E+01±1.15E+00	**-9.16E+01±2.49E+00**
Rotated Michalewicz	-8.16E+00±6.48E-01	-7.98E+00±4.31E-01	-8.28E+00±9.11E-01	**-1.09E+01±1.37E+00**
Parallel	2.74E-04±6.36E-04	**4.39E-18±6.96E-18**	3.76E-08±4.92E-08	2.13E-08±3.05E-08
Rotated Parallel	5.13E+00±3.82E+00	**1.50E+00±8.23E-01**	6.44E+00±3.98E+00	3.40E+00±2.10E+00
Rastrigin	2.28E+02±8.02E+01	3.74E+01±8.29E+00	1.28E+01±5.39E+00	**1.27E+01±4.92E+00**
Rotated Rastrigin	**1.64E+02±5.15E+01**	1.89E+02±4.76E+01	3.05E+02±7.78E+01	1.96E+02±6.22E+01
Rosenbrock	2.24E+02±5.59E+01	1.82E+02±5.26E+01	**1.15E+02±2.72E+01**	1.45E+02±4.55E+01
Rotated Rosenbrock	1.34E+02±5.59E+01	1.25E+02±5.52E+01	9.71E+01±1.30E+00	**9.69E+01±1.43E+00**
Schwefel	1.60E+04±2.19E+03	4.46E+03±1.27E+03	1.10E+03±3.03E+02	**9.76E+02±4.30E+02**
Rotated Schwefel	1.99E+04±3.78E+03	1.64E+04±1.03E+03	1.74E+04±1.21E+03	**1.59E+04±2.00E+03**
Tirronen	-1.75E+00±1.17E-01	-2.48E+00±1.32E-02	-2.47E+00±7.98E-03	**-2.49E+00±6.16E-03**
Rotated Tirronen	-7.82E-01±6.42E-02	-1.03E+00±1.00E-01	-1.01E+00±1.57E-01	**-1.61E+00±2.08E-01**

Results in Table 12.2 show that the proposed DELS obtained the best results for 10 problems out of the 18 considered in the 100 dimension benchmark. Thus, the DELS seems clearly to be the most efficient algorithm in terms of final solutions.

In the remaining 8 test problems the DELS, in any case is never by far outperformed by other algorithms and still demonstrates a competitive performance. For example, with the Griewangk function, although the DELS does not seem to have a very promising behavior, reaches satisfactory results anyway.

In order to prove the statistical significance of the results, the Student's t-test has been applied according to the description given in [34] for a confidence level of 0.95. Final values obtained by the DELS have been compared to the final value returned by each algorithm used as a benchmark. Table 12.3 shows the results of the test. Indicated with "+" is the case when the DELS statistically outperforms, for the corresponding test problem, the algorithm mentioned in the column; indicated with "=" is the case when pairwise comparison leads to success of the t-test i.e. the two algorithms have the same performance; indicated with "-" is the case when the DELS is outperformed.

Table 12.3 Results of the Student's t-test in 100 dimensions

Test Problem	DE	SACPDE	jDEdynNP-F
Ackley	+	+	=
Rotated Ackley	+	+	=
Ellipsoid	−	=	=
Rotated Ellipsoid	−	=	=
Griewangk	=	=	=
Rotated Griewangk	=	=	=
Michalewicz	+	=	+
Rotated Michalewicz	+	+	+
Parallel	=	−	=
Rotated Parallel	=	−	+
Rastrigin	+	+	=
Rotated Rastrigin	=	=	+
Rosenbrock	+	=	=
Rotated Rosenbrock	=	=	=
Schwefel	+	+	=
Rotated Schwefel	+	=	=
Tirronen	+	+	+
Rotated Tirronen	+	+	+

The t-test results listed in Table 12.3 show that the DELS loses the comparison in only 4 cases out of the 54 comparisons carried out i.e. the DELS loses in only 7.4% of the pairwise comparisons. In addition, it should be remarked that the scale factor local search never reduces the performance of the jDEdynNP-F framework, as the right hand column of Table 12.3 proves.

In addition to the t-test also the Friedman test has been performed, see [34]. In a nutshell, Friedman test is a non-parametric test equivalent of the repeated-measures

ANOVA. Under the null-hypothesis, it states that all the algorithms are equivalent. If the hypothesis is rejected, the algorithms have a different performance. Details of the test can be found in [34] and the application of this test in the context of algorithm comparisons is described in [11]. In this study, rotated and non-rotated problems have been treated separately and in both cases level of significance has been set to 0.05. We can conclude that the probability that the algorithms under analysis have the same performance for non-rotated problems is 0 while the probability that this events happens for rotated problems is 1.4618×10^{-8}, i.e. this event is very unlike.

In order to carry out a numerical comparison of the convergence speed performance, for each test problem, the average final fitness value returned by the best performing algorithm G has been considered. Subsequently, the average fitness value at the beginning of the optimization process J has also been computed. The threshold value $THR = J - 0.95(G - J)$ has then been calculated. The value THR represents 95% of the decay in the fitness value of the algorithm with the best performance. If an algorithm succeeds during a certain run to reach the value THR, the run is said to be successful. For each test problem, the average amount of fitness evaluations $\bar{ne}$ required, for each algorithm, to reach THR has been computed. Subsequently, the Q-test (Q stands for Quality) described in [10] has been applied. For each test problem and each algorithm, the Q measure is computed as:

$$Q = \frac{\bar{ne}}{R} \tag{12.9}$$

where the robustness R is the percentage of successful runs. It is clear that, for each test problem, the smallest value equals the best performance in terms of convergence speed.The value "Inf" means that $R = 0$, i.e. the algorithm never reached the THR.

Table 12.4 shows the Q values in 100 dimensions. The best results are highlighted in bold face.

Results in Table 12.4 show that the best performance values in terms of Q-measure are distributed among the considered algorithms. In other words, there is not a clear best algorithm in terms of Q-measure. The jDEdynNP-F seems to have a slightly lower performance than the other algorithms. It is important to notice that the DELS has a very robust behaviour compared to the other algorithms considered in this study. As a matter of fact, as shown in Table 12.4, the DELS is the only algorithm whose Q-measure never takes the "Inf" value. This means that the DELS is always able to detect candidate solutions with a high performance and is never outperformed consistently by other algorithms. We can conclude that in 100 dimension case the proposed DELS tends either to have an excellent performance with respect to the other algorithms (e.g. Rotated Schwefel) or is anyway competitive with the other algorithms (e.g. Rastrigin).

In order to graphically show the behaviour of the algorithms, some examples of the average performance are plotted against the number of fitness evaluations (for some of the test problems listed in Table 12.1) and represented in Figure 12.5.

Table 12.4 Results of the Q-test in 100 dimensions

Test Problem	DE	SACPDE	jDEdynNP-F	DELS
Ackley	Inf	Inf	**3.27E+02**	4.61E+02
Rotated Ackley	Inf	Inf	6.83E+02	**6.51E+02**
Ellipsoid	**9.94E+01**	1.43E+02	3.90E+02	5.40E+02
Rotated Ellipsoid	**1.16E+02**	1.79E+02	4.09E+02	5.92E+02
Griewangk	7.35E+01	**3.96E+01**	1.39E+02	1.99E+02
Rotated Griewangk	6.58E+01	**4.04E+01**	1.39E+02	1.92E+02
Michalewicz	Inf	3.19E+03	Inf	**1.80E+03**
Rotated Michalewicz	Inf	Inf	Inf	**1.45E+04**
Parallel	6.67E+01	**3.63E+01**	1.39E+02	1.83E+02
Rotated Parallel	6.39E+01	**3.94E+01**	1.72E+02	2.22E+02
Rastrigin	Inf	**3.40E+02**	6.61E+02	8.63E+02
Rotated Rastrigin	**5.09E+02**	2.06E+03	9.94E+03	1.74E+03
Rosenbrock	**1.96E+01**	2.07E+01	1.06E+02	1.34E+02
Rotated Rosenbrock	**1.45E+01**	2.12E+01	1.04E+02	1.35E+02
Schwefel	Inf	Inf	7.38E+02	9.57E+02
Rotated Schwefel	Inf	Inf	Inf	**4.53E+03**
Tirronen	Inf	**3.88E+02**	7.13E+02	7.84E+02
Rotated Tirronen	Inf	Inf	Inf	**1.49E+04**

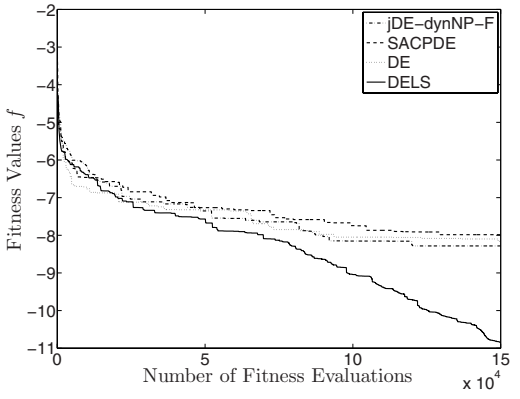
12.3.2 Results in 500 Dimensions

The experiments performed in 100 dimensions have been repeated for the test problem listed in Table 12.1 for 500 dimensions. Numerical results in terms of final solutions, t-test, and Q-measure are shown in Table 12.5, 12.6, and 12.7, respectively.

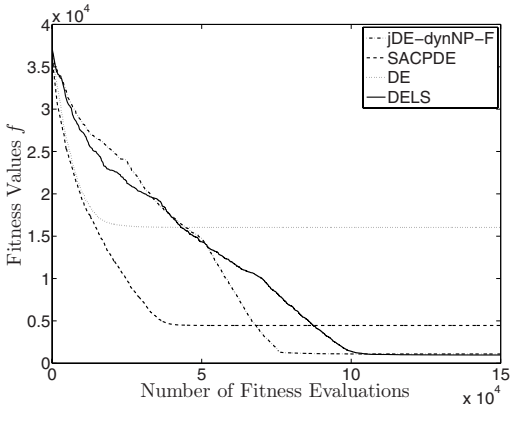
The Friedman test in 500 variables shows that the probability that the algorithms have the same performance for non-rotated problems is 3.3307×10^{-16} and for rotated problems 7.0201×10^{-7}. Also in this case the probability is lower then considered level of significance $\alpha = 0.05$, therefore we can conclude that we are almost sure that the algorithms have a different performance.

Numerical results in the 500 dimension case (see Table 12.5) show that the proposed DELS reaches the best results for 50% of the best functions. The results related to the t-test, listed in Table 12.6, show that the DELS loses only 3 pairwise comparisons out of the 54 carried out and wins 26 comparisons. In other words, regarding the final solution, the DELS significantly outperforms the other algorithms in 48.1% of the comparisons, is outperformed in 5.5% of the comparisons and has the same performance for 44.4% of the comparisons. Concerning the convergence speed performance, Table 12.7 shows that it is impossible to identify clearly the overall best algorithm in terms of Q-measure values. Nevertheless, the SACPDE seems to have a convergence speed performance slightly superior with respect to the other algorithms. On the other hand, the DELS clearly has the best performance in terms of algorithmic robustness since it is the only algorithm which always succeeds at detecting a competitive value (there are no "Inf" values in the DELS column).

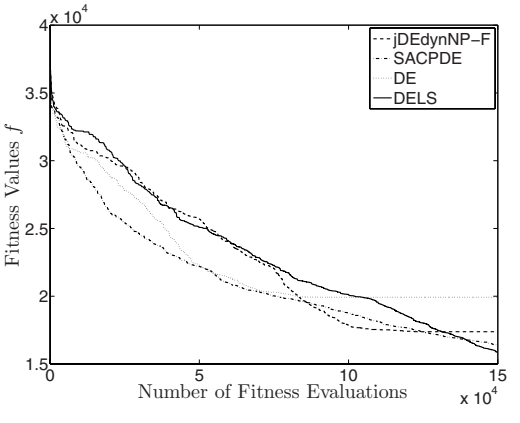
Some examples of the average performance are shown in Figures 12.6.



(a) Rotated Michalewicz



(b) Schwefel



(c) Rotated Schwefel

Fig. 12.5 Performance trends in 100 dimensions

Table 12.5 Average final fitness values ± standard deviations in 500 dimensions

Test Problem	DE	SACPDE	jDEdynNP-F	DELS
Ackley	5.02E+00±5.56E-01	2.09E+00±3.75E-01	1.11E-12±8.86E-13	**6.01E-13**±5.64E-14
Rotated Ackley	3.97E+00±3.58E-01	2.88E+00±2.44E-01	7.85E-13±1.08E-13	**7.13E-13**±5.08E-14
Ellipsoid	6.14E+04±5.47E+03	1.12E+04±1.35E+03	**8.06E+03**±5.77E+02	8.86E+03±1.10E+03
Rotated Ellipsoid	6.18E+04±1.66E+04	1.04E+04±1.03E+03	**7.42E+03**±1.09E+03	9.50E+03±1.07E+03
Griewangk	3.58E-02±4.08E-02	1.72E-02±3.44E-02	**4.97E-15**±2.03E-15	5.52E-15±1.78E-15
Rotated Griewangk	2.31E-02±2.01E-02	**1.84E-10**±7.26E-11	2.57E-09±2.60E-09	1.40E-08±1.33E-08
Michalewicz	-2.42E+02±8.51E+00	-4.57E+02±2.66E+00	-4.46E+02±2.65E+00	**-4.69E+02**±8.75E-01
Rotated Michalewicz	-1.24E+01±8.41E-01	-1.21E+01±5.86E-01	-1.21E+01±4.41E-01	**-1.44E+01**±2.74E+00
Parallel	3.18E-03±1.53E-03	**4.67E-52**±7.24E-53	3.95E-42±5.64E-42	2.87E-38±2.79E-38
Rotated Parallel	5.58E+01±2.11E+01	4.22E+00±3.69E+00	3.94E+00±1.59E+00	**2.95E+00**±1.14E+00
Rastrigin	4.50E+02±6.74E+01	**9.95E-01**±1.41E+00	1.27E+00±2.54E+00	5.60E+00±4.63E+00
Rotated Rastrigin	**3.67E+02**±3.56E+01	7.59E+02±3.26E+01	7.14E+02±5.32E+01	9.21E+02±1.76E+02
Rosenbrock	6.14E+02±5.40E+01	6.16E+02±8.47E+01	**4.91E+02**±3.25E+01	4.97E+02±3.17E+01
Rotated Rosenbrock	5.08E+02±2.82E+01	5.17E+02±2.96E+01	**4.91E+02**±3.57E-01	4.92E+02±5.15E-01
Schwefel	9.23E+04±3.21E+03	3.26E+02±1.78E+02	8.88E+01±5.92E+01	**6.36E-03**±2.84E-07
Rotated Schwefel	1.79E+05±1.55E+03	9.60E+04±5.26E+02	9.14E+04±1.52E+03	**8.77E+04**±4.66E+03
Tirronen	-1.78E+00±4.99E-02	-2.49E+00±1.96E-03	-2.49E+00±3.91E-03	**-2.50E+00**±1.86E-03
Rotated Tirronen	-4.06E-01±2.42E-02	-8.72E-01±4.14E-02	-9.57E-01±4.99E-02	**-1.41E+00**±1.64E-01

Table 12.6 Results of the Student's t-test in 500 dimensions

Function	DE	SACPDE	jDEdynNP-F
Ackley	+	+	=
Rotated Ackley	+	+	=
Ellipsoid	+	+	=
Rotated Ellipsoid	+	=	−
Griewangk	=	=	=
Rotated Griewangk	=	=	=
Michalewicz	+	+	+
Rotated Michalewicz	=	=	=
Parallel	+	=	=
Rotated Parallel	+	=	=
Rastrigin	+	=	=
Rotated Rastrigin	−	=	−
Rosenbrock	+	+	=
Rotated Rosenbrock	=	=	=
Schwefel	+	+	+
Rotated Schwefel	+	+	=
Tirronen	+	+	+
Rotated Tirronen	+	+	+

Table 12.7 Results of the Q-test in 500 dimensions

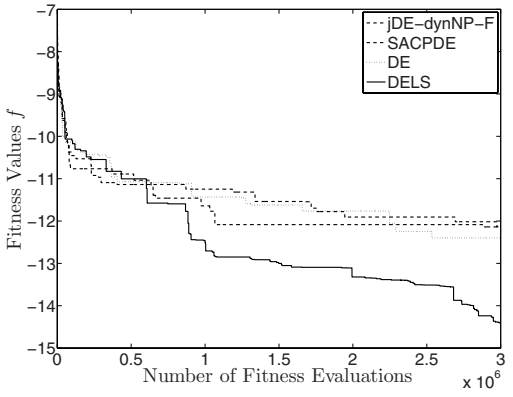
Test Problem	DE	SACPDE	jDEdynNP-F	DELS
Ackley	Inf	Inf	**5.00E+03**	5.33E+03
Rotated Ackley	Inf	Inf	**5.73E+03**	6.22E+03
Ellipsoid	8.50E+03	**1.46E+03**	6.63E+03	7.05E+03
Rotated Ellipsoid	9.46E+03	**1.46E+03**	6.59E+03	7.00E+03
Griewangk	8.92E+02	**4.01E+02**	1.56E+03	1.71E+03
Rotated Griewangk	9.64E+02	**3.73E+02**	1.64E+03	1.66E+03
Michalewicz	Inf	2.84E+04	4.86E+04	**2.29E+04**
Rotated Michalewicz	Inf	Inf	Inf	**9.55E+04**
Parallel	8.96E+02	**3.83E+02**	1.50E+03	1.65E+03
Rotated Parallel	8.98E+02	**4.01E+02**	1.76E+03	1.91E+03
Rastrigin	2.30E+04	**1.08E+04**	1.85E+04	1.67E+04
Rotated Rastrigin	**8.48E+03**	2.98E+04	2.73E+04	9.42E+04
Rosenbrock	**2.01E+02**	2.17E+02	1.11E+03	1.17E+03
Rotated Rosenbrock	**1.79E+02**	1.99E+02	1.10E+03	1.09E+03
Schwefel	Inf	**1.45E+04**	1.96E+04	2.07E+04
Rotated Schwefel	Inf	Inf	Inf	**5.21E+04**
Tirronen	Inf	**1.51E+04**	1.98E+04	1.64E+04
Rotated Tirronen	Inf	Inf	Inf	**5.38E+04**

12.3.3 Results in 1000 Dimensions

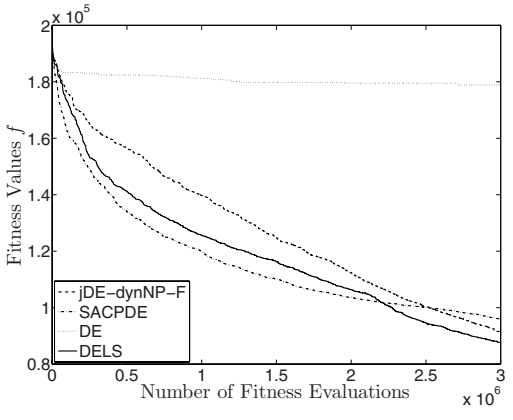
The same experiments carried out in 100 and 500 dimensions have also been performed in 1000 dimensions. Numerical results in in 1000 dimensions are shown in Table 12.8, 12.9, and 12.10. Some performance trends are given in Figures 12.7.

The Friedman test in 1000 variables shows that the probability that the algorithms have the same performance for non-rotated problems is 0 and for rotated problems 2.4636×10^{-8}. Also for 1000 variables the probability is very low and we can conclude that we are almost sure that the algorithms have a different performance.

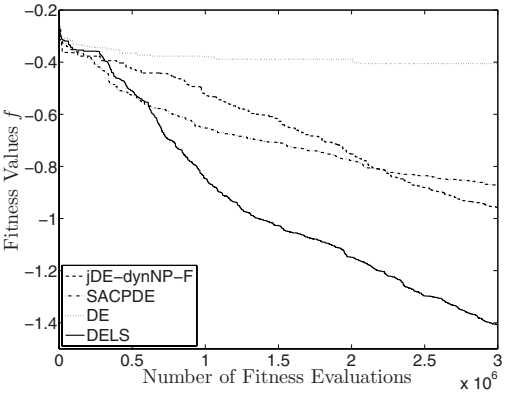
Results in Table 12.8 show that the proposed DELS has the best performance in terms of final solution since it converged to the best solutions in 50% of the test problems analyzed. Regarding the t-test shown in Table 12.9, it can be observed that the DELS significantly outperforms the other algorithms for the 55.5% of the pairwise comparisons while is outperformed for only the 3.7% of the comparisons. Most importantly, it must be remarked that, as shown in Table 12.10, also for the 1000 dimension case, the Q-test shows that the DELS is the only algorithm which does not displays "Inf" values. Thus, the DELS confirms its high performance in terms of robustness despite the high dimensionality of this set of experiments. In summary, the DELS seems to be less affected than the other algorithms from the curse of dimensionality.



(a) Rotated Michalewicz



(b) Rotated Schwefel



(c) Rotated Tirronen

Fig. 12.6 Performance trends in 500 dimensions

Table 12.8 Average final fitness values ± standard deviations in 1000 dimensions

Test Problem	DE	SACPDE	jDEdynNP-F	DELS
Ackley	4.47E+00±3.10E-01	1.47E+00±3.38E-01	5.41E-12±4.99E-12	**1.80E-12**±8.74E-14
Rotated Ackley	4.01E+00±5.22E-02	3.24E+00±2.62E-01	1.14E-11±1.82E-11	**2.67E-12**±1.11E-12
Ellipsoid	2.31E+06±6.82E+05	4.19E+04±6.78E+03	**2.94E+04**±4.37E+03	3.11E+04±3.64E+03
Rotated Ellipsoid	2.39E+06±5.79E+05	4.26E+04±2.93E+03	3.32E+04±4.55E+03	**2.86E+04**±4.39E+03
Griewangk	1.86E+00±1.24E-01	3.83E-02±7.67E-02	1.93E-14±1.05E-14	**1.58E-14**±7.79E-15
Rotated Griewangk	2.40E+00±4.95E-01	2.47E-03±4.93E-03	**3.39E-06**±1.12E-06	8.66E-06±5.27E-06
Michalewicz	-2.20E+02±6.37E+00	-4.26E+02±1.17E+01	-7.51E+02±4.31E+00	**-8.23E+02**±2.50E+01
Rotated Michalewicz	-1.49E+01±4.88E-01	-1.46E+01±1.66E-01	-1.46E+01±6.53E-01	**-1.63E+01**±3.06E+00
Parallel	2.18E+02±8.66E+01	**6.24E-36**±7.30E-36	5.75E-29±6.28E-29	2.17E-27±2.92E-27
Rotated Parallel	8.98E+02±1.71E+02	**2.31E+01**±8.83E+01	2.55E+01±1.08E+02	2.41E+01±5.65E+01
Rastrigin	5.62E+02±1.53E+01	3.05E+02±5.20E+01	**2.29E+00**±7.56E-01	8.95E+01±6.31E+01
Rotated Rastrigin	**5.82E+02**±8.38E+01	1.45E+03±2.32E+02	1.49E+03±2.51E+02	1.59E+03±2.35E+02
Rosenbrock	1.39E+03±1.88E+02	1.29E+03±1.61E+02	**9.94E+02**±3.90E+01	1.05E+03±1.02E+02
Rotated Rosenbrock	1.34E+03±9.43E+01	1.05E+03±7.29E+01	**9.87E+02**±4.67E-01	1.00E+03±2.62E+01
Schwefel	2.30E+05±1.30E+03	1.86E+04±1.86E+03	**1.48E+02**±1.14E+02	2.18E+02±2.57E+02
Rotated Schwefel	3.75E+05±1.50E+03	2.27E+05±2.78E+03	2.14E+05±5.24E+03	**2.07E+05**±1.25E+03
Tirronen	-1.47E+00±2.96E-02	-2.36E+00±8.76E-03	-2.46E+00±1.32E-03	**-2.49E+00**±2.12E-03
Rotated Tirronen	-2.87E-01±2.01E-02	-6.96E-01±4.56E-02	-7.86E-01±2.88E-02	**-1.15E+00**±6.71E-02

Table 12.9 Results of the Student's t-test in 1000 dimensions

Test Problem	DE	SACPDE	jDEdynNP-F
Ackley	+	+	=
Rotated Ackley	+	+	=
Ellipsoid	+	+	=
Rotated Ellipsoid	+	+	=
Griewangk	+	=	=
Rotated Griewangk	+	=	=
Michalewicz	+	+	+
Rotated Michalewicz	=	=	=
Parallel	+	=	=
Rotated Parallel	+	=	=
Rastrigin	+	+	−
Rotated Rastrigin	−	=	=
Rosenbrock	+	+	=
Rotated Rosenbrock	+	=	=
Schwefel	+	+	=
Rotated Schwefel	+	+	+
Tirronen	+	+	+
Rotated Tirronen	+	+	+

Table 12.10 Results of the Q-test in 1000 dimensions

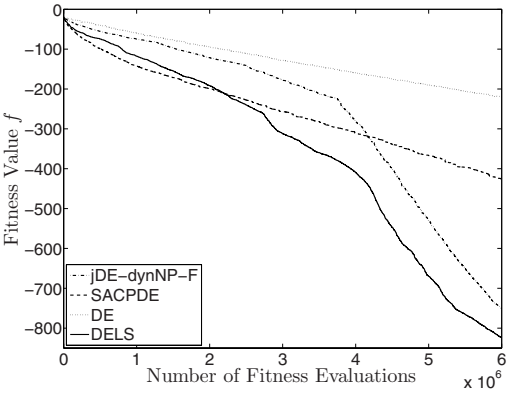
Test Problem	DE	SACPDE	jDEdynNP-F	DELS
Ackley	Inf	Inf	**1.35E+04**	1.41E+04
Rotated Ackley	Inf	Inf	**1.54E+04**	1.62E+04
Ellipsoid	2.25E+05	**4.09E+03**	1.36E+04	1.42E+04
Rotated Ellipsoid	Inf	4.13E+04	1.41E+04	**1.39E+04**
Griewangk	2.53E+03	**1.04E+03**	4.39E+03	4.54E+03
Rotated Griewangk	2.54E+03	**1.08E+03**	4.33E+03	4.64E+03
Michalewicz	Inf	Inf	Inf	**1.16E+05**
Rotated Michalewicz	Inf	Inf	Inf	**2.17E+05**
Parallel	2.33E+03	**9.83E+02**	4.00E+03	4.32E+03
Rotated Parallel	2.49E+03	**1.06E+03**	4.69E+03	1.79E+03
Rastrigin	**3.14E+04**	4.32E+04	4.71E+04	4.39E+04
Rotated Rastrigin	**3.77E+04**	1.89E+05	9.00E+04	1.22E+05
Rosenbrock	7.76E+02	**5.65E+02**	3.10E+03	3.11E+03
Rotated Rosenbrock	5.50E+02	**4.85E+02**	2.49E+03	2.64E+03
Schwefel	Inf	7.91E+04	5.12E+04	**4.98E+04**
Rotated Schwefel	Inf	Inf	7.76E+04	**5.40E+04**
Tirronen	Inf	Inf	5.10E+04	**4.65E+04**
Rotated Tirronen	Inf	Inf	Inf	**1.66E+05**

12.3.4 Discussion about the Algorithmic Components

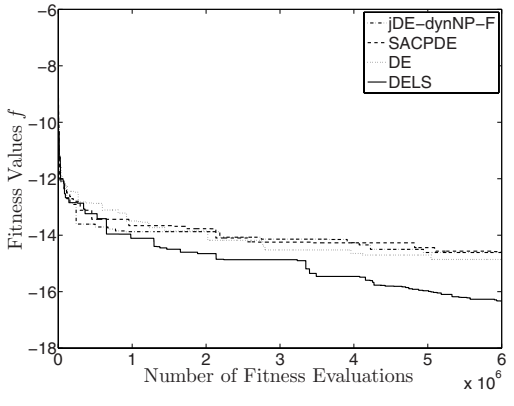
This subsection gives an explanation and an interpretation of the results on the basis of the algorithmic components employed.

As highlighted in [10], the success of the DE is due to an implicit self-adaptation contained within the algorithmic structure. More specifically, since, for each candidate solution, the search rule depends on other solutions belonging to the population (e.g. x_t, x_r, and x_s), the capability of detecting new promising offspring solutions depends on the current distribution of the solutions within the decision space. During the early stages of the optimization process, the solutions tend to be spread out within the decision space. For a given scale factor value, this implies that the mutation appears to generate new solutions by exploring the space by means of a large step size (if x_r and x_s are distant solutions, $F(x_r - x_s)$ is a vector characterized by a large modulus). During the optimization process, the solutions of the population tend to concentrate in specific parts of the decision space. Therefore, the step size in the mutation is progressively reduced and the search is performed in the neighborhood of the solutions. In other words, due to its structure, a DE scheme is highly explorative at the beginning of the evolution and subsequently becomes more exploitative during the optimization.

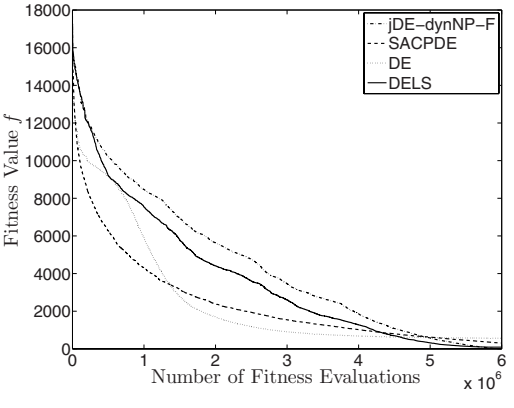
Although this mechanism seems, at the first glance, very efficient, it hides a limitation. If for some reasons, the algorithm does not succeed at generating offspring solutions which outperform the corresponding parent, the search is repeated again with the similar step size values and likely fails by falling into the undesired stagnation condition (see [16]). In other words, the main drawback of the DE is that



(a) Michalewicz

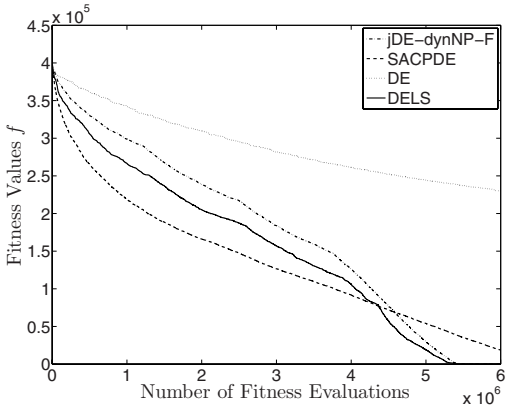


(b) Rotated Michalewicz

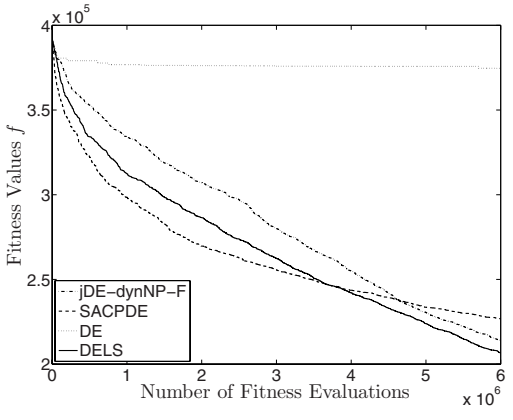


(c) Rastrigin

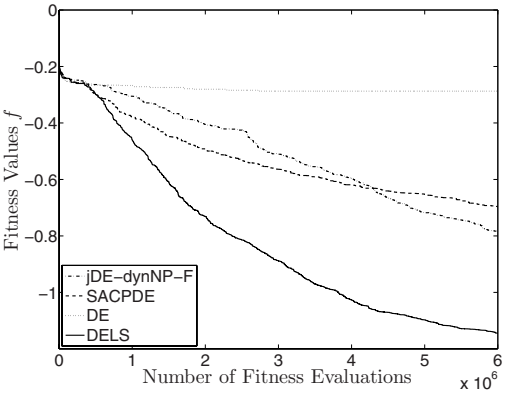
Fig. 12.7 Performance trends in 1000 dimensions



(a) Schwefel



(b) Rotated Schwefel



(c) Rotated Tirronen

Fig. 12.7 (*continued*)

the scheme has, for each stage of the optimization process, a limited amount of exploratory moves and if these moves are not enough for generating new promising solutions, the search can be heavily compromised. Clearly, the risk of the DE stagnation is higher for larger decision spaces and worsens as number of the dimensions of the problem increases. A large decision space (in terms of dimensions) requires a wide range of possible moves to enhance the capability of detecting new promising solutions.

Experimental observations from Fig 5(c), 6(b), and 7(b) show that for a complex fitness landscape (Rotated Schwefel) the DE is heavily influenced by curse of dimensionality. It can be observed that for 100 dimensions the DE performance is competitive compared to the other algorithms, for 500 variables the performance is poor and for 1000 variables stagnates early and detects a completely unsatisfactory solutions.

In order to enhance the performance of the DE by widening the range of its search moves, in [7], [8], and [9], a randomization of the scale factor is proposed. Although this operation seems to be beneficial for the DE in some specific cases (noisy problems), according to our opinion, it leads to excessive random search within the decision space possibly leading to a significant slowing down of the optimization in high dimensional problems. Conversely, the probabilistic update of the scale factor proposed in [2] seems to be an effective alternative to handle complex and multivariate functions. As a matter of fact the SACPDE tends to outperform, on a regular basis, the standard DE for most of the test problems analyzed in this chapter.

The inversion of the scale factor described in equation (12.5) and proposed in [5] can be seen as a single step local search which detects the most promising search directions on the basis of an estimation of the gradient. Thus, for high dimensional problems, the limited amount of moves of the DE is increased by means of a randomized update of the scale factor and on a knowledge based correction of this parameter during the algorithmic search. The scale factor local search, originally proposed in [21] and here proposed for LSPs is a further step in this direction. As mentioned above, the scale factor local search is independent of the amount of variables and is thus suitable for highly multivariate problems. In addition, the SFHC integrated into the framework has the crucial role of offering an alternative move to the DE which is the selection of the most suitable scale factor for a specific offspring generation. This move should lead towards the generation of promising offspring, significantly contributing to the search of more promising solutions during the subsequent generations. This effect can be easily visualized in Fig. 6(c) where the improvements appear to be not only due to the application of the local search but also (and mainly) due to the presence of the individuals generated during the local search while the global search is performed.

Finally, the population size reduction proposed in [1] plays a different, but nevertheless important role. Although this component does not explicitly offers alternative search moves, it progressively narrows the space where the search is performed by eliminating the individuals characterized by a poor performance. This makes the algorithm more exploitative and thus reduces the risk of stagnation. In other words, this component does not help to detect the global optimum in a LSP but is

fundamental in order to quickly improve upon the obtained results after completing the exploratory procedure. To give an analogy, the progressive reduction in the population size is similar to progressive increase in selection pressure in Genetic Algorithms. Following a different analogy, this mechanism is similar to a cascade algorithm composed of as many algorithms as the amount of stages N_s, see equation (12.7). The search by each algorithm is progressively focused in smaller decision space after that promising search directions are detected.

The combination of these algorithmic components appears to be very beneficial for LSPs and helpful in improving the performance of a standard DE.

12.4 Conclusion

This chapter proposes a novel Computational Intelligence algorithm for real-valued parameter optimization in high dimensions. The proposed algorithm employs a local search on the scale factor of a DE framework in order to control generation of high performance offspring solutions. The DE framework also includes self-adaptive parameter control and automatic re-sizing of the population.

It should be remarked that the proposed memetic algorithm performs the local search on the scale factor and thus on one parameter, regardless of the dimensionality of the problem. This kind of hybridization seems to be very efficient in enhancing the offspring generation and have a dramatic impact on stagnation prevention in the Differential Evolution framework. More specifically, these improved solutions seem to be beneficial in ”refreshing” the genotypes and assisting the global search in the optimization process.

Numerical results show that the algorithmic behaviour in 100 dimensions is very promising and the scale factor local search leads to good results in terms of robustness over various optimization problems. The results in 500 and 1000 dimensions show that the standard DE is much affected but the curse of dimensionality. The SACPDE and the jDEdynNP-F can have, in various cases good performance but in some test problems, fail to detect competitive solutions. On the contrary, the proposed DELS appears to be competitive in all the problems analyzed in this chapter, as the Q-tests prove. To be specific, for some test problems the DELS displays performance competitive to other algorithms considered in this study while in other cases significantly outperforms them.

In summary, the scale factor local search in DE frameworks seems to be a powerful component for handling LSPs and appears to be very promising in terms of robustness notwithstanding the complexity of the fitness landscape and high dimensionality characterizing the problem. In this sense, the proposed logic can be potentially be very useful for various real-world applications.

Acknowledgement

This work is supported by IEEE Computational Intelligence Society, Walter J. Karplus Grant, and by Academy of Finland, Akatemiatutkija 00853, Algorithmic Design Issues in Memetic

Computing. The second author would also like to acknowledge the financial support from the FP6 Marie Curie EST project "COFLUIDS", contract number MEST-CT-2005-020327.

References

1. Brest, J., Maučec, M.S.: Population size reduction for the differential evolution algorithm. Applied Intelligence 29(3), 228–247 (2008)
2. Brest, J., Greiner, S., Bošković, B., Mernik, M., Žumer, V.: Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation 10(6), 646–657 (2006)
3. Brest, J., Žumer, V., Maucec, M.: Self-adaptive differential evolution algorithm in constrained real-parameter optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 215–222 (2006)
4. Brest, J., Bošković, B., Greiner, S., Žumer, V., Maučec, M.S.: Performance comparison of self-adaptive and adaptive differential evolution algorithms. Soft Computing 11(7), 617–629 (2007)
5. Brest, J., Zamuda, A., Bošković, B., Maucec, M.S., Žumer, V.: High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction. In: Proceedings of the IEEE World Congress on Computational Intelligence, pp. 2032–2039 (2008)
6. Chakraborty, U.K. (ed.): Advances in Differential Evolution. Studies in Computational Intelligence, vol. 143. Springer, Heidelberg (2008)
7. Das, S., Konar, A.: An improved differential evolution scheme for noisy optimization problems. In: Pal, S.K., Bandyopadhyay, S., Biswas, S. (eds.) PReMI 2005. LNCS, vol. 3776, pp. 417–421. Springer, Heidelberg (2005)
8. Das, S., Konar, A., Chakraborty, U.: Improved differential evolution algorithms for handling noisy optimization problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 2, pp. 1691–1698 (2005)
9. Das, S., Konar, A., Chakraborty, U.K.: Two improved differential evolution schemes for faster global search. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 991–998. ACM, New York (2005)
10. Feoktistov, V.: Differential Evolution in Search of Solutions (2006)
11. García, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC 2005 special session on real parameter optimization. Journal of Heuristics 15(6), 617–644 (2009)
12. Hart, W.E., Krasnogor, N., Smith, J.E.: Memetic evolutionary algorithms. In: Hart, W.E., Krasnogor, N., Smith, J.E. (eds.) Recent Advances in Memetic Algorithms, pp. 3–27. Springer, Berlin (2004)
13. Kononova, A.V., Hughes, K.J., Pourkashanian, M., Ingham, D.B.: Fitness diversity based adaptive memetic algorithm for solving inverse problems of chemical kinetics. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 2366–2373 (2007)
14. Kononova, A.V., Ingham, D.B., Pourkashanian, M.: Simple scheduled memetic algorithm for inverse problems in higher dimensions: Application to chemical kinetics. In: Proceedings of the IEEE World Congress on Computational Intelligence, pp. 3906–3913 (2008)

15. Korošec, P., Šilc, J.: The differential ant-stigmergy algorithm for large scale real-parameter optimization. In: Dorigo, M., Birattari, M., Blum, C., Clerc, M., Stützle, T., Winfield, A.F.T. (eds.) ANTS 2008. LNCS, vol. 5217, pp. 413–414. Springer, Heidelberg (2008)
16. Lampinen, J., Zelinka, I.: On stagnation of the differential evolution algorithm. In: Ošmera, P. (ed.) Proceedings of 6th International Mendel Conference on Soft Computing, pp. 76–83 (2000)
17. Liu, Y., Zhao, Q.: Scaling up fast evolutionary programming with cooperative coevolution. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1101–1108 (2001)
18. Lozano, M., Herrera, F., Krasnogor, N., Molina, D.: Real-coded memetic algorithms with crossover hill-climbing. Evolutionary Computation, Special Issue on Memetic Algorithms 12(3), 273–302 (2004)
19. Marchiori, E., Steenbeek, A.: An evolutionary algorithm for large scale set covering problems with application to airline crew scheduling. In: Oates, M.J., Lanzi, P.L., Li, Y., Cagnoni, S., Corne, D.W., Fogarty, T.C., Poli, R., Smith, G.D. (eds.) EvoIASP 2000, EvoWorkshops 2000, EvoFlight 2000, EvoSCONDI 2000, EvoSTIM 2000, EvoTEL 2000, and EvoROB/EvoRobot 2000. LNCS, vol. 1803, pp. 367–381. Springer, Heidelberg (2000)
20. Moscato, P., Norman, M.: A competitive and cooperative approach to complex combinatorial search. Tech. Rep. 790 (1989)
21. Neri, F., Tirronen, V.: Scale factor local search in differential evolution. Memetic Computing Journal 1(2), 153–171 (2009)
22. Noman, N., Iba, H.: Enhancing differential evolution performance with local search for high dimensional function optimization. In: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 967–974. ACM, New York (2005)
23. Noman, N., Iba, H.: Accelerating differential evolution using an adaptive local search. IEEE Transactions on Evolutionary Computation 12(1), 107–125 (2008)
24. Olorunda, O., Engelbrecht, A.: Differential evolution in high-dimensional search spaces. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1934–1941 (2007)
25. Ong, Y.S., Keane, A.J.: Meta-lamarkian learning in memetic algorithms. IEEE Transactions on Evolutionary Computation 8(2), 99–110 (2004)
26. Potter, M.A., De Jong, K.: Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evolutionary Computation 8(1), 1–29 (2000)
27. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866, pp. 249–257. Springer, Heidelberg (1994)
28. Price, K.V., Storn, R., Lampinen, J.: Differential Evolution: A Practical Approach to Global Optimization. Springer, Heidelberg (2005)
29. Qin, A.K., Suganthan, P.N.: Self-adaptive differential evolution algorithm for numerical optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 2, pp. 1785–1791 (2005)
30. Rahnamayan, S., Wang, G.G.: Solving large scale optimization problems by opposition-based differential evolution (ode). WSEAS Transactions on Computers 7(10), 1792–1804 (2008)
31. Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.: Opposition-based differential evolution. IEEE Transactions on Evolutionary Computation 12(1), 64–79 (2008)

32. Rechemberg, I.: Evolutionstrategie: Optimierung Technisher Systeme nach prinzipien des Biologishen Evolution. Fromman-Hozlboog Verlag (1973)
33. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 2nd edn., pp. 111–114. Prentice-Hall, Englewood Cliffs (2003)
34. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures. CRC Press, Boca Raton (2000)
35. Shi, Y.J., Teng, H.F., Li, Z.Q.: Cooperative co-evolutionary differential evolution for function optimization. In: Wang, L., Chen, K., Ong, Y.S. (eds.) ICNC 2005. LNCS, vol. 3611, pp. 1080–1088. Springer, Heidelberg (2005)
36. Sofge, D., De Jong, K., Schultz, A.: A blended population approach to cooperative co-evolution fordecomposition of complex problems. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 413–418 (2002)
37. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. Rep. TR-95-012, ICSI (1995)
38. van den Bergh, F., Engelbrecht, A.P.: A cooperative approach to particle swarm optimization. IEEE Transactions on Evolutionary Computation 8(3), 225–239 (2004)
39. Yang, Z., Tang, K., Yao, X.: Differential evolution for high-dimensional function optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 3523–3530 (2007)
40. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Information Sciences 178(15), 2985–2999 (2008)
41. Zamuda, A., Brest, J., Bošković, B., Žumer, V.: Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution. In: Proceedings of the IEEE World Congress on Computational Intelligence, pp. 3719–3726 (2008)
42. Zielinski, K., Weitkemper, P., Laur, R., Kammeyer, K.D.: Parameter study for differential evolution using a power allocation problem including interference cancellation. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1857–1864 (2006)

Chapter 13
Large-Scale Network Optimization with Evolutionary Hybrid Algorithms: Ten Years' Experience with the Electric Power Distribution Industry

Pedro M.S. Carvalho and Luis A.F.M. Ferreira

Abstract. Electric power distribution networks are large-scale infrastructures that need to be planned regularly and operated continuously. The planning and operation tasks involve difficult decision-making processes that can be formulated as optimization problems: large-scale combinatorial optimization problems. These problems have been addressed successfully with specially designed evolutionary hybrid approaches. Such approaches rely upon Lamarckian evolutionary hybrid algorithms. In this chapter, we present the most successful implementations of such algorithms and discuss such implementations based on our experience in the development of industrial applications for planning and operation of electric power distribution networks for a period of over ten years.

13.1 Introduction

Electric power distribution network planning and operation is the subject matter of most of the research conducted in network optimization in power systems since the 90's. The problem has been formulated in many different ways but its solution always relies on computationally expensive optimization approaches [1]–[6]. Realistic formulations lead to large-scale combinatorial problems where the objective function and constraints are not possible to express analytically. We have been working on several instances of the problem since the early 90's and succeed to deploy industrial applications to solve such problems with evolutionary based algorithms since 1997 [7]–[10]. In the following we state the network planning and operation problems focusing on the problem aspects that lead to the main optimization difficulties.

Pedro M.S. Carvalho · Luis A.F.M. Ferreira
Instituto Superior Técnico, Technical University of Lisbon,
Av. Rovisco Pais, 1049-001 Lisbon
e-mail: pcarvalho@ist.utl.pt, lmf@ist.utl.pt

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 325–343.
springerlink.com

Both network planning and operation problems are hard optimization problems but for different reasons. Distribution network planning consists in choosing a new distribution system from a set of possible distribution systems so as to meet the expected load profile in a better way, more reliably, and with fewer losses. The new distribution system is a plan, a plan to be carried out by project services if the plan comprises acquisition or installation of new equipment or a plan to be carried out by dispatch services if the plan involves only changes in the network configuration (i.e., switching operations). If one can define criteria to measure the goodness of a plan, then there will be one plan that ranks higher than others; that plan will be the optimal distribution plan. Finding such plan is computationally difficult because:

1. The number of possible plans is very large, as distribution networks have thousands of nodes and thousands of branches and new investments in one network area impact considerably in neighbor areas (see Fig. 13.1 for an illustration of a medium voltage distribution network);
2. The criterion to measure the goodness of a plan is complex. Plan analysis involves complex judgment usually impossible to express analytically, e.g., criteria involve reliability and security analysis which must be carried out by simulation for each candidate plan.

Distribution network operation consists in several different network dispatch activities. The most computationally demanding activity is to follow contingency conditions by attenuating the effects of the contingency and leading the distribution system to a satisfactory point of operation (when possible). For a given contingency, the problem consists in selecting and sequencing a set of switching operations to restore power in a secure and prompt manner. The problem is dynamic. The switching operations require a substantial cumputational effort to be sequenced in a securely and optimal manner. The sequence must be investigated in order to keep the network radial, minimize customer outage costs and not violate voltage and branch capacity constraints during the several reconfiguration stages [11]–[12]. The dynamic restoration problem can be addressed in two phases: *(i)* in the first phase a network optimization approach finds the post-contingency final configuration; and *(ii)* in the second phase an optimal sequencing approach finds the order of the switching operations that better changes the original network configuration into the post-contingency final configuration. Finding the post-contingency final configuration is a hard optimization problem because the network optimization objective is twofold: finding a secure configuration is not enough - one must find one that does not involve many switching operations in order to be able to promptly restore power to the maximum number of customers [1].

Both planning and operation problem solutions rely upon network optimization. Network optimization is a broad area of research. In this chapter we address a particular type of network optimization - radial network optimization. The chapter is organized as follows. In Sect. 13.2 we formulate the planning and operation problems as network optimization problems. In Sect. 13.3 we present the evolutionary solution approach together with the necessary framework to deal effectively with the

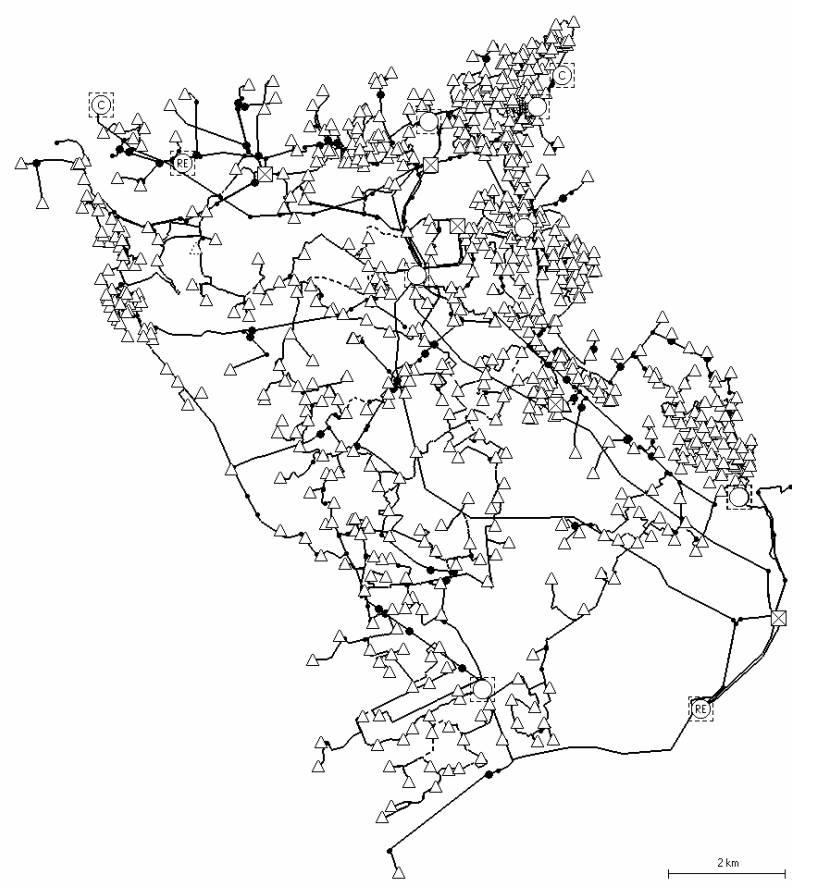


Fig. 13.1 Geographic representation of a medium-voltage distribution network. Circles represent substations (feeding points) and triangles low-voltage transformation stations (load points)

network topology constraints. In Sect. 13.4 we present application examples and use these to discuss implementation practicalities. Section 13.5 concludes the chapter.

13.2 Optimization of Electric Power Distribution Networks

Electrical power distribution networks are composed of thousands of nodes, most of which correspond to power delivery points or load points, and thousands of branches, most of which correspond to electrical cables or lines. The other nodes correspond to connecting points, and the other branches correspond to switching busbars. From the topology perspective the physical network infra-structure can be represented by a graph G.

In normal operation, each node of the graph is connected to a single power delivery point through a single path. The operating network configuration is radial and connected. Thus, from the topology perspective the operating configuration of the network can be represented by a spanning-tree T of the graph G. See Fig. 13.2 for an illustration of the relationship between the electrical network topology and the graph concepts.

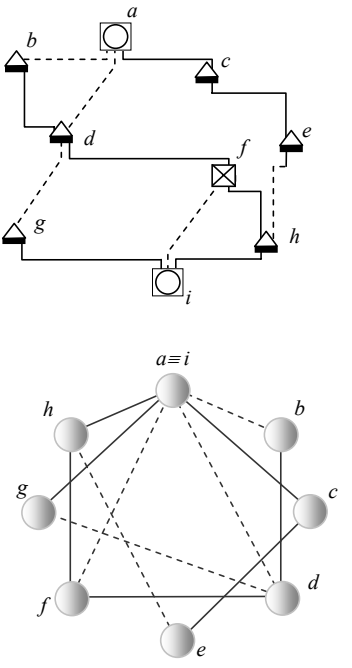


Fig. 13.2 Schematic representation of an electrical network (upper diagram), and its correspondent graph and spanning tree solution (lower diagram). The figure shows a small-scale network with two power delivery points (busbars a and i), six load points (busbars $b, c, d, e, g,$ and h), and a connecting point (busbar f). The dashed lines identify the network branches not used by power flow purposes. In the graph representation, the two delivery points are represented by a single node (the tree root $a \equiv i$), the spanning-tree arcs are represented by solid lines, and the co-tree arcs are represented by dashed lines

Many operating configurations can be found for the same network infra-structure. The co-tree arcs of the graph can be used to change the operating topology so as to improve its performance. It may also happen that even the optimal operating topology will not be satisfactory. In such case, the network infra-structure must be upgraded or expanded, thus leading to investment costs in new cables and switching busbars. The problem of finding the optimal operating network configuration can be formulated as in the following:

$$(P) \qquad \text{Minimize } f(\mathrm{T}) \text{ over all } \mathrm{T} \in \mathrm{G}$$

Where,

f: Operating cost and investment cost function;
T: Spanning tree of G;
G: Graph of the physical network infra-structure.

Problem (P) falls into general network problem formulations. It can be stated as connecting all nodes by selecting a spanning-tree T (out of graph G) to minimize f. The specificities of this problem are that the objective function is non-analytical. The operating costs must involve at least efficiency and reliability costs.

Efficiency is determined by computing the electrical losses in the network cables (Joule losses) and transformers (Joule, Foulcault and hysteresis losses), which must be obtained after finding the network node voltages and branch currents. The voltages and currents depend on loads and are obtained by running an AC power flow given the network configuration. The AC power flow problem is non-linear and is usually solved by Newton-like algorithms.

Reliability must be obtained by simulation analysis of possible faults [2, 3]. Faults in distribution networks cause the trigger of one or more breakers upstream the fault. The opening of a breaker is followed by a sequence of switching actions to isolate the fault and restore power to the downstream load points (customers). Some switching actions are automatic others are manual; the manual switching can be remote-controlled or operated in site. Some switching actions may cause additional customer interruptions. Reliability is a measure of the interruption impact. Several reliability indices can be defined. A popular index is the value of the expected Energy Not Supplied (ENS). Such value is a function of *(i)* the number of faults, *(ii)* the chosen sequence of switching actions and their operating times, and *(iii)* the load demand of the customers interrupted during the sequence. The function is complex. Instead of expressing it analytically or even formulating it mathematically, we describe it as follows.

A fault in a line-section leads to the automatic opening of the feeder breaker. If the fault is fugitive, the feeder is usually reclosed successfully in very short time and, therefore, the energy that is not supplied is negligible (very short interruptions are usually not included in system average reliability indices). If the fault is persistent, then either the breaker reopens the feeder or some line-section upstream automatic device isolates the fault from the feeder, in which case the breaker recloses. The breaker or the automatic device stays opened until the faulted line-section is isolated. After fault isolation, the breaker or the automatic device may be closed so as to supply the customers upstream from the fault.

After fault isolation, the network may be reconfigured to feed the customers downstream the fault. When normally opened devices exist, (one of) these may be closed to feed the downstream customers. However, if the downstream demand is high, the backup circuit prompted by closing the normally opened device may not be able to feed all customers without circuit overloads. If overloads appear, some demand must be discarded. If the backup circuit cannot feed all customers, the discarded demand stays out of service during fault repair time. After repairing the

cable or line, the network may return to its original configuration without additional interruptions.

The expected value of the ENS is obtained by summing the contributions of every switching step, for every possible line-section fault, and multiplying the result by the line-section fault probability.

Synthetically, the objective function f can be defined by the sum:

$$f(\mathrm{T}) = i(\mathrm{T}) + e(\mathrm{T}) + r(\mathrm{T}) \tag{13.1}$$

Where,

i: Investment cost
e: Efficiency cost (from AC power flow losses)
r: Reliability cost (from fault simulation)

Both e and r are strongly dependent on the configuration T as losses are quadratic functions of branch-currents and ENS is strongly dependent on branch failure rates and neighbor switching capabilities.

Problem (P) can get more complex [6]. Increased complexity results from: *(i)* node information (load, for instance) being considered as a stage dependent variable. In that case the problem becomes dynamic, as decisions must be scheduled to gather a sequence of network solutions, and *(ii)* information for future stages being considered uncertain. This happens when important future investments stand a chance of being impossible to realize, or some important future information is unknown. If uncertainty is also considered, the problem becomes a stochastic-dynamic problem.

The more complex versions of the network optimization problem can however be decoupled into sequences of (P)-like problems [13]. Here, we will address the solution of (P)-like problems with evolutionary algorithms.

13.3 Evolutionary Approach

Evolutionary algorithms (EA) are algorithms that manage the survival of a set of solutions based on the principles of natural selection (neo-Darwinian evolution). Such principles are: *(i)* generation sequencing, *(ii)* survival of the fittest, and *(iii)* genetic exchange [14].

Evolution takes place in time (generation sequence) if only the fittest get a chance (competition) to combine their genes (mating), and this way contributing to the subsequent generation. Such an evolutionary process can be synthetically presented as the following algorithm.

Evolutionary Algorithm

Make $t = 0$;
Initialize the population $p(0)$, at random.
Evaluate $p(0)$

Repeat Steps 1 to 4 (until close to genetic saturation)
Step 1 $t \leftarrow t+1$
Step 2 Select the fittest from $p(t-1)$ to build $p(t)$
Step 3 Change $p(t)$
Step 4 Evaluate $p(t)$

The algorithm ends when the genetic material represented by the solutions of the population is no longer diverse enough, i.e., when close to the so-called genetic saturation.

The change in Step 3 is a crucial process of any evolutionary algorithm. Usually, it is undertaken in two independent processes:

1. A solution-pair recombination process where information between solutions is exchanged (this process is usually denoted by crossover); the exchange should be designed to be a solution-pair neighborhood variation that explores the differences and (more important) keeps the likenesses of pairs; and
2. an individual solution arbitrary information change (usually denoted by mutation); these changes are usually designed to be local and rare.

Evolutionary algorithms that use these two processes are usually called Genetic Algorithms (GA).

13.3.1 Working within the Feasibility Domain

The problem (P) requires satisfaction of nontrivial constraints. Feasible solutions of (P) must be spanning-trees of a graph. The canonical recombination operators can hardly transmit radiality and connectivity to the offspring. Even when they do so, important similarities about solutions can hardly propagate genetically. We will explain why in the following.

A trivial representation of a spanning tree of a graph consists in identifying the arcs from G that belong to the tree. That could be done by defining an array of the graph arcs and by using a binary array of the same size to identify the ones that belong to the tree. Canonical recombination operators could then be defined as variants of the one point crossover operator that we state in the following.

Canonical Recombination. Let **A** be the array of all arcs, e.g., [(a-b); (b-c); ... (e-f)]. Let $\mathbf{B}^i$ and $\mathbf{B}^{ii}$ be binary arrays (strings), e.g., [1;0 ... 1] if (a-b) and (e-f) are tree arcs, and (b-c) is a co-tree arc.

Step 1: Randomly select a position p in the string A.
Step 2: Swap the first p binary information between solutions, i.e., substitute $\mathbf{B}^i$ [1:p] by $\mathbf{B}^{ii}$ [1:p], and $\mathbf{B}^{ii}$ [1:p] by $\mathbf{B}^i$ [1:p].

The binary array representation is very poor as it defines a solution domain much larger than the space of spanning trees. Note that the domain of the binary array representation is 2^m where m is the number of graph arcs and the space of spanning-trees is much smaller than that [15]. In such circumstances, the canonical recombination operators would hardly find a feasible solution in the binary string domain.

Another popular operator that has been proposed by several under different names is the so-called edge-set encoding [16]–[18] that consists in superimposing both parents to create a subgraph of G and then randomly generating two spanning trees of such subgraph as possible offspring. This is a simple idea that guarantees offspring feasibility but the result it is not effective. The tree generating process (Prim-like) is time consuming and too much random, which leads to slow convergence to the optimum. We state such operator in the following under the name of Tree Generation.

TG Recombination. Let H, T^i and T^{ii} be subgraphs of G and G be defined by the pair (A, N) where A is the set of arcs and N is the ser of nodes N.

Step 1: Built $H = T^i \cup T^{ii} = (A^i \cup A^{ii}, N)$ as the subgraph of G with the arcs of the two spanning-trees only.
Step 2: Randomly generate the offspring trees T^i and T^{ii} as spanning-trees of H.

In the following we propose a more natural problem-related representation of spanning trees. In our approach we propose to recombine by interchanging paths between solutions. The idea is to take the information to be interchanged between solutions as sub-networks of each solution. As sub-networks, connectivity and radiality can be ensured and meaningful information gets propagated along generations [5].

The main idea behind our recombination approach will be presented together with the theoretical results that allow going into the implementation details. We start by defining the spanning tree genotype space as a partially ordered set of nodes $(A, \leq)$, i.e., a set where:

(i) $a \leq a$;
(ii) $a \leq b$ and $b \leq a$ implies $a = b$;
(iii) $a \leq b$ and $b \leq c$ implies $a \leq c$, for every $a, b, c \in$ A [20].

An element a is called the *direct precedent* of the element b in A, iff: (i) $a \neq b$; (ii) $a \leq b$; (iii) there is no element $c \in$A such that $a \leq c$ and $c \leq b$. The relation is denoted by $b \hookleftarrow a$. Similarly, an element b is denoted the *direct follower* of an element a in A, iff a is one of its direct precedents. The elements of the set are the nodes of the spanning-tree, and the order relation $a \leq b$ denotes that node a precedes node b on the path from a to b. Spanning trees have a specific property as partially ordered sets: each tree element is preceded directly by one and just one single element; an exception is made for the first element (the root), which is not preceded.

Then, we define possible changes as changes that do not violate order as defined in properties (i)-(ii)-(iii). We call these consistent changes. Take Lemma 1 to identify non-consistent changes.

Lemma 1. *A b direct-precedence change $b \hookleftarrow a$ taken over a tree ordered set T violates order (i)-(ii)-(iii) iff $b \leq a$.*

Proof. Sufficiency – If $b \leq a$ there exists in T a direct ordered sequence like $a \hookleftarrow x \hookleftarrow y \hookleftarrow \ldots \hookleftarrow b$. A change $b \hookleftarrow a$ forces a circulation $a \hookleftarrow x \hookleftarrow y \hookleftarrow \ldots \hookleftarrow b \hookleftarrow a$, and thus an order violation (property-ii).

Necessity – If $b \leq a$ does not apply, either (1) $a \leq v$, or (2) no order exists between a and b. In case (1), a change $b \hookleftarrow a$ eliminates the order relationship between every $x : a \leq x \leq b$ and $y : b \leq y$ by eliminating the existent b-precedence. The order of the x-elements is not changed: the x-elements remain as followers of a. The same applies for the y-elements, they remain as followers of b, and by change $b \hookleftarrow a$, also followers of a. In case (2), a change $b \hookleftarrow a$ forces b to become a follower of a, and thus every $y : b \leq y$ becomes a follower of a, instead of being a follower of the existing $p(b)$. □

Lemma 1 allows classifying direct precedence changes as consistent or nonconsistent. When consistent, direct precedence changing is a simple way to change tree information, guaranteeing network radiality and connectivity. Simplicity is important but is not enough. Information to be interchanged should also be meaningful. One simple and meaningful information structure of a network is a path between two nodes of the spanning-tree. We propose to interchange path information between solutions as a recombination operator.

Paths can be interchanged between spanning trees if they do not enclose inconsistencies. A path is not just a set – it is a partially ordered set – and thus precedence change consistency must be tested orderly. Path precedence relationships must be submitted and tested, starting from the path's smallest element to the largest one, by the order defined in the path itself. The following algorithms summarize the path interchange approach:

Path Interchange Algorithm. (Submit a path P to a tree T)
Name a as the path's smallest element. Denote by $F(x)$ the set of direct followers of x in the path P. Consider a set E of tree elements, and start by setting it to $\mathrm{E} = F(a)$.

Step 1: Change T by changing every precedence relation $x \hookleftarrow y$ of T to $x \hookleftarrow z$ of P, iff (i) $x \in E$, and (ii) $x \hookleftarrow z$ is consistent in T.
Step 2: Update T, set $\mathrm{E} = \cup F(x \in \mathrm{E})$, and repeat Step 1 until $\mathrm{E} = \phi$.

Recombination Algorithm. (Interchange paths between solutions T^i and T^{ii})

Step 1: Randomly select two nodes, a and b.
Step 2: Find the paths P^i and P^{ii} between a and b: P^i in T^i and P^{ii} in T^{ii}.
Step 3: If possible, submit P^i to T^{ii}, and P^{ii} to T^i. If not possible, go to Step 1.

Recombination Example. (Recombine solutions T^i and T^{ii}) Solutions T^i and T^{ii} are represented in Fig. 13.3. The descendants are represented in Fig. 13.4. P is the path between the two randomly selected nodes, say 1 and 6. Consider 1 as the path's smallest element. The procedure is summarized in the following:

Step 1: Represent the solutions as tree ordered sets $\mathrm{T}^i = \{2 \hookleftarrow 1, 5 \hookleftarrow 1, 4 \hookleftarrow 2, 3 \hookleftarrow 5, 6 \hookleftarrow 2, 7 \hookleftarrow 3, 8 \hookleftarrow 7\}$ and $\mathrm{T}^{ii} = \{5 \hookleftarrow 1, 8 \hookleftarrow 1, 4 \hookleftarrow 5, 6 \hookleftarrow 8, 7 \hookleftarrow 8, 2 \hookleftarrow 6, 3 \hookleftarrow 7\}$ and the paths P^i in T^i and P^{ii} in T^{ii} as $\mathrm{P}^i = \{2 \hookleftarrow 1, 6 \hookleftarrow 2\}$ and $\mathrm{P}^{ii} = \{8 \hookleftarrow 1, 6 \hookleftarrow 8\}$

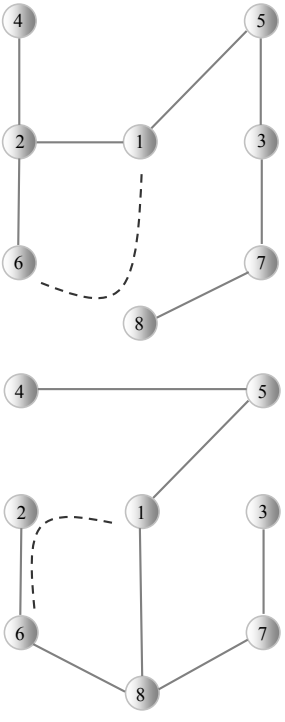


Fig. 13.3 Two spanning trees T^i and T^{ii} (upper and lower figures) and the complementary paths between nodes 1 and 6, respectively P^{ii} and P^i (in dashed line)

Step 2: Submit $P^i = \{2 \hookleftarrow 1, 6 \hookleftarrow 2\}$ to $T^{ii} = \{5 \hookleftarrow 1, 8 \hookleftarrow 1, 4 \hookleftarrow 5, 6 \hookleftarrow 8, 7 \hookleftarrow 8, 2 \hookleftarrow 6, 3 \hookleftarrow 7\}$ Node 1 is the path P^i smallest element. $F(1) = \{2\}$. The change $2 \hookleftarrow 1$ is a consistent change in T^{ii} as node 2 is also a descendent of node 1 in T^{ii}. Note that $6 \hookleftarrow 2$ is not consistent in T^{ii} at this stage. By updating the tree with $2 \hookleftarrow 1$ (in bold) one gets $T^{ii} = \{5 \hookleftarrow 1, 8 \hookleftarrow 1, 4 \hookleftarrow 5, 6 \hookleftarrow 2, 7 \hookleftarrow 8, 2 \hookleftarrow 1, 3 \hookleftarrow 7\}$, in which $2 \hookleftarrow 6$ changes to $6 \hookleftarrow 2$. So, the change of the second element of the path is no longer necessary. The result of the path submission in shown in Fig. 13.4.

Now submit $P^{ii} = \{8 \hookleftarrow 1, 6 \hookleftarrow 8\}$ to $T^i = \{2 \hookleftarrow 1, 5 \hookleftarrow 1, 4 \hookleftarrow 2, 3 \hookleftarrow 5, 6 \hookleftarrow 2, 7 \hookleftarrow 3, 8 \hookleftarrow 7\}$. Element 1 is the smallest element of the path P^{ii}. $F(1) = \{8\}$. $8 \hookleftarrow 1$ is a consistent change as $8 \leq 2$ in T^i. Changing the precedence results in the tree update $\{2 \hookleftarrow 1, 5 \hookleftarrow 1, 4 \hookleftarrow 2, 3 \hookleftarrow 5, 6 \hookleftarrow 2, 7 \hookleftarrow 3, 8 \hookleftarrow 1\}$. The follower of 8 in P^{ii} is $6, F(8) = 6$. The change $6 \hookleftarrow 8$ is again a consistent one as there is not an order relationship between node 6 and node 8 in T^i. The change results in the spanning-tree $\{2 \hookleftarrow 1, 5 \hookleftarrow 1, 4 \hookleftarrow 2, 3 \hookleftarrow 5, 6 \hookleftarrow 8, 7 \hookleftarrow 3, 8 \hookleftarrow 1\}$. The result of the path submission in shown in Fig. 13.4.

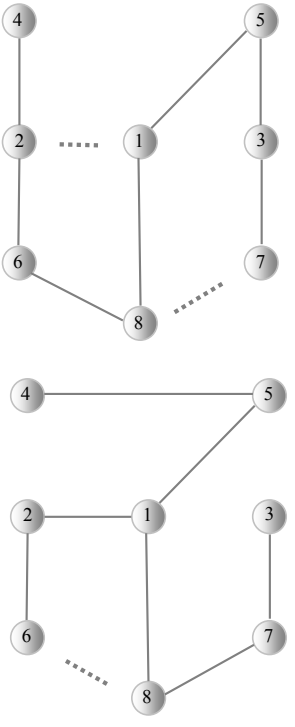


Fig. 13.4 Two descendant spanning trees that result from path interchange between node 1 and node 6 and the branches that were lost in each spanning tree (dotted line). The figures represent the changed spanning trees T^{i} and T^{ii} (upper and lower figures). The originals are represented in Fig. 13.3

13.3.2 Lamarckian Hybridization

GAs abstract the fundamental process of neo-Darwinian evolution, namely selection and genetic variation through crossover and mutation. According to neo-Darwinian evolution, these are the unique processes that change genotype information. However, in the early nineteenth century, Lamarck suggested that characteristics of organisms acquired during lifetime could be inherited by their descendants. Despite this theory being discredited nowadays, the idea of modifying genotypes during the individual lifetime and allowing the modification to be transmitted to the offspring is a very interesting idea (in what concerns optimization).The lifetime modification can be understood as lifetime learning and be implemented by undertaking individual local improvements. This is the central idea of most hybridization processes and is known as Lamarckian learning. Such an evolutionary hybrid process can be written as an algorithm like the following:

Lamarckian Hybrid

Make $t = 0$;
Initialize the population $p(0)$, at random.
Evaluate $p(0)$
Repeat Steps 1 to 5 (until close to genetic saturation)
Step 1 $t \leftarrow t + 1$
Step 2 Select the fittest from $p(t-1)$ to build $p(t)$
Step 3 Recombine $p(t)$ and mutate $p(t)$
Step 4 Improve $p(t)$ with local heuristics
Step 5 Evaluate $p(t)$

Some implementation difficulties spring out when trying to implement Step 4. The difficulties rely upon deciding what solutions will be improved and in what extent they will be improved. To answer this, note that: *(i)* if too many solutions are to be improved, the GA process will become too slow – and would be the merit of improving very bad solutions if they will barely survive after all, and *(ii)* if solutions are to be improved in a high extent, population will lose diversity as solutions will tend to be very much alike (each one similar to its "closer" local optimum).

Some authors have proposed empirical rules for undertaking Lamarckian steps (e.g., the Rule of 10); others have proposed theories for coordinating global and local search [19]. Here, we present a very simple but effective approach to coordinate local search effort with global search effort. We call it Diversity Driven hybridization. Despite being very simple to implement, the presented coordination approach observes population diversity and solution relative quality. The approach is summarized in the following two steps.

Diversity Driven Hybridization

Step 1: Given a population of solutions to be improved (Step 4), identify the subset of solutions that have at least one clone (a copy) in $p(t)$. Name this set $q(t)$ and remark that $p(t) \setminus q(t)$ has the same genetic material as $p(t)$.
Step 2: Use local search to improve some solutions of $q(t)$. Randomly choose *(i)* the solutions of $q(t)$ to be improved, e.g., with a fixed probability, as well as *(ii)* the number of local improvement to make in each solution.

The coordination process just presented is made dependent on diversity as local improvements are rare in the GA insipient generations (where populations are much diverse) but frequent for the ending generations (where populations are close to genetic saturation).

Moreover, the coordination process is also made dependent on the solution quality as local search does not spend considerable effort in locally improving bad solution as these do not get many chances of having a significant number of copies in future generations.

But what would be the definition of local neighborhood in the context of network optimization? In the context of network optimization, local improvements can be seen as any optimization subproblem in a small neighborhood. We propose that the

neighborhood is defined as a fundamental cycle of the graph and the subproblem be formulated as a (Q)-like problem.

$$(Q) \qquad \text{Minimize } f(y) \text{ over all } y \in Y$$

Where,

f : Operating and investment cost function
y : Co-tree arc of Y
Y: Fundamental cycle of the graph G
G : Graph of the physical network infra-structure

Problem (Q) is very simple when compared to (P). The fundamental cycles of the graph G with respect of a spanning tree T defined by co-tree arcs represent the operating network open-loops. For a single of such cycles, the subproblem objective function is often convex in the space of possible co-tree arcs [20], and can be solved approximately very easily.

Two questions seem pertinent about the proposed hybridization procedure: *(i)* what should be the number of fundamental cycle changes to be operated in each solution, a single one, $y \in Y$, or a generalized operation, $y \in \text{G} \setminus \text{T}$ (single vs. multiple changes), and *(ii)* why should local optimization be performed on non-diverse solutions only. The rationale is the following:

1. Local optimization must not perform exhaustive modifications on each solution – that would lead to a dramatic diversity lack
2. Above average solutions are more likely to get copies in the descendant's generation – selection is a competitive mechanism
3. Local modifications performed at above average solutions are more likely to propagate to the descendants.

The proposed hybridization has the additional advantage of guaranteeing local optimality, which is a very important aspect for industry applications. See the following result where optimality is related to genetic saturation.

Lemma 2. *Genetic convergence guarantees* (Q)*-optimality.*

Proof. Genetic convergence presumes the stable absence of diversity, i.e., $p(t) \equiv q(t)$. If a solution is not (Q)-optimal, then it is possible to perform a single loop reconfiguration to improve such solution, and as improved, selection will guarantee its propagation to the descendants. So, for a non empty random subset q, (Q) sub-optimal solutions are unstable. □

13.4 Application Examples and Illustration

In this section we present two application examples that illustrate the main difficulties in addressing large-scale distribution optimization problems. Scale influences algorithm robustness if genetic operators and parameters do not change accordingly.

Knowing how to change is not an easy task. We present the results of our experience and discuss their limitations.

We start by solving a network operation problem. We chose to optimize the operating configuration of the distribution network represented in Fig. 13.1. The network has nine substations, 1613 nodes from which 768 are load points and 1667 branches, which are connected through 35 different feeders (sub-trees). The network has a 300 MVA installed capacity for a peak load of 185 MVA. High-voltage losses are 1 GWh/year and medium-voltage losses are 6 GWh/year. Yearly operational costs include losses costs and reliability costs amounting to about 1 M€/year.

We start from the actual configuration and generate a population of 120 random solutions by undertaking a random number of fundamental cycle changes over the actual spanning-tree configuration. Changes are made sequentially for each individual configuration. The generated population has a cost distribution that varies between 0.9 M€ and 3.4 M€ (see Fig. 13.5).

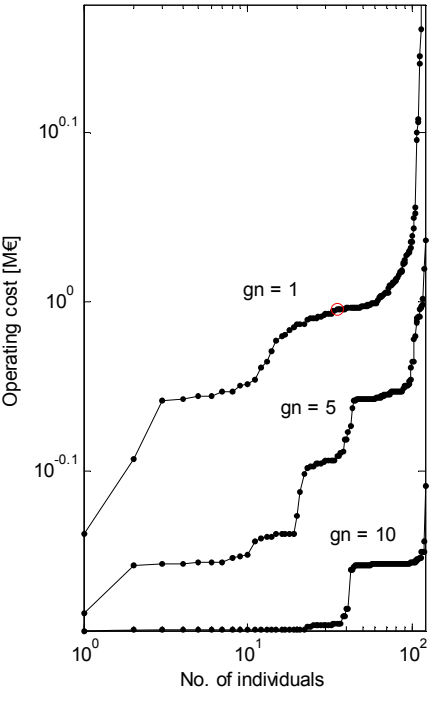


Fig. 13.5 Operating cost distribution for generations 1, 5, and 10. In generation number 1 most of the individuals have costs around 1 M€, in generation number 5 there is already a large distribution of the costs, some still have high values but many are already below 800 k€, and after 10 generations most are already below 650 k€. The initial cost that corresponds to the actual configuration is shown in the figure as a circle

Then, we select the better solutions from this population with binary tournaments without elitism and recombine 80% of the selected configurations by interchanging

paths between spanning-trees. We do not mutate. From the set of recombined configurations we find out clones (repeated solutions) and modify these by undertaking fundamental cycle changes to solve (Q)-like subproblems. The modified population is then evaluated by computing f before going again into the selection process.

This sequence continues for 20 generations until genetic saturation is achieved (see Fig. 13.6). The result obtained has a total operating cost of 650 k€, which represent a cost reduction of 35%. This has been possible in such few generations because the operators are very effective for the problem at hand. Parameters are also important for effectiveness. How many paths did we submitted when recombining? How many cycle changes did we undertake when improving clones? These are important questions that experience can answer.

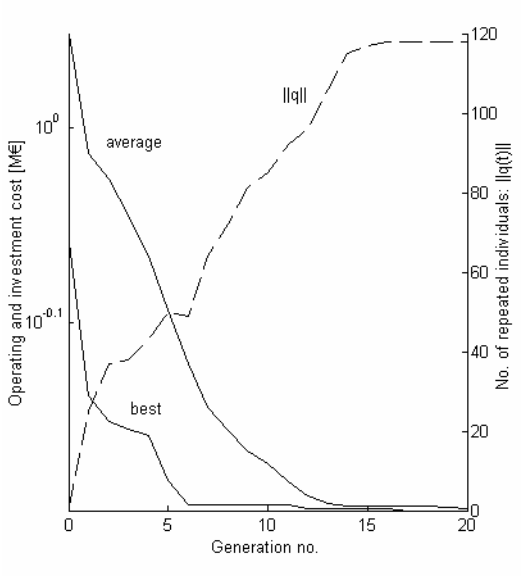


Fig. 13.6 Generation evolution of the operating cost and the corresponding number of repeated individuals that are locally optimized by solving Q-subproblems

Before answering the questions on parameters used, let us solve a network planning problem, i.e., an optimization problem that also involves investment possibilities. That problem is similar to the operation problem but harder. It is harder because the investment costs increase significantly the spanning-tree building block cost variance, the so-called collateral noise [21]. A small change either in a path or in a cycle might be responsible for a big difference in performance. That makes the evolutionary algorithm job much more difficult.

For the planning problem, we use the same distribution network as before (the network represented in Fig. 13.1) but now with some of the nodes and some of branches as new possible investments. The starting investment plan involves investment costs of 330 k€, high-voltage losses of 1 GWh/year and medium-voltage

losses 8 GWh/year. Part of the network shows serious under-voltage and over-current problems, which are penalized. Penalties related to electrical constraints amount to 315 k€. Yearly operational costs include losses costs and reliability costs amounting to about 2 M€/year.

Like before, we start from the actual configuration plan and generate a population of 120 random solutions. The generated population has a cost distribution that varies between 1.6 M€ and 7.2 M€ (see Fig. 13.7). The evolutionary algorithm evolves now for 37 generations until genetic saturation is achieved (see Fig. 13.8). The process is now longer than before (for the operation problem) and also more sensitive to the algorithm parameters. We will address this problem in the following.

The evolution of the process depends on the classical GA parameters, such as crossover probability, population size, etc., but also on other parameters that are required by our specific approach. These parameters are *(i)* the number of paths exchanged, np, at the recombination of two individuals, and *(ii)* the number of cycle changes, nc, undertaken in each repeated solution (for each clone).

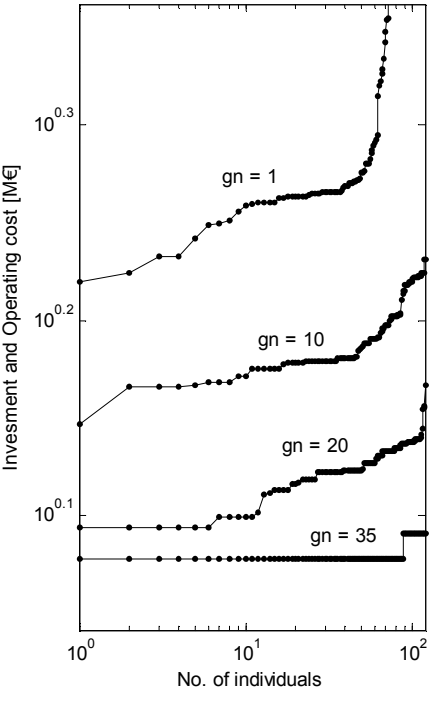


Fig. 13.7 Operating and investment cost distribution for generations 1, 10, and 20 and 35. In generation number 1 most of the individuals have costs around 2 M€, in generation number 10 there is already a large number of individuals with costs around 1.6 M€, and after 20 generations most are already around 1.35 M€. The population saturates for an optimum cost is below 1.2 M€ after generation number 33

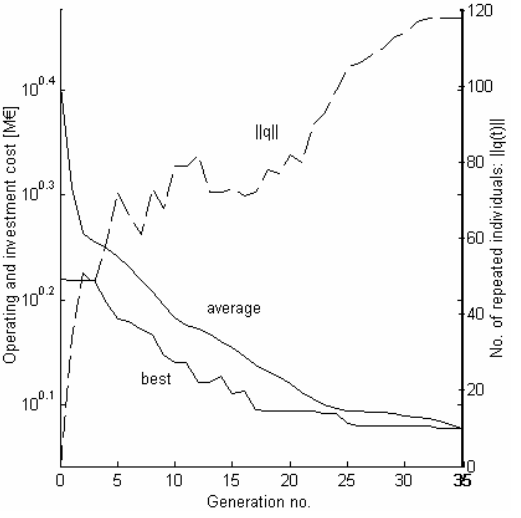


Fig. 13.8 Generation evolution of the investment and operating cost and the corresponding number of repeated individuals that are locally optimized by solving Q-subproblems

Our experience with distribution networks lead to the conclusion that these numbers should be random but bounded. Upper and lower bounds can be defined for the number of paths to be exchanged, say np^U and np^L, and for the number of cycles to be changed, say nc^U and nc^L. The bounds can be defined as a function of the number of feeders, nf (independent sub-trees). Simple functions can be used with good results. In the cases presented previously we used the bounds given in Table 13.1.

Table 13.1 Bounds for the number of paths to be exchanged and the number of cycles to be changed

Name	np^L	np^U	nc^L	nc^U
Value	1	$\lfloor 0.2nf \rfloor$	0	$\lfloor 0.15nf \rfloor$

In Table 13.1, the function $\lfloor \cdot \rfloor$ is the floor function that maps a real number to the next smallest integer. In the example above, the network had 35 feeders, for which the upper bounds were set to $np^U = 7$ and $nc^U = 5$. Increasing the upper bounds leads to a decrease in the algorithm performance, e.g., if one sets $np^U \approx nf$ the cost of the final solution yielded for the planning problem would become 10% higher.

13.5 Summary

In this chapter we have presented an evolutionary approach to the electric power distribution network planning and operation problems. The problems have been formulated as large-scale optimization problems and addressed by especially designed evolutionary hybrids. The designed evolutionary operators and hybridization process have been presented and their role in optimality discussed. Application examples have been provided to support the discussion and illustrate critical implementation practicalities.

References

1. Carvalho, P.M.S., Ferreira, L.A.F.M., Barruncho, L.M.F.: Optimization Approach to Dynamic Restoration of Distribution Systems. International Journal of Electrical Power & Energy Systems 29(3), 222–229 (2007)
2. Carvalho, P.M.S., Ferreira, L.A.F.M.: Distribution Quality of Service and Reliability Optimal Design: Individual Standards and Regulation Effectiveness. IEEE Transactions on Power Systems 20(4) (November 2005)
3. Carvalho, P.M.S., Ferreira, L.A.F.M.: Urban Distribution Network Investment Criteria for Reliability Adequacy. IEEE Transactions on Power Systems 19(2) (May 2004)
4. Carvalho, P.M.S., Ferreira, L.A.F.M.: On the Robust Application of Loop Optimization Heuristics in Distribution Operations Planning. IEEE Transactions on Power Systems 17(4), 1245–1249 (2002)
5. Carvalho, P.M.S., Ferreira, L.A.F.M., Barruncho, L.M.F.: On Spanning-Tree Recombination in Evolutionary Large-Scale Network Problems: Application to Electrical Distribution Planning. IEEE Transactions on Evolutionary Computation 5(6), 613–630 (2001)
6. Carvalho, P.M.S., Ferreira, L.A.F.M., Lobo, F.G., Barruncho, L.M.F.: Optimal Distribution Network Expansion Planning Under Uncertainty by Evolutionary Decision Convergence. International Journal of Electrical Power & Energy Systems – Special Issue on PSCC 1996 20(2), 125–129 (1998)
7. Ferreira, L.A.F.M., Carvalho, P.M.S., Barruncho, L.M.F.: An Evolutionary Approach to Decision-Making in Distribution Systems. In: Proceedings of the 14th International Conference and Exhibition on Electricity Distribution (CIRED 1997), Birmingham, UK (1997)
8. Carvalho, P.M.S., Ferreira, L.A.F.M., Barruncho, L.M.F.: Hybrid Evolutionary Approach to the Distribution Minimum-Loss Network Configuration. In: Proceedings of the Simulated Evolution And Learning (SEAL 1998 Special Session), Canberra, Australia (1998)
9. Ferreira, L.A.F.M., Carvalho, P.M.S., Jorge, L.A., Grave, S.N.C., Barruncho, L.M.F.: Optimal Distribution Planning by Evolutionary Computation – How to Make it Work. In: Proceedings of the Transmission and Distribution Conference and Exposition (TDCE 2001), Atlanta, USA (2001)
10. Mira, F., Jorge, L.A., Quaresma, E., Ferreira, L.A.F.M., Carvalho, P.M.S.: New Technologies for Distribution Planning: Optimal Design for Efficiency, Reliability and New Regulation Criteria. In: Proceedings of XI ERIAC CIGRÉ, Paraguay (2005)
11. Carvalho, P.M.S., Ferreira, L.A.F.M., Rojão, T.L.: Dynamic Programming for Optimal Sequencing of Operations in Distribution Networks. In: Proc. 15th Power System Comp. Conf. (PSCC 2005), Liège, Belgium (2005)

12. Carvalho, P.M.S., Carvalho, F.J.D., Ferreira, L.A.F.M.: Dynamic Restoration of Large-Scale Distribution Network Contingencies: Crew Dispatch Assessment. In: IEEE Power Tech. 2007, Lausanne, Switzerland (2007)
13. Carvalho, P.M.S., Ferreira, L.A.F.M., Lobo, F.G., Barruncho, L.M.F.: Distribution Network Expansion Planning Under Uncertainty: A Hedging Algorithm in an Evolutionary Approach. IEEE Transactions on Power Delivery 15(1), 412–416 (2000)
14. Michalewicz, Z.: Genetic algorithm + Data structures = Evolution programs, 3rd edn. Springer, New York (1996)
15. Behzad, M., Chartrand, G.: Introduction to the Theory of Graphs. Allyn and Bacon Inc., Boston (1971)
16. Walters, G.A., Smith, D.K.: Evolutionary Design Algorithm for Optimal Layout of Tree Solutions. Eng. Optimization 24(4), 261–281 (1995)
17. Dengiz, B., Altiparmak, F., Smith, A.E.: Local Search Genetic Algorithm for Optimal Design of Reliable Networks. IEEE Trans. Evolutionary Computation 1(3), 179–188 (1997)
18. Raidl, G.R., Julstrom, B.A.: Edge Sets: An Effective Evolutionary Coding of Spanning Trees. IEEE Transactions on Evolutionary Computation 7(3), 225–239 (2003)
19. Goldberg, D.E., Voessner, S.: Optimizing global-local search hybrids, IlliGAL Report No. 99001 (January 1999)
20. Civanlar, S., Grainger, J.J., Yin, H., Lee, S.S.: Distribution Feeder Reconfiguration for Loss Reduction. IEEE Trans. Power Delivery 4(2), 1217–1223 (1988)
21. Goldberg, D.E., Deb, K., Clark, J.H.: Genetic Algorithms, Noise and the Sizing of Populations. Complex Systems 6, 333–362 (1992)

Chapter 14
A Parallel Hybrid Implementation Using Genetic Algorithms, GRASP and Reinforcement Learning for the Salesman Traveling Problem

João Paulo Queiroz dos Santos, Francisco Chagas de Lima Júnior,
Rafael Marrocos Magalhães, Jorge Dantas de Melo, and Adrião Duarte Doria Neto

Abstract. Many problems formerly considered intractable have been satisfactorily resolved using approximate optimization methods called metaheuristics. These methods use a non-deterministic approach that finds good solutions, despite not ensuring the determination of the overall optimum. The success of a metaheuristic is conditioned on its capacity of alternating properly between the exploration and exploitation of solution spaces. During the process of searching for better solutions, a metaheuristic can be guided to regions of promising solutions using the acquisition of information on the problem under study. In this study this is done through the use of reinforcement learning. The performance of a metaheuristic can also be improved using multiple search trajectories, which act competitively and/or cooperatively. This can be accomplished using parallel processing. Thus, in this paper we propose a hybrid parallel implementation for the GRASP metaheuristics and the genetic al gorithm, using reinforcement learning, applied to the symmetric traveling salesman problem.

14.1 Introduction

Modeling and resolving complex problems in the world we live in is not an easy task, given that there are some situations in which it is impossible to build a detailed

João Paulo Queiroz dos Santos · Jorge Dantas de Melo ·
Adrião Duarte Doria Neto
Department of Automation and Control, Federal University of Rio Grande do Norte
e-mail: {jxpx,jdmelo,adriao}@dca.ufrn.br

Rafael Marrocos Magalhães
Department of Exact Sciences, Federal University of Paraiba
e-mail: rafael@ccae.ufpb.br

Francisco Chagas de Lima Júnior
Department of Computing
State University of Rio Grande do Norte,
College of Science and Technology Mater Christi
e-mail: lima@dca.ufrn.br

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 345–369.
springerlink.com

model for the problem, owing to its high complexity. On the other hand, a process of simplifying this model leads to loss of relevant information that may compromise its quality. In addition to the inherent difficulty in building models for these problems, a characteristic during the resolution phase is the need for large scale computational processing, which, in most cases, leads to these problems being considered intractable. In this context researchers have dedicated themselves to the development of techniques aimed at facilitating modeling and, mainly, at resolving these problems [13], [11] and [10].

A widely used approach for solving intractable problems has been the usage of so-called metaheuristics, which are strategies based on heuristic procedures, mainly applicable to optimization problems and which produce a simplified process of a stochastic search in the solution space [12]. Despite achieving good results without an exhaustive search, metaheuristics do not ensure obtaining the optimal solution of the problem.

The great challenge of a metaheuristic is to maintain the equilibrium between exploration and exploitation processes. Exploration (or diversification) is used to allow the solution to escape from the so-called local minima, whereas exploitation (or intensification) is used to improve the quality of the solution locally, in search of the overall optimum.

Resolving the dilemma of when to "explore" and when to "exploit" is not an easy task. Thus, many researchers have been involved in seeking improvements that help the metaheuristics in the exploration and/or exploitation process. In this context a very interesting study [7] was conducted using reinforcement learning, but specifically the *Q-learning* algorithm, as an exploration/exploitation strategy for GRASP metaheuristics and the genetic algorithm, applied to the traveling salesman problem - TSP. In addition to the "explore or exploit" dilemma, another aspect to consider is the large number of possible solutions that problems such as TSP present.

This high dimension of the universe of solutions of problems like TSP generates a large processing demand, which may be met by the use of architectures with parallel processing capacity, able to increase, by some orders of magnitude, the processing power available in monoprocessed architectures.

The use of parallel processing promotes the development of new algorithms and opens possibilities for the exploration of aspects of the problem not approached in the usual architectures such as competition and cooperation [5].

Based on the success obtained by the aforementioned techniques and motivated by the difficulties of complex problems in the real world, this study proposes the development of hybrid parallel methods, using reinforcement learning, GRASP metaheuristic and genetic algorithms.

With the use of these techniques together, better efficiency in obtaining solutions is expected. In this case, instead of using the *Q-learning* algorithm of reinforcement learning only as a technique to generate the initial metaheuristic solution, we intend to use it cooperatively/competitively with the other strategies in a parallel implementation, which will be described in detail in the continuation of the text.

14.2 Theoretical Foundation

A brief theoretical foundation, required for a better understanding of the rest of the text, will be presented in this section.

14.2.1 GRASP Metaheuristic

The Greedy Randomized Adaptive Search Procedure (GRASP) metaheuristic [4] is a multi-start iterative process, in which each iteration consists of two phases: construction and local search. The construction phase generates a viable solution, whose neighborhood will be investigated until a local minimum is found during the local search phase. The best solution among these is used as the result.

In each iteration of the constructive phase, the set of candidate elements is formed by all the elements that can be incorporated into the partial solution under construction without compromising viability.

The selection of the following element to be incorporated is determined by the assessment of all the candidate elements, according to a function of greedy assessment. This greedy function generally represents the incremental increase in the cost function, owing to the incorporation of this element in the solution under construction.

The assessment of the elements for this function leads to the creation of a restricted candidate list (*RCL*) formed by the best elements, that is, those whose incorporation into the current partial solution results in lower incremental costs (this is the greedy aspect of the algorithm). Once the selected element is incorporated into the partial solution, the candidate list is updated and the incremental costs are reassessed (this is the adaptable aspect of the heuristic).

The probabilistic aspect of GRASP is due to the fact of randomly choosing one of the *RCL* elements and not necessarily the best, except when the *RCL* is of unitary size, where the selection criterion is reduced to the greedy option.

The improvement phase consists typically of a local search procedure aimed at enhancing the solution obtained in the construction phase, given that the construction phase solution may not represent an overall optimum. In GRASP metaheuristic it is always beneficial to use a local search to improve the solutions obtained in the constructive phase. Additional information on the GRASP metaheuristic can be found in [4], [8].

14.2.2 Genetic Algorithm

Genetic Algorithms (GA) are based on a biological metaphor. They ”visualize” the resolution of a problem as competition between a population of candidate solutions that evolve. A function of fitness assesses each solution and decides if it will contribute to the evolution of the population in the next generation. Thus, using operators analogous to the genetic transfer in sexual reproduction, the algorithm creates a new generation of candidate solutions.

At the start of the execution of a genetic algorithm, a population of chromosomes is generated. Each of these chromosomes, when decoded, will represent a different solution to the problem. Considering that there are N chromosomes in the initial population, the following steps will be repeated until a stop criterion is reached:

1. Assess each chromosome to determine how good it is in resolving the problem, associating a score to each chromosome according to the function of "fitness".
2. Select two members of the current population. The selection probability must be proportional to the fitness function value.
3. Depending on the crossover rate, cross the genes of the selected chromosomes at a randomly chosen point.
4. Depending on the mutation rate, exchange the genes of one selected chromosome. Repeat steps 2, 3 and 4, until a new population of N chromosomes has been generated.

Genetic algorithms are very efficient in the search of optimal or nearly optimal solutions, in a wide variety of problems, such as: network project optimization [1], vehicle routing [15], timetable problem [3], task grouping [16], among others. Additional information on genetic algorithms can be found in [2] and [9].

14.2.3 Reinforcement Learning: Q-Learning Algorithm

Not all reinforcement learning algorithms need a complete modeling of the environment; that is, it is not necessary to know the matrix of transition probabilities or the expected values of the reinforcement signal for all the possible states actions in the environment. This is the case, among others, for reinforcement learning techniques based on temporal differences [14].

One of these techniques is the *Q-learning* algorithm (Watkins, 1989), which is considered one of the most important contributions in reinforcement learning, given that its convergence to optimal Q values does not depend on the policy that is being used. The updated expression of the Q value in the *Q-learning* algorithm is the following:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a \in A} Q(s',a)] \tag{14.1}$$

where s_t is the current state, at is the action performed in the s_t state, r_t is the reinforcement signal received after executing at in s_t, s_{t+1} is the next state, γ is the discount factor $(0 \leq \gamma < 1)$ and α $(0 < \alpha < 1)$ is the learning coefficient. The function $Q(s,a)$ is the value associated to the state-action pair (s,a) and represents how good the choice of this action is in minimizing the accumulated reward function, designated by:

$$R = \sum_{k=0}^{n} \gamma_{t+k+1}^{k} \tag{14.2}$$

One important characteristic of this algorithm is that the choice of actions to be executed during the process of iterative approximation of the Q function can be made using any exploration/exploitation criterion, including in a random manner. A

widely used technique for this choice is the so-called ε-greedy exploration, which consists of choosing the action associated to the highest Q-value with probability $1-\varepsilon+\varepsilon|A(s)|$, where $|A(s)|$ corresponds to the number of possible actions to be executed starting from s. *Q-learning* was the first reinforcement learning method to display strong evidence of convergence. Watkins [17] showed that if each pair (s,a) is visited an infinite number of times, the Q-value function $Q(s,a)$ will converge with probability one for $Q*$, with α sufficiently small. As long as the optimal Q-value is known, an optimal choice of actions can be made according to the expression:

$$a^*(s) = \max_a Q(s,a) \tag{14.3}$$

The pseudocode for the *Q-learning* algorithm is:

Q-learning Procedure(r,α, ε, γ)
Initialize $Q(s,a)$
Repeat for each episode
Initialize s
Repeat for each state of the episode
Select a in accordance with the rule ε-greedy
Observe the values of r and s'
$Q(s,a) \leftarrow Q(s,a)+\alpha[r+\gamma\max_a Q(s',a)-Q(s,a)]$
$s \leftarrow s'$
Until a final state is reached
Until the convergence is gotten
End-Q-Learning.

The convergence criteria for the *Q-learning* algorithm are:

1. The model of the system is a deterministic MDP.
2. The immediate reward values are bounded by some constant.
3. The agent visits every possible state-action pair infinitely often.

Additional information on the *Q-learning* algorithm can be found in [17]

14.3 Hybrid Methods Using Metaheuristic and Reinforcement Learning

The hybrid parallel implementation proposed in this paper is based on work originally published by [7]. In the paper the authors propose the use of *Q-learning* algorithm as a intelligent strategy for exploitation and/or exploitation for GRASP metaheuristics and Genetic Algorithm. For better understanding of the hybrid parallel method proposed here, the Sec. 14.3.1 and 14.3.2 describe of brief way, the sequential implementations proposals by previously cited authors.

14.3.1 GRASP-Learning

The GRASP metaheuristic need to work with good initial solution. Considering this dependence, the use of the *Q-learning* algorithm is here proposed as a constructor of initial solutions, in substitution to the partially greedy algorithm generally used.

The *Q-learning* algorithm use as rule of state transition will use $\varepsilon - greedy$ strategy, mentioned previously, defined by:

$$\pi(s') = \begin{cases} a_{\text{random}} \text{ if } v < \varepsilon \\ \text{argmax}_{a'} Q(s', a') \text{ otherwise} \end{cases} \tag{14.4}$$

where: v is a random value with uniform probability distribution between $[0, 1]$, ε $(0 \leq \varepsilon \leq 1)$ is the parameter that defines the exploration rate so that, the lesser the value of ε, the lesser the probability of making a random choice of the action will be, and a_{random} is an action randomly chosen amongst the possible actions to be executed in the state s'.

As already mentioned, the *Q-learning* algorithm will be used as a constructor of initial solutions for GRASP metaheuristics. Therefore, each iteration of the algorithm intends to construct a solution of good quality, since the *Q-learning* will explore the knowledge of the environment (solution space of the problem) through the use of the matrix of rewards. The matrix of rewards is generating using the matrix of distance of each instance do TSP, and is computed of the following form:

$$r(s', a') = \frac{M_i}{d_{ij}} \tag{14.5}$$

where, d_{ij} corresponds to the distance between cities i and j that compose a route and are represented in the model by the states s and s', respectively, while M_i is the distance average of the city i for all another cities.

The control between "exploitation" and "exploration" will be made by the parameter of the transition rule described in (14.4). Higher the value of ε, more rarely the *Q-learning* algorithm will make use of the knowledge of the environment, while, lower value of ε, means more random choice of actions.

The basic idea of the *GRASP-Learning* method is to make use of the information contained in the matrix of Q-values as a kind of adaptive memory, that allows repeat the good decisions made in previous iterations, and avoid those that were not interesting. Thus, considering for example the traveling salesman problem, the method used in each GRASP iteration, the state-action pairs $Q(s, a)$ stored in the matrix of the Q-values, to decide which visits are promising for the traveling salesman.

The policy $\varepsilon-$ greedy is used with the objective guarantee certain level of randomness, thus avoiding the construction of locally optimal solutions. The Fig. 14.1 presents an overview of the *GRASP-Learning* metaheuristic.

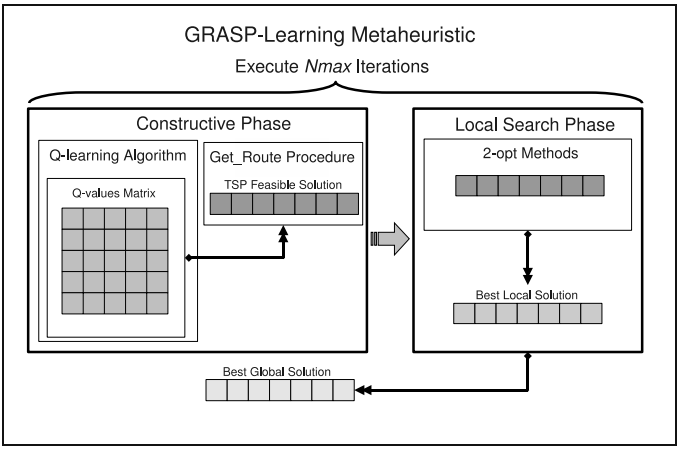


Fig. 14.1 Framework of GRASP-Learning Metaheuristic

14.3.2 Genetic-Learning

The Genetic-Learning algorithm use the idea of introducing knowledge of the environment through the Reinforcement Learning. The main focus of this method is to explore of efficient way the space of search through the learning of the environment of the problem, using the *Q-learning* algorithm with a genetic algorithm - GA. Making use of a genetic algorithm, the solution search space of a problem can be explored adequately through the generation of an initial population of high fitness in

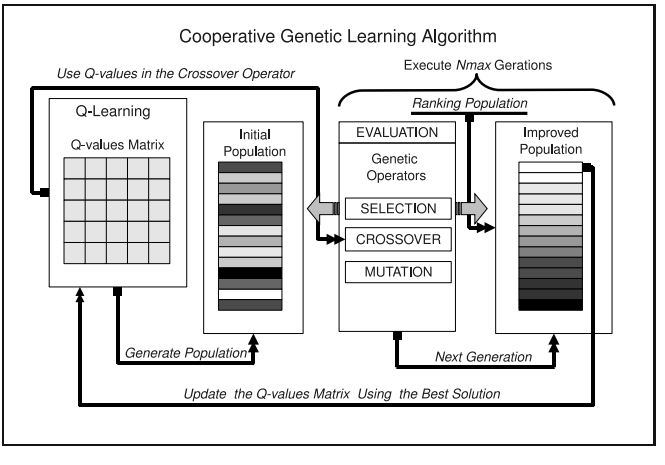


Fig. 14.2 Framework of Cooperative Genetic-Learning Algorithm

relation to the objective function. Therefore, the genetic algorithm considered here has its generated initial population through the *Q-learning* algorithm.

Another modification in this method occurs in the crossover operator of the GA. In this operator, one of the parents will be taken from the improved population for the action of the operators of the current generation, as in the traditional GA, while the other will be generated by the *Q-learning* algorithm without any action from the genetic operators. The other operators (selection and mutation) are implemented in the traditional way. The Fig. 14.2 presents an overview of the Cooptative *Genetic-Learning* algorithm.

The following Section describe the parallel hybrid implementation of the *GRASP-Learning* and *Genetic-Learning* metaheuristics.

14.4 Parallel Hybrid Implementation Proposed

This study proposes the development and implementation of a cooperative and/or competitive parallel strategy of *Q-learning*, genetic and GRASP algorithms, to resolve the traveling salesman problem (TSP). The basic idea is to collectively use the solutions found for each algorithm and with this make up the deficiencies they exhibit when used separately. For this interaction to be effective, one must establish the communication interfaces that enable the exchange of information. Similarly, the existence of multiple processing elements will allow different algorithm parameterizations to be used, as described below.

14.4.1 Methodology

As explained in section 14.2, the genetic algorithms work with populations of solutions, GRASP provides a local optimal solution and *Q-learning* a table of Q-values that enables the building of solutions starting from any point on the table. All the algorithms involved are iterative; that is, the quality of their solutions tends to improve with an increase in the number of iterations.

For a better understanding of the proposal, consider the schematic diagram in Figure 14.3, which shows the interaction structure between the algorithms. In this scheme the critical has the task of managing the quality of solutions generated by each of the algorithms, i.e., when necessary it replace a bad solution by another best.

14.4.1.1 Communication Sent by Q-Learning Algorithm

Using the table of Q-values, one or more solutions are generated as follows: Choose a starting city s_0 for TSP and refer to the table of values $Q(s,a)$, obtaining the best value $Q*(s_0,a) = \max_a Q(s_0,a), \forall a \in A(s_0)$, where $A(s_0)$ are all the possible actions from the s_0 state.

Choose the city indicated by the choice of action a as being the next in the TSP route. Repeat the process until all the cities have been visited a single time,

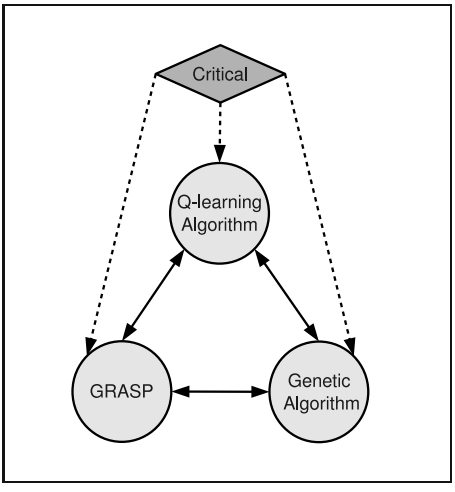


Fig. 14.3 Cooperation scheme between the *Q-learning*, GRASP and Genetic algorithms

generating thus a solution S_q for TSP. For simplicity, consider that only one solution is generated.

For GRASP, solution S_q will be used as the initial solution instead of that obtained in the construction phase and will be improved using local search in a new iteration of the algorithm.

14.4.1.2 Communication Sent by Genetic Algorithm

At each iteration of the algorithm, choose the individual of the population with the highest suitability value (best fitness); this individual will be a solution S_g. Let R_g be the cost of the route associated to S_g.

For GRASP, solution S_g will be used as the initial solution, substituting the solution that would be generated in the construction phase, in the same way that this solution will be improved before using local search in the next iteration of the algorithm.

For *Q-learning*, the Q-values table must be updated based on information available in solution S_g. Therefore, it should be remembered that a value $Q(s,a)$ represents an estimate of the total expected return that will be obtained when in state s and choosing action a. In the case of TSP this value represents a cost estimate of the cycle starting from s and having as the next city visited the one indicated by action a. Similarly, solution S_g has a cost associated to the cycle, that is, R_g. Thus, a series of pairs (s, a) can be established from this solution, corresponding to the order in which the cities were visited and the values $Q(s,a)$ can be updated as follows:

$$Q(s,a) = \beta.(Q(s,a) + R_g), \quad (0 < \beta < 1) \tag{14.6}$$

14.4.1.3 Communication Sent by GRASP

At each iteration of the algorithm, take the solution obtained using the GRASP local search, S_G. Let R_G be the cycle cost associated to S_G.

For *Q-learning*, use the same procedure described in the previous item and represented by equation 14.6, replacing R_g by R_G.

It is important to observe that in a parallel application, each of the algorithms will constitute an independent task that will be executed at its own pace. This means that communication between tasks cannot be synchronous; otherwise the execution of one of the tasks will be blocked while it waits for results from another to be sent. Therefore, asynchronous communications could be avoided with a specific parameters settings. It is done in a fashion that each algorithm execute a number of steps that is approximately proportional of others algorithms. For example, if the GRASP algorithm run ten times faster than genetic algorithm for one step, so the number of steps for that algorithm will be adjusted proportionally greater than another. This solution reduce problems with asynchronous communications and avoid idleness time of tasks.

Since there are three algorithms involved in the parallel implementation, it can be supposed that three distinct tasks are needed for such a purpose. If we consider that a parallel architecture has considerably more than 3 processing elements, it follows that the efficiency of the implementation will be compromised, given that several of these elements will be idle during the execution of the application.

To avoid this idleness and to make use of the entire potential of parallel architecture, multiple parameterizations of the algorithms will be used in this study. We understand multiple parameterizations as being the execution of a same algorithm with different behaviors.

In the case of genetic algorithms, this corresponds to associating to some parallel tasks, instances of genetic algorithms with different population behaviors; that is, different mutation rates, crossover and selection mechanisms.

For GRASP, different lengths can be used for the restricted list of candidates and different local search mechanisms.

In the case of *Q-Learning*, the possibilities of multiple parameterizations are associated to the choices of the parameters involved in updating Q-values. These values are the α and the γ presented in equation 14.1.

It should be pointed out that the occurrence of multiple parameterizations, in addition to allowing a more efficient use of parallel architecture, will make the communication structure between the tasks more complex, given that two different instances of a same algorithm will be able to exchange information. The cost of these communications, in terms of processing time, will be analyzed during the implementation to avoid compromising performance.

14.5 Experimental Results

In this section all the experiments done in this work are explained and their results are presented and compared. The proposed methods in the Sec. 14.4 use the

Q-learning algorithm, GRASP metaheuristic and Genetic Algorithm in a parallel hybrid implementation. Therefore, to evaluate the proposed method the well known traveling salesman problem was utilized. To use the proposed method in resolution of the TSP is necessary an adequate modeling of the problem. To do that, the section 14.5.1 presents the TSP modeled as a Reinforcement Learning problem.

The section 14.5.2 shows how the computational tests were conducted, the first sub section introduces the utilized methodology. The sub sections 14.5.2.2 to 14.5.2.5 explain the four different setups developed and their individual results, namely Serial execution, Parallel execution, Parallel limited execution, and Parallel Group execution respectively. Therefore, to conclude the section, in sub section 14.5.2.6 was conducted the performance analysis and in 14.5.2.7 was done a collective analysis of all the experiments.

14.5.1 The Traveling Salesman Problem

The traveling salesman problem (TSP) is a classical problem of combinatorial optimization that consists of determining a minimum-cost Hamiltonian cycle on a weighted graph. The TSP is classified as NP-complete [6], which means that resolving this problem by examining all the possible solutions is computationally impracticable.

The problem can be formally defined as follows: Consider a set V of vertices, representing cities, and a set A of arches totally connecting the V vertices. Let d_{ij} be the length of the arch $(i, j) \in A$, which is, the distance between cities i and j, with $i, j \in V$. Resolving TSP is finding a minimum-length Hamiltonian circuit on graph $G(V,A)$, a Hamiltonian circuit being a closed pathway visiting once and only once all the $n = |V|$ vertices of G, and the circuit length is given by the sum of the length of all the arches that it is made up of. In this work the TSP will be modeled as a reinforcement learning task.

In this work the environment for the traveling salesman problem can be seen as a system whose dynamics can be represented by a Markovian decision process of finite horizon time, characterized as follows:

- The environment evolves probabilistically occupying a finite set of discrete states. This evolution is episodic where, to each episode, one of the states of the system is chosen as the final one, i.e., the final state is not known a priori. The state of the system is completely observable and the transition probabilities, in each discrete time t, can assume two distinct values:

$$Pr = \{s_{t+1} = j | s_t = i, a_t = a\} = p_{ij}(a) = \begin{cases} 0 \\ 1 \end{cases} \tag{14.7}$$

  It must be observed that, if $P_{ij}(a) = 0$ then $j = i$;
- For each state of the environment there is a finite set of possible actions that can be carried through. This set of actions is invariant in time, being represented by $A(s_t, i) = A(i)$;
- Every time the agent carries through an action, it causes certain rewards that can assume two distinct values in each discrete time:

$$r_{ij}(a) = \begin{cases} -\gamma \max_{a'} Q(i,a') + Q(i,a) \text{ if } p_{ij} = 0 \\ r_{ij} \text{ if } p_{ij}(a) = 1 \end{cases} \tag{14.8}$$

As previously established, if $P_{ij}(a) = 0$ then $j = i$. Soon, the $Q(i,a)$ associated to state i and the action a must not be updated, which justifies the choice of expression for rewards $r_{ij}(a)$ in (14.8) when $P_{ij}(a) = 0$. The observation of the states, the accomplishment of the actions and the incidence of the reward occur in discrete time. Considering the description of the environment for the *TSP* made here, the modeling process as a reinforcement learning problem can be made in the following way:

- States: $S = \{s \in N | N$ is the set of the cities that compose a route for the TSP$\}$;
- Actions: $A = \{a \in A(i) | A(i)$ is the set of all possible options of cities to be added in the route, from a city $i\}$,
- Rewards: $r(s,a)$ is the expected return in state s to decide for the action a.

In practice, the rewards can be seen as the prize (or save) of choosing the shortest distance of the current city in the route, and can also be trivially seen by Equation 14.5.

14.5.2 Computational Test

This section evaluates the potential use of parallel processing approach for the cooperative and competitive algorithms for integration of (GRASP, and Genetic *Q-learning*), aimed to observe the best type of parallel configuration for the three algorithms involved. With these goals, a programming environment has been developed to allows the parallel evaluation of the performance in terms of speedup and efficiency parameters, and also the quality of results.

14.5.2.1 Methodology

The computer architecture was used as a dedicated network of computers that form a *cluster* system of *Beowulf* type. This architecture is composed of nine computers that have the following configuration: Intel Core 2 Duo processor with 2.33GHz of clock, 2GB of RAM, 80GB hard disk and Gigabit Ethernet network card. The operating system used on the machines is GNU/Linux Ubuntu version 8.04. The library for parallel programming used was OpenMPI version 1.2.5, that is an implementation of a message passing interface library. The MPI is based on simulating of the existence of many different programs exchange information simultaneously. The Zabbix software version 1.4.2 was used as monitoring performance. The implementation of the algorithms were developed with the programming language C++ and analysis using MATLAB.

For the Traveling Salesman Problem (TSP) was used the TSPLIB[1] repository, this library presents various instances for many different variants of TSP. Eight TSP

[1] The instances of the TSP problem used were obtained from *TSPLIB site*, hosted under the domain http://ftp.zib.de/pub/Packages/mp-testdata/tsp/tsplib/tsplib.html

instances were evaluated in this parallel implementation, they are: gr17, bays29, gr48, berlin52, eil76 and a280. The information about the instances used in the experiment are presented in Table 14.1.

Table 14.1 Information about the TSPLIB instances

Instance Name	Description	Best value
gr17	City problem (Groetschel)	2085.00
bays29	Bavaria (street distance)	2020.00
gr48	City problem (Groetschel)	5046.00
berlin52	Locations in Berlin (Germany)	7542.00
eil76	city problem (Christofides/Eilon)	538.00
a280	Drilling problem (Ludwig)	2579.00

14.5.2.2 Serial Execution

This experiment makes the implementation of the serialized algorithm, where all algorithms were implemented on the same machine, that is in the same processing node. Table 14.2 presents the results of serial execution in each of eight instances evaluated. This experiment was conducted comparing purposes with the parallel implementation and measurement of speedup gain and efficiency in terms of time and quality of results.

Table 14.2 Serial execution time and Objective Function values for each instance

Instance	bays29	swiss42	berlin52	eil76	gr120	ch150	si175	a280
Objective Function	2020	1284	7705	591	8789	9163	23795	4682
Time in seconds	220	577	952	1390	3308	7082	9975	20300

14.5.2.3 Parallel Execution

The parallel implementation has been done distributing the algorithm parts over the nodes of the cluster. One node with genetic algorithm, other with GRASP algorithm, and another focused on the reinforcement learning. The communication between algorithms nodes was done as showed in Figure 14.3. For each evaluated instance was generated thirty executions for a statistical analysis of implementation.

Table 14.3 shows the parameters used for implementation of algorithms for this set of experiments. The number of individuals in the population for genetic algorithm was one hundred (100) elements, the *crossover* rate equals to 0.7, the mutation rate equal to 0.2, while for *Q-learning* algorithm the parameters were adjusted as follows, $\alpha = 0.8$, $\varepsilon = 0.01$, $\gamma = 1$ and $\beta = 0.2$, which is the actualization parameter

of *Q*-values with solutions from the GRASP and Genetic Algorithm. Considering the fact that the execution time of Genetic Algorithm (GA) is slower than others algorithms, the number of executions of the GRASP and the *Q-Learning* are higher than the Genetic Algorithm, this quantity is expressed in columns (QL/GA) and (GRASP/GA) mean that the number of executions of the algorithm *Q-Learning* per Genetic Algorithm iteration and the GRASP execution per GA iteration. The Rand index means an exchan ge of position in the current solution in order to avoid a local minimum or to diversify the current solution, the Communication index is the number of iterations that the algorithms exchange information.

Table 14.3 Parameters for algorithms executions

Instance	iterations	(QL/GA)	(GRASP/GA)	Rand	Communications
bays29	20.00	8.00	28.00	5.00	10.00
swiss42	40.00	7.00	18.00	5.00	20.00
berlin52	50.00	7.00	18.00	5.00	20.00
eil76	50.00	8.00	10.00	5.00	20.00
gr120	70.00	8.00	7.00	5.00	30.00
ch150	100.00	10.00	6.00	6.00	50.00
si175	100.00	12.00	5.00	5.00	50.00
a280	100.00	12.00	5.00	6.00	50.00

The graphs shown in Figures from 14.4 to 14.11 presents for each instance the behavior of the value of objective function achieved by each test in comparison with the best known normalized value of the objective function, in addition are plotted the tracks of superior and inferior standard deviation for a better visual perception of the quality of solution. The limits of the plot area were chosen in a range of 20% around the value of the average objective function in each experiment.

Figure 14.4 shows homogeneous result in all executions, in this case the objective function reached the optimal value known in thirty executions. Figures 14.5 to 14.10 have a very stable behavior in all executions, it can be confirmed by the value of standard deviations that in all case were less than 2%. The less homogeneous behavior and more proportional variance was in Figure 14.11, this is probably caused by the complexity of this instance and the parameters of execution that was selected empirically for this implementation.

Table 14.4 presents a compilation of statistical data for this experiments. The first line shows the average values for the objective function obtained with thirty runs in each instance studied. The second line shows the standard deviation of each instance in relation to mean value of objective function obtained, the third line presents the same information about standard deviation but in percentage value for better understanding of the data. The fourth line shows the optimal value (better value) of the objective function found in literature and TSPLIB database for each instance.

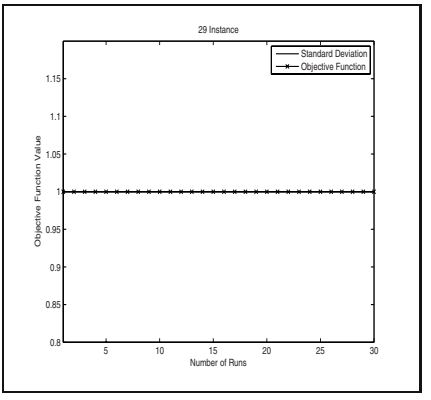


Fig. 14.4 Standard Deviation of bays29 test

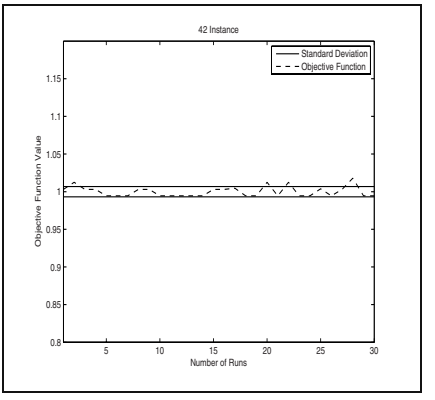


Fig. 14.5 Standard Deviation of swiss42 test

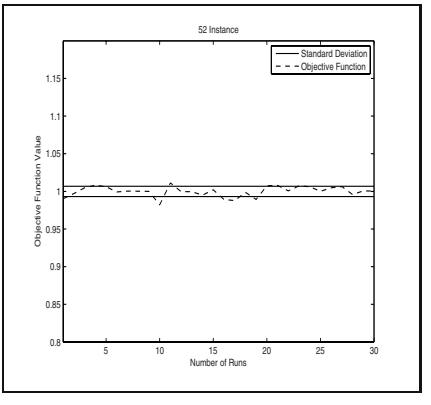


Fig. 14.6 Standard Deviation of berlin52 test

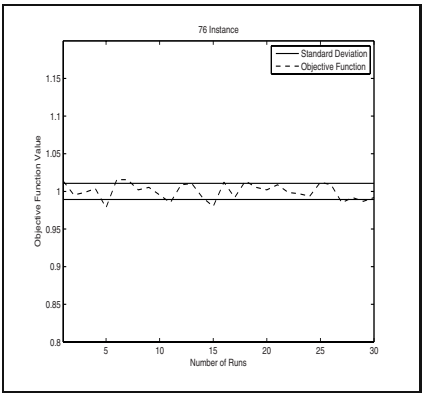


Fig. 14.7 Standard Deviation of eil76 test

The fifth line shows the distance between the mean value and the known optimum values, which is shown again in the last line in percentages.

It is possible to interpret from the last row of table 14.4 that result from the execution of the bayes29 instance in average 100% near (in a proximity way) of the best objective function value known, for instance 42 is 99.45% near of better value known and so on, as shown in the graph of the Figure 14.12 and Table 14.5, where the values closer to 100% represent better solutions. The average distance of maximum and minimum proximity are obtained through the standard deviation percentage value shown in Table 14.4, which indicates in average, more than half of the instances have an proximity of objective function a value greater than 90% closer to the optimal function value.

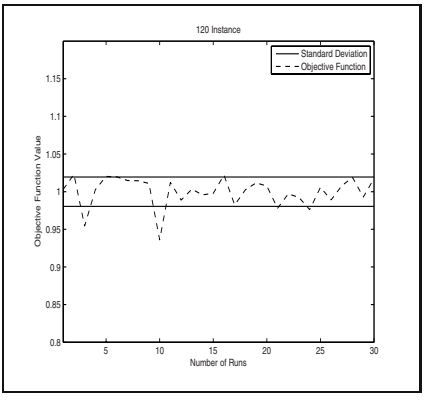


Fig. 14.8 Standard Deviation of gr120 test

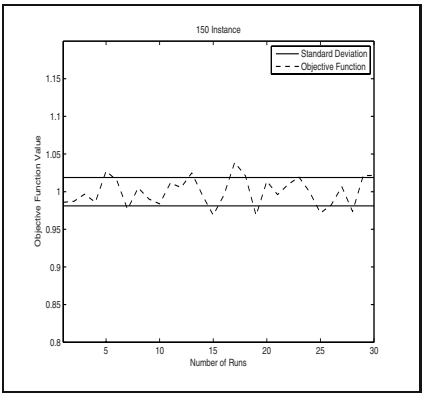


Fig. 14.9 Standard Deviation of ch150 test

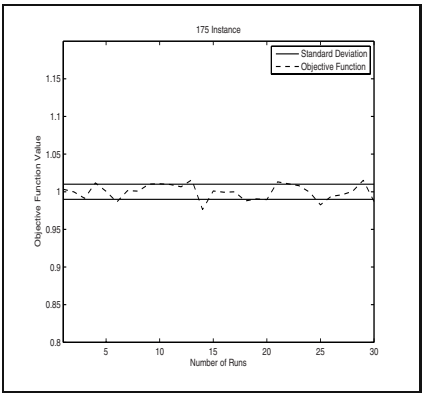


Fig. 14.10 Standard Deviation of si175 test

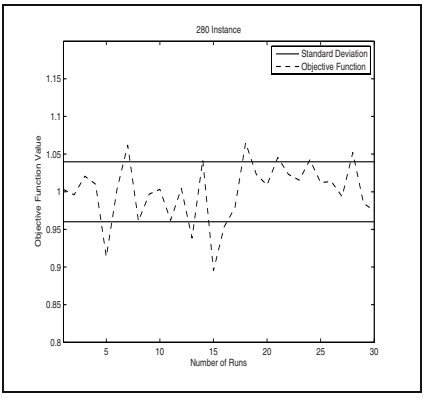


Fig. 14.11 Standard Deviation of a280 test

Table 14.4 Statistical data about parallel experiments

Instances	bays29	swiss42	berlin52	eil76	gr120	ch150	si175	a280
O.F. Mean Value $^{O.F.}$	2020.00	1280.00	7678.70	590.80	8638.50	8723.10	23936.90	4608.70
Standard Deviation	0.00	8.70	52.10	6.30	168.60	164.70	240.60	183.40
Standard Deviation (%)	0.68	0.00	0.67	1.07	1.95	1.88	1.00	3.90
Optimal O.F.	2020.00	1273.00	7542.00	538.00	6942.00	6528.00	21407.00	2579.00
O.F. Mean Dist	0.00	7.10	136.70	52.80	1696.5	2195.17	2529.90	2029.70
O.F. Mean Dist. (%)	0.00	0.50	1.80	9.80	24.40	33.60	11.80	78.70

$^{O.F.}$ Objective Function.

Table 14.5 Mean percentage distance of best objective value evaluated

Instance	mean proximity(%)	max prox. value(%)	min prox. value(%)
bays29	100.00	100.00	100.00
swiss42	99.44	100.00	98.77
berlin52	98.19	98.87	97.51
eil76	90.19	91.26	97.51
gr120	75.56	77.51	73.62
ch150	66.37	68.26	64.48
si175	88.18	89.19	87.18
a280	21.30	25.28	17.32

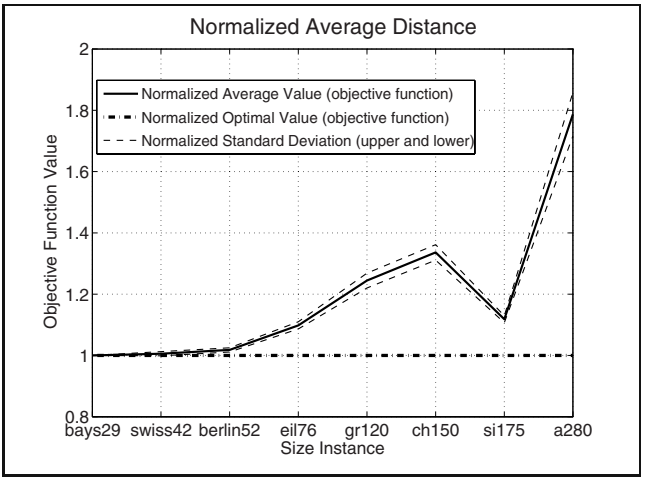


Fig. 14.12 Mean percentage distance of optimal objective function value for each instance

14.5.2.4 Parallel Limited Time Execution

In this experiment the procedure adopted for implementation of algorithms in the *cluster* was the same as the parallel experiment previously described, the difference was in limit the execution time that was chosen as half of the average time spent with the standard parallel implementation, the algorithm was stopped independent of the iteration it has achieved. To obtain statistical data each instance was run thirty (30) times.

Table 14.6 shows the values obtained with the experiments. Comparing them with the results of execution without the limitation of time, it is observed that the results although they are lower in quality, are very close to the values found in experiment. This behavior was expected because the time to search for the minimum was reduced.

Table 14.6 Statistical data about limited time parallel experiments

Instances	bays29	swiss42	berlin52	eil76	gr120	ch150	si175	a280
Mean O.F. $^{O.F.}$	2020.00	1273.00	7695.90	589.40	8560.00	8674.30	23789.00	4519.00
Standard Deviation	0.00	11.00	92.80	6.40	175.40	222.50	179.60	138.30
Standard Deviation (%)	0.00	0.80	1.20	1.10	2.00	2.50	0.70	2.90
Optimal O.F.	2020.00	1273.00	7542.00	538.00	6942.00	6528.00	21407.00	2579.00
O.F. Optimal Dist.	0.00	12.90	209.90	60.90	1793.70	2368.10	2683.80	2153.90
O.F. Optimal Dist. (%)	0.00	1.01	2.80	11.30	25.90	36.20	12.50	83.50

$^{O.F.}$ Objective Function.

Figure 14.13 shows the normalized distances between the values obtained with the experiment and the optimum values found in literature, where can be visualized a similar behavior to the parallel experiment without time limitation.

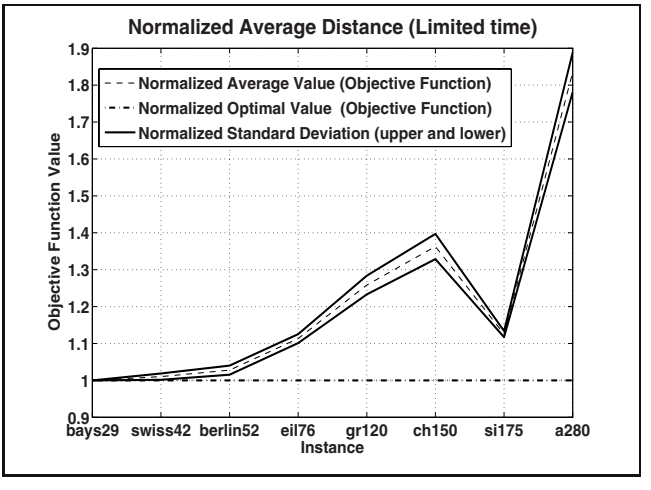


Fig. 14.13 Normalized distance between the mean Objective Functions evaluated and the optimal Object Function known

14.5.2.5 Parallel Group Execution

This experiment deals with a cooperative and competitive in a bigger hierarchy. That was done ten (10) executions for each experiment. Each experiment was done with collaboration of two parallel groups, each group consisting of components with Genetic Algorithms, *Q-learning* and GRASP. The architecture is similar to the grouping of multiple executions of the structure shown in Figure 14.3 doing communication through the critical. The aim was to assess if the collaboration would lead Objective Function value better than those obtained with previous approaches.

Table 14.7 shows the settings parameters for algorithms implementation in this set of experiments. For both Genetic Algorithm and *Q-learning* the same parameters were used rather than in the previous experiment, except for β value, that was set with different values (multiparameterized) because of the changes in values of parameters have fundamental importance for initial solutions provided from *Q-learning*, making possible to continue the produce of good solutions and further enhance these solutions. Two groups of structures were created, for Group 1, $\beta = 0.2$ and for Group 2, the $\beta = 0.3$.

Table 14.7 Parameters for algorithms executions in groups parallel experiments

Instance	iterations	(QL/GA)	(GRASP/GA)	Rand	Communications
bays29	20.000	5.00	20.00	5.00	10.00
swiss42	40.00	5.00	16.00	5.00	20.00
berlin52	50.00	5.00	13.00	5.00	20.00
eil76	50.00	6.00	8.00	5.00	20.00
gr120	70.00	7.00	7.00	5.00	30.00
ch150	100.00	8.00	6.00	6.00	50.00
si175	100.00	8.00	5.00	5.00	50.00

When comparing the table 14.3 with the table 14.7, there is a decrease of the values in columns (QL/GA) or (GRASP/GA). This is due to cooperation and competition between the groups, because it can get the same solution quality of the previous architecture with less iterations due to greater diversity.

Table 14.8 presents the results obtained in this application. As previous tables are presented in table 14.8 data with statistical values about achieved objective function values, their absolute and percentage standard deviations, the best known optimal value and the distance between the optimum values known and the obtained absolute and percentage values.

Table 14.8 Statistical data about Group parallel experiments

Instances	bays29	swiss42	berlin52	eil76	gr120	ch150	si175	a280
Mean O.F.$^{O.F.}$	2020.00	1273.00	7695.90	589.40	8560.00	8674.30	23789.00	4519.00
Standard Deviation	0.00	0.00	43.05	5.80	157.08	181.30	202.30	131.80
Standard Deviation (%)	0.00	0.00	0.56	0.98	1.81	2.10	0.85	2.90
Optimal O.F.	2020.00	1273.00	7542.00	538.00	6942.00	6528.00	21407.00	2579
O.F. Optimal Dist.	0.00	0.00	153.90	51.40	1618.00	2146.30	2381.70	1940.00
O.F. Optimal Dist. (%).	0.00	0.00	2.10	9.50	23.30	32.80	11.10	75.20

$^{O.F.}$ Objective Function.

From the table 14.8 is observed that in addition to instance bays29 the instance swiss42 also gets the average value for Objective Function with a zero % distance on the optimal value, i.e., all executions of the experiment reach the best value of solution.

14.5.2.6 Analysis of Performance

The performance analysis achieved in this section relates only to data processing and time, not including the quality of results information. A final consideration on the results is presented in the following subsection (collective analysis). The comparison of the results include data of execution time of the serial experiments (using only one machine), parallel experiments (using three machines per experiment) and in parallel groups (using six machines per experiment). Table 14.9 contains the values of the time in seconds of execution for each instance in each type of tests serial, parallel and in groups. The same information are expressed graphically in the Figure 14.14 included here for better visual perception.

Table 14.9 Execution time for each experiment

Instances	bays29	swiss42	berlin52	eil76	gr120	ch150	si175	a280
Serial Time (sec.)	220.00	577.00	952.00	1390.00	3308.00	7082.00	9975.00	20300
Parallel Time (sec.)	112.00	302.00	470.00	790.00	2030.00	4500.00	6660.00	13050.00
Group Time (sec.)	115.00	307.00	464.00	750.00	2041.00	4237.00	5487.00	13413.00

In all cases, the serial execution time is longer than the parallel execution time and parallel group time. The experimental of parallel group, has a similar time to the simple parallel implementation, a less time if readily apparent at the eil76, ch150 and si175 instances, while the simple parallel implementation time only highlights in a280 instance.

The statistical measures commonly used for evaluating performance in parallel software are the speedup and efficiency [5]. The *speedup* is obtained by the expression 14.9.

$$speedup = \frac{T_s}{T_p} \tag{14.9}$$

where T_s is the best time of serial algorithm execution and T_p the best parallel time execution, this analysis uses the average time for parallel execution as T_s . The efficiency can be calculated as shown in expression 14.10.

$$efficency = \frac{T_s}{pT_p} \tag{14.10}$$

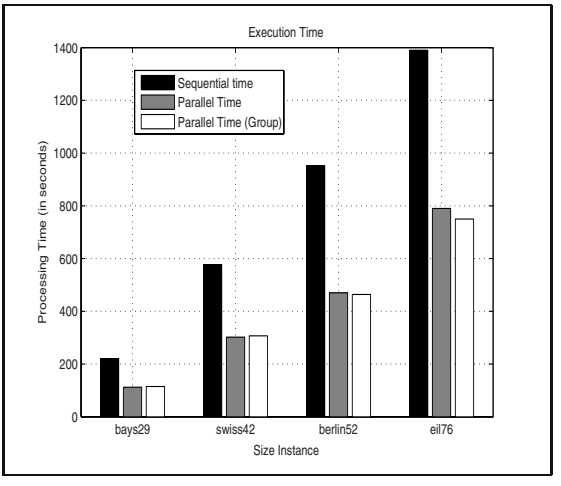


Fig. 14.14 Time for execution of experiments

where again T_s is the serial time, T_p parallel time and p is the number of processors used in the experiment. Thus the value of p is equals three in parallel experiments and equals six in group parallel experiments.

Table 14.10 gives the values of *speedup* and efficiency for each experiment in each instance examined. Since it is not possible to evaluate the characteristics of a instance only by its number of cities, can not be regarded as increasing the number of cities produces a linear increase in processing time, several factors must be considered, such as the value parameters of each algorithm and distribution of cities (the distance between them).

Table 14.10 *Speedup* and Efficiency of implementations

Instances	bays29	swiss42	berlin52	eil76	gr120	ch150	si175	a280
Parallel Speedup	1.96	1.91	2.02	1.73	1.62	1.57	1.49	1.55
Parallel Efficiency	0.65	0.63	0.67	0.57	0.54	0.52	0.49	0.51
Group Speedup	1.91	1.87	2.05	1.85	1.62	1.67	1.81	1.51
Group Efficiency	0.31	0.31	0.34	0.30	0.27	0.27	0.30	0.25

Figure 14.15 shows the curves that represent the *speedup* for parallel and parallel group implamentations for purposes of comparison. Figure 14.16 shows the efficiency curves of the parallel and parallel group experiments. That is possible to observe that the cost to maintain a good speedup for all the instances, which shows almost constant, there is just a slight reduction in efficiency, seen in Figure 14.16 because of the addition of node processors in parallel group experiment.

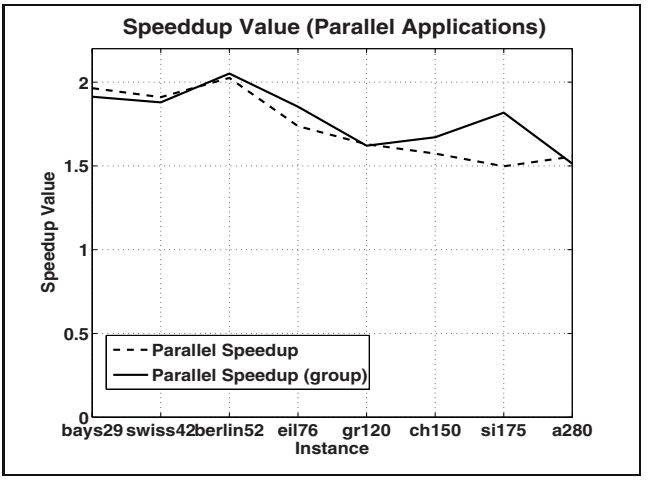


Fig. 14.15 Speedup of implementations

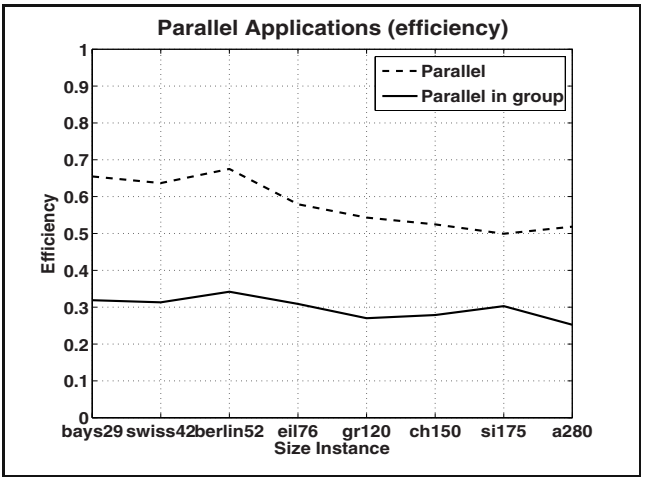


Fig. 14.16 Efficiency of implementations

14.5.2.7 Collective Analysis

Tables 14.11 and 14.12 present a qualitative comparison among all experiments in this work. In the first table are presented for each experiment: serial, parallel, time limited parallel and in group parallel, the best average values for objective function. In the last line there is the best objective function value known.

In Table 14.12 is presented the percentage distance among the average O.F. values obtained in each experiment and the known optimal value. From these tables 14.11

Table 14.11 Comparison between obtained values of objective function in all implementations

Instances	bays29	swiss42	berlin52	eil76	gr120	ch150	si175	a280
Serial	2020.00	284.00	7705.00	591.00	8789.00	9163.00	23795.00	4682.00
Parallel	2020.00	1280.00	7678.00	590.00	8638.00	8723.00	23936.00	4608.00
Limited Parallel	2020.00	1285.00	7751.00	598.00	8735.00	8896.00	24090.00	4732.00
Parallel Group	2020.00	1273.00	7695.00	589.00	8560.00	8674.00	23788.00	4519
Optimal O.F. Value	2020.00	1273.00	7542.00	538.00	6942.00	6528.00	21407.00	2579.00

Table 14.12 Percentage distance between the average values of objective function achieved by implementations

Instances	bays29	swiss42	berlin52	eil76	gr120	ch150	si175	a280
Serial (%)	0.00	0.86	2.16	9.85	26.60	40.36	11.15	81.54
Parallel (%)	0.00	0.55	1.80	9.66	24.43	33.62	11.81	78.67
Limited Parallel (%)	0.00	0.94	2.77	11.15	25.82	36.27	12.53	83.48
Group Parallel (%)	0.00	0.00	2.02	9.47	23.30	32.87	11.12	75.22

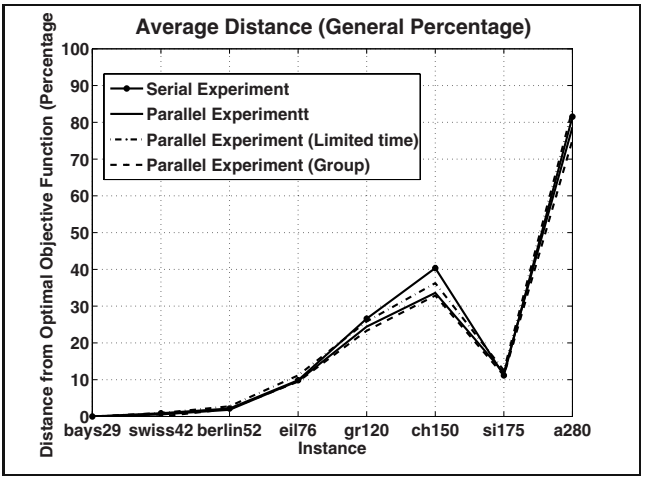


Fig. 14.17 Distance between avarege values of O.F. obtained in each experiment and the best O.F. value

and 14.12 can be seen that there is some proximity between the data obtained in each type of experiment per instance evaluated. As expected the values of the time limited parallel execution were a slight reduction in quality results when compared to other executions, but in the worst case the difference is about 8% for another instance when compared to the known optimal value, this is the case of a280 instance.

The graph of Figure 14.17 was created from these data. In this chart the values closer to zero are better because they correspond to shorter distances between the values obtained and the optimum values known. Its possible to see that the experiment that better contribute to the average solution in all instances is the parallel group communication, being the only exception the berlin52 instance, where parallel execution without time limitation has a slightly higher performance.

14.6 Conclusions

The computational results presented in this work show that the cooperative and competitive approaches achieved satisfactory results in both of cooperation and competition between them (algorithms), and cooperation and competition between groups, which in both instances tested, the bays29 and swiss42, was found the global optimum in all executions, the rest of the instances get good results as presented.

Furthermore, an performance analysis was made from the proposed approach and there was a good performance on questions that prove the efficiency and *speedup* of performed implementations. The new parallel implementation developed here reduced the execution time by increasing the number of processor nodes. The modular form of implementation of algorithms and communication infrastructure enables to create differentiated and good adaptability to high scalability, which can be used for problems of high dimensionality.

References

1. White, A., Mann, J., Smith, G.: Genetic algorithms and network ring design. Annals of Operational Research 86, 347–371 (1999)
2. Darrell, W.: A genetic algorithm tutorial. Statistics and Computing 4(2), 65–85 (1994)
3. Fang, H.: Genetic algorithms in timetabling and scheduling. PhD thesis, Department of Artificial Intelligence. University of Edinburgh, Scotland (1994)
4. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133 (1995)
5. Foster, I.: Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison-Wesley Longman Publishing Co., Inc., Boston (1995)
6. Karp, R.: On the computational complexity of combinatorial problems. Networks 5, 45–68 (1975)
7. Lima Junior, F.C., Melo, J.D., Doria Neto, A.D.: Using q-learning algorithm for initialization of the GRASP metaheuristic and genetic algorithm. In: IEEE International Joint Conference on Neural Networks, ITE - ISPEC, Orlando, FL, USA, pp. 1243–1248 (2007)
8. Prais, M., Ribeiro, C.C.: Reactive grasp: An application to a matrix decomposition problem in tdma traffic assignment. Jornal on Computing 12(3), 164–176 (2000)
9. Randy, H., Haupt, S.E.: Patrical Genetic Algoritms, 2nd edn. Wiley Intercience, Chichester (1998)
10. Reinelt, G.: The Traveling Salesman: Computational Solutions for TSP Applications, vol. 840, pp. 187–199. Springer, Heidelberg (1994)

11. Resende, M., Ribeiro, C.: GRASP with Path-relinking: Recent Advances and Applications, pp. 29–63. Springer, Heidelberg (2005)
12. Resende, M.G.C., de Sousa, J.P., Viana, A. (eds.): Metaheuristics: computer decision-making. Kluwer Academic Publishers, Norwell (2004)
13. Ribeiro, C.: Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, Norwell (2002)
14. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
15. Thangiah, S.: Vehicle Routing with Time Windows using Genetic Algorithms. In: Application Handbook of Genetic Algorithms, New Frontiers, vol. II, pp. 253–277 (1995)
16. Vázquez, M., Whitley, L.D.: A comparison of genetic algorithms for the static job shop scheduling problem. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 303–312. Springer, Heidelberg (2000)
17. Watkins, C.: Learning from delayed rewards. PhD thesis, University of Cambridge, England (1989)

Chapter 15
An Evolutionary Approach for the TSP and the TSP with Backhauls

Haldun Süral, Nur Evin Özdemirel, İlter Önder, and Meltem Sönmez Turan

Abstract. This chapter presents an evolutionary approach for solving the traveling salesman problem (TSP) and the TSP with backhauls (TSPB). We propose two evolutionary algorithms for solving the difficult TSPs. Our focus is on developing evolutionary operators based on conventional heuristics. We rely on a set of detailed computational experiments and statistical tests for developing an effective algorithm.

The chapter starts with a careful survey of the algorithms for the TSP and the TSPB, with a special emphasis on crossover and mutation operators and applications on benchmark test instances. The second part addresses our first evolutionary algorithm. We explore the use of two tour construction heuristics, nearest neighbor and greedy, in developing new crossover operators. We focus on preserving the edges in the union graph constructed by edges of the parent tours. We let the heuristics exploit the building blocks found in this graph. This way, new solutions can inherit good blocks from both parents. We also combine the two crossover operators together in generating offspring to explore the potential gain due to synergy. In addition, we make use of 2-edge exchange moves as the mutation operator to incorporate more problem specific information in the evolution process. Our reproduction strategy is based on the generational approach. Experimental results indicate that our operators are promising in terms of both solution quality and computation time.

In the third part of the chapter, we present the second evolutionary algorithm developed. This part can be thought of as an enhancement of the first algorithm. A

Haldun Süral · Nur Evin Özdemirel · Meltem Sönmez Turan
Industrial Engineering Department,
Middle East Technical University, 06531, Ankara, Turkey
e-mail: sural@ie.metu.edu.tr, nurevin@ie.metu.edu.tr, meltemsturan@gmail.com

İlter Önder
Graduate School of Informatics,
Middle East Technical University, 06531, Ankara, Turkey
e-mail: ilteronder@gmail.com

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 371–396.
springerlink.com

common practice with such algorithms is to generate one child or two children from two parents. In the second implementation, we investigate the preservation of good edges available in more than two parents and generate multiple children. We use the steady-state evolution as a reproduction strategy this time and test the replacement of the worst parent or the worst population member to find the better replacement strategy. Our two mutation operators try to eliminate the longest and randomly selected edges and a third operator makes use of the cheapest insertion heuristic. The algorithm is finalized after conducting a set of experiments for best parameter settings and testing on larger TSPLIB instances. The second evolutionary algorithm is also implemented for solving randomly generated instances of the TSPB. Our experiments reveal that the algorithm is significantly better than the competitors in the literature. The last part concludes the chapter.

Keywords: Networks-Graphs, Traveling Salesman Problems, Evolutionary Algorithms, Crossover Operator, Mutation Operator, Heuristics.

15.1 The Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is a well-known NP-hard problem widely investigated in combinatorial optimization. Given a set of n cities, the TSP seeks the shortest tour that visits every city once. The problem definition is very simple in words but finding its optimal solution is very difficult and computationally expensive. In 2006 a TSP instance with 85,900 cities was solved to optimality. The total computation time was equivalent to 136 CPU years, scaled to a 2.4 GHz AMD Opteron 250 compute node. As of July 2009 finding an optimal solution to an 100,000-city problem instance would set a new world record for the TSP [1]. Gutin and Punnen [2] provide an up-to-date coverage of the TSP literature in their recent book.

Various methods for solving the TSP to optimality have been proposed so far. Branch-and-cut algorithms have given the most effective results in general. Solution quality guarantees of 5% are obtained quickly, but it takes a very long time to get close to 1% [3]. As Rego and Glover [2] state, exact solution procedures for the TSP seem to "require computational effort that exceeds the realm of practicality". There exist heuristics that provide near optimal solutions in reasonable computation time. The collection of these heuristics provides a wide range of tradeoffs between solution quality and computation time. However, there is not a perfect correlation between increased computation time and improved quality.

In this chapter, we represent the TSP on a graph. Consider a complete graph where the node set denotes the cities and the edge set represents the arcs between them. There is a cost (or distance) for traversing each arc. The problem is to find the cheapest (shortest) tour that visits every node once on the graph.

15.1.1 Conventional TSP Heuristics

In general, conventional TSP heuristics can be classified into two categories: tour construction and tour improvement heuristics.

Construction heuristics generate a solution from scratch by a growth process that selects and inserts a node or an edge iteratively into a partial tour until a complete feasible tour is constructed. They typically get within roughly 10-15% of optimal solution in relatively short time [4]. Among the numerous procedures, we review the two that we use in this study, namely nearest neighbor and greedy.

The nearest neighbor is perhaps the most natural and simplest construction heuristic. It starts by choosing an initial node randomly, then the nearest unvisited node is added to the partial tour until a complete tour containing all nodes is constructed. The algorithm produces connections with short edges in the beginning. However, some of the nodes are "forgotten" during the process. These nodes are inserted at a high cost in the end, decreasing the tour quality. Nevertheless, nearest neighbor tours contain only a few severe mistakes, and there are several long segments connecting nodes with short edges. Therefore, these tours can serve as a good starting point for improvement heuristics. Nearest neighbor tours appear to be roughly 25% longer than the Held-Karp bound for Euclidean instances [2].

Greedy heuristic starts by sorting all of the edges in the order of increasing length. Then, a tour is generated by inserting the shortest edges in turn as long as no node has a degree greater than two, and an insertion does not result in a subtour. The tours obtained by greedy are about 17% longer than the optimal [2].

Tour improvement heuristics start with a feasible tour and try to improve it iteratively by making incremental changes on the tour. These procedures are called local search methods because they only consider tours that are the neighbors of the current tour. If a neighbor is better than the current tour, the current tour is replaced with this neighbor. This process is repeated until there are no better neighbors. Perhaps the most commonly used search technique is based on k-edge exchanges. The 2-edge exchange heuristic systematically checks each pair of nonadjacent edges of the tour and determines whether the tour length decreases by removing these two edges and adding a new pair of edges that would result in a tour. If the tour length is shortened, the exchange is performed and the search for another continues. The 2-opt examines all such exchanges for the current tour and chooses the best improving one. Average tour length with 2-opt is about 5% above the Held-Karp bound for Euclidean instances [2].

Conventional local search methods have the ability to find good, sometimes even optimal, solutions. However, because of their limited modification capabilities, they are able to find only a local minimum most of the time. Starting the search many times with different initial tours increases the chance of finding better local minima. Metaheuristics, on the other hand, try to escape from local minima in a more systematic way so as to explore the solution space methodically. Several metaheuristics such as neural networks, simulated annealing, tabu search, and evolutionary algorithms are used to solve the TSP.

15.1.2 Metaheuristics for the TSP

Leung et al. [5] propose a neural network (ESOM) that solves the TSP with reasonable quality. DePuy et al. [6] introduce a metaheuristic (Meta-RaPS) that modifies conventional heuristics by allowing randomness in construction rules. Considering nearest neighbor and cheapest insertion heuristics, [6] solves the TSP with very good results. Based on their survey, Johnson and McGeoch [4] state that simple tabu search is not very effective compared to local search heuristics such as 2-opt. Their experimentation with simulated annealing shows that the resulting solution quality is comparable to that of 2-opt. Variations of simulated annealing can catch up to multi-start Lin-Kernighan with 1% deviation, but they still have longer computation times. The evolutionary approaches Johnson and McGeoch discuss seem to be promising in solution quality.

15.1.3 Evolutionary TSP Algorithms

In Evolutionary Algorithms (EAs), crossover is a genetic operator that typically combines two parent solutions to produce new offspring, carrying solution components from one generation to the next through the evolution process. The idea is that a new offspring may be better than both of its parents if it inherits their best characteristics. The strength of crossover comes from its ability to preserve and recombine the "building blocks" when assembling new solutions. It is widely recognized that the crossover operator has a great influence on performance of EAs, since it rapidly explores the search space and exploits good solutions.

Crossover operators for the TSP can be classified by their preservation capabilities. The three main classes are position, order, and edge preserving operators [7]. Position preserving operators interpret a tour as a series of independent slots, as if the cost of assigning a node to a given position is independent of the nodes assigned to neighboring positions. Typical examples are partially mapped crossover-PMX, cycle crossover-CX, and position based crossover. Order preserving operators aim at preserving relative position of nodes. PMX tries to preserve order as well as position of nodes. Other examples of this class are order crossover-OX and order based crossover. Edge preserving operators interpret the sequence of nodes as a series of adjacency relations. Heuristic crossover, edge recombination-ER, and edge assembly crossover-EAX by [8] are examples of this class. An experimental study to compare various operators is presented by Schmitt and Amini [9].

The objective of the TSP is primarily related with the edges in the tour and secondarily with the order of nodes. According to [7] Homaifar and Guan argue that basic building blocks of the TSP are the edges as opposed to the nodes, and a good crossover operator should extract edge information from parents as much as possible. This argument is partially supported by Oliver's findings given in [7] that OX does 11% better than PMX and 15% better than CX. Xiaoming et al. [10] argue that crossovers that preserve the order or position of cities are redundant in optimization. Potvin [11] reports a similar conclusion in a survey of genetic algorithms for

the TSP. It seems that the idea of using edges rather than position or order of nodes is more promising.

Among the edge preserving operators, EAX proposed by Nagata and Kobayashi [8] seems to be particularly promising. Deviation from optimal is less than 0.03% in 21 instances from the TSPLIB [12] with $100 \leq n \leq 3000$. In generating an offspring, [8] starts with the union graph constructed with all the edges from two parents. They preserve edges from the union graph, but their detection of good edges or segments in the graph seems to be limited. Chen [13], on the other hand, assumes that the edges that are common to the parents lead to good solutions and concentrates on the intersection graph of parental edges.

Jung and Moon [14] devise natural crossover-NX where the parent tours are partitioned randomly, and partitions from different parents are merged to give partial tours. The partial tours are then merged using the shortest edges. They argue that the results they present are better than EAX and faster than distance preserving crossover-DPX. They also report that EAX "showed poorer performance than the original paper [8]". Lin-Kernighan heuristic is used to improve the results of NX, and a deviation of 0.085% is obtained for a problem instance with 11849 cities.

Merz [15] proposes a new edge recombination-NEX operator where the probabilities of inheriting an edge from the parents and selecting an edge from the complete graph can be adjusted. The results are comparable with the results of EAX for small problems. Ray et al. [16] present a crossover to improve the tours generated with the nearest neighbor heuristic. They propose fragmentation of tours generated with the heuristic and connecting the fragments using the shortest possible edges.

Considering conventional heuristics as hill climbing methods, the combination of conventional heuristics and EA seems a promising approach for solving the TSP, since EAs are also able to find the "hills" for the conventional heuristics to "climb".

15.1.4 The TSP with Backhauls

The TSP with backhauls (TSPB) is a TSP with precedence constraints, where the cities are grouped as linehauls and backhauls such that all linehauls must precede all backhauls on the tour. The TSPB arises from three different routing applications. It is a strictly constrained version of the TSP with pickup and delivery, and a special case of the vehicle routing with backhauls. Moreover, the TSPB is a three-cluster version of the clustered TSP, where one cluster contains only the depot and two others contain linehauls and backhauls [17].

Chisman [17] transforms the clustered TSP into a TSP, by adding large numbers to the inter-cluster distances, and reports that the transformed problems are solved to optimality without exception.

Gendreau et al. [18] use GENIUS heuristic [19], which basically consists of two parts. GENI tries to insert a city v between two cities, each of which becomes adjacent to v after insertion; US tries to improve the tour by using GENI operations. Gendreau et al. [18] conduct experiments with six different heuristics to solve the TSPB. The first heuristic, GENIUS, solves the TSPB using the modified cost

matrix in which large numbers are added to the inter-cluster distances. In the second heuristic, GENIUS constructs linehaul and backhaul tours separately, and then connects these tours. The third heuristic is similar to the second one except the depot is not included in the beginning. The fourth heuristic is cheapest insertion coupled with US for improving the solutions, and the fifth one is GENI coupled with Or-opt improvement moves. The last heuristic uses cheapest insertion that incorporates Or-opt. [18] reports that the first heuristic is the best, and the best results are 3-4% larger than the lower bound on average. Mladenović and Hansen [20] improve GENIUS for solving the TSPB, incorporating variable neighborhood search (VNS). VNS is a random neighborhood search mechanism in which the neighborhood size is increased until an improving move has been found. [20] reports that GENIUS coupled with VNS, G+VNS, is better than the original GENIUS by an average of 0.40% with an increase of 30% in computation time.

Ghaziri and Osman [21] use an artificial neural network (SOFM) and demonstrate that SOFM coupled with 2-opt (SOFM*) can improve the solution quality. Their test results are comparable to those of the methods that transform TSPB to TSP.

The only EA to solve the clustered TSP, developed by Potvin and Guertin [22], use the edge recombination crossover-ER and 2-opt as a mutation operator. ER is used to preserve the inter-cluster edges in the first phase and the intra-cluster edges in the second phase. The 2-opt mutation operator is applied within clusters. The results are better than those of GENIUS.

15.1.5 Outline

Our aim in this chapter is to illustrate that conventional TSP heuristics can be used as effective crossover and mutation operators. We restrict the use of crossover operators on the union graph of parents in an attempt to preserve parental edges. We describe two EAs and report their computational results in Sect. 15.2 and Sect. 15.3. The first EA uses nearest neighbor and greedy heuristics for crossover and 2-edge exchange for mutation. We choose these heuristics for illustrative purposes, but others can also be considered. We also explore combined use of multiple crossovers operators and make use of generational evolution approach. The second EA focuses on nearest neighbor crossover and explores generation of multiple offspring from more than two parents. Mutation operators used are 2-edge exchange to eliminate the longest or random edges and node insertion. The second EA takes a steady-state evolution approach. Considering the TSP materializes with side constraints in practice, we implement the second EA to solve the TSP with backhauls using the modified cost matrix and report our computational results. Sect. 15.4 concludes the chapter.

15.2 The First Evolutionary Algorithm for the TSP

In an EA, an initial population of solutions is generated either randomly or by using some heuristics. This population evolves through a number of generations

until stopping conditions are satisfied to indicate convergence. In each generation, some members of the population are placed in a mating pool as potential parents. Crossover operator is applied to selected pairs of parents and new offspring are generated. Mutation is applied to offspring mainly for diversification. These offspring replace either their parents or other members of the population, resulting in a new generation.

15.2.1 Generating Offspring from the Union Graph

The basic building blocks of the TSP are the edges. Therefore, we concentrate on preserving edges from the parents. Edges of two parents are combined to form a union graph where the degree of each node is at most four. Proposed crossover operators use this union graph while generating new tours.

In the TSP literature, heuristics are often applied on a graph of k-nearest neighbors of each node, rather than the complete graph, because most of the edges in an optimal tour appear in the reduced search space, and it brings significant savings in computation time. Our operators also use a kind of reduction, as the union graph has at most four neighbors for each node. However, it is not always possible to find a feasible edge in the union graph for constructing a complete tour. When this happens, the restriction is relaxed and all edges on the complete graph become eligible to construct a valid tour.

15.2.2 Nearest Neighbor Crossover (NNX)

Nearest neighbor takes $O(n^2)$ time [23]. Its solutions have long segments containing short edges, and it is possible to exploit these segments in the union graph and transfer them from parents to offspring. Therefore, nearest neighbor is an appropriate choice for use in a crossover operator. NNX randomly selects a node as the starting point. A single offspring is generated by visiting the nearest unvisited node, using only the edges in the union graph. In general, if we construct the union graph using edges of k parents, each node in the graph has at most $2k$ neighbors. As one of these neighbors is used to arrive at the current node, NNX constructs a tour in $n(2k-1)$ steps, provided that it always finds a feasible edge in the union graph. Otherwise, NNX resorts to the complete graph to search for the nearest unvisited node.

Suppose we have parent A (1 2 4 7 5 9 8 6 3 14 15 11 12 13 10) and parent B (1 5 4 8 7 3 2 6 9 10 11 15 13 12 14), where the numbers represent the cities in the order they are visited. Fitness values of parent tours are 43 and 51 according to the distance matrix given in Table 15.1. With the random starting node 11, NNX generates offspring (11 12 13 15 14 3 2 1 10 9 8 6 7 5 4) depicted in Figure 15.1a. The nodes adjacent to 11 are 12, 15 and 10 with distances 2, 4 and 2, respectively. There is a tie between 12 and 10, and 12 is chosen randomly. Now the unvisited nodes adjacent to 12 are 13 and 14. The distance from 12 to 13 is the shortest; hence node 13 is added to the tour. Nodes 15, 14, 3, 2, 1, 10, 9, 8 and 6 are also added

to the tour in this manner. The nodes adjacent to 6 are 8, 3, 2 and 9, all of which are already in the tour. In this case, from the complete graph, we choose 7 among the unvisited nodes 4, 5 and 7. Then, 5 and 4 are added to the tour. Note that all edges except 6-7 and 4-11 are taken from the union graph. Fitness value of the new tour is 31, which is less than fitness values of both parents. In Figure 15.1a, most of the edges in the tour are short except the lastly inserted edge 4-11. Notice that, for instance, removing edges 4-11 and 1-2, and inserting edges 1-11 and 2-4 would improve the tour, which can be achieved with 2-edge exchange.

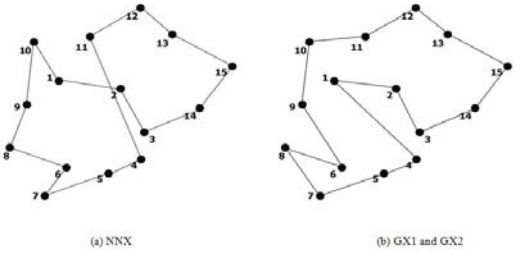


Fig. 15.1 Offspring generated by the two crossover operators

Table 15.1 Distance matrix for the example problem

Node	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	3													
3	4	2												
4	3	4	1											
5	5	5	4	1										
6	4	6	5	2	2									
7	7	8	7	5	2	2								
8	5	7	5	4	4	1	1							
9	1	4	6	5	5	3	1	2						
10	3	6	7	6	6	4	4	5	2					
11	3	4	4	5	7	5	6	7	4	2				
12	6	3	6	7	10	8	7	8	6	3	2			
13	5	2	4	7	8	8	12	8	6	3	2	1		
14	6	2	2	4	6	6	12	6	6	6	4	4	3	
15	6	3	4	7	9	9	11	7	7	7	6	3	1	1

Given a starting node, there is only one tour a deterministic NNX can generate unless there is a tie in selection. We have also tried a stochastic version of NNX where one of the edges incident to the current node is selected probabilistically. The selection probability is inversely proportional with the length of the edge. Edge selection in this version of NNX is similar to the heuristic crossover

[7], which preserves 60% of parental edges. Stochastic NNX has a potential advantage over deterministic NNX. Given two parents and a starting node, it can produce different offspring because of randomization. Hence, it is possible to increase the population diversity and portion of search space covered. Our pilot run results have shown that stochastic NNX indeed provides higher diversification, resulting in slower population convergence at the expense of longer computation times. In solution quality, however, stochastic NNX has proven to be significantly inferior to the deterministic version. Therefore we used only the deterministic version in further experimentation.

15.2.3 Greedy Crossover (GX)

Greedy heuristic constructs a tour by adding edges one at a time, starting from the shortest edge. Every time a new edge is added to the partial tour, the algorithm checks if there exists a node with a degree higher than two or a cycle with fewer than n nodes. Greedy algorithm runs in $O(n^2 logn)$ time [23].

We have tried two versions of the greedy crossover. GX1 is the same as the greedy heuristic but restricted to the union graph. Edge preservation percentage is very high with GX1 since it is usually possible to generate feasible tours by taking only the last few edges from the complete graph. Edge preservation is mainly related with the exploitation aspect of crossover. However, a desirable property of a crossover operator is to have a balance between exploration and exploitation. GX1, with extremely high edge preservation, misses the exploration aspect, which comes from inserting new edges to the offspring. This results in premature convergence of the EA with low quality solutions. To overcome lack of exploration, in GX2, we place an upper bound on the number of edges taken from the union graph. The bound is taken as the problem size n. The edges that are present in both parents are counted as two. Whenever the number of edges from the union graph reaches the bound, the union graph restriction is relaxed.

In applying GX1, the edges in the union graph are first sorted according to edge length. In our example, after inserting edges from the union graph sequentially, we see that the edges in this graph are sufficient to generate a feasible path but not a tour. Therefore, the edge 1-4 is taken from the complete graph to obtain a feasible tour. The resulting tour is offspring (11 12 13 15 14 3 2 1 4 5 7 8 6 9 10) shown in Figure 15.1b. Fitness value of this solution is 28. For this example, GX2 also results in the same tour. Although only the first half of the edges are intentionally taken from the union graph, the second half that come from the complete graph happen to be in the union graph with the exception of edge 1-4. If two parents are identical, the offspring generated by GX1 is the same as the parents with 100% edge preservation (note that it is also true for NNX). With GX2, however, half of the edges come from the complete graph and edge preservation can be as low as 50%. In our pilot runs, GX2 has yielded much better solution quality with only a slight increase in

the computation time. Therefore, we have decided to continue our experimentation only with GX2, referring to it as GX.

15.2.4 Combined Use of the Crossover Operators

Potvin and Bengio [22] used a mixture of two crossover operators for a routing problem and found that combined use of operators performed better than a single operator alone. In our multi-crossover approach, we use NNX and GX together in the same EA. Although both operators are designed to preserve short parental edges, they are different in nature. NNX selects the shortest edge among at most three edges going out of the current node in the union graph; hence the choice is myopic. GX, on the other hand, considers the union graph as a whole, resulting in less myopic behavior. Our experimentation with the two operators individually shows that they also differ in solution quality, percent edge preservation, population diversity, and computation time. NNX can find good solutions in short times whereas GX searches different parts of the solution space. Our multi-crossover approach works as follows. After a pair of parents is selected from the mating pool, one of the crossover operators is chosen probabilistically. Hence, on average, a certain fraction of offspring is generated using NNX and the remaining fraction by GX. We have tried giving different weights to two operators in our experimentation.

15.2.5 Proposed Mutation Operators

Ordinarily, mutation is a genetic operator that prevents an EA from premature convergence to a suboptimal solution. In combinatorial optimization improvement heuristics are frequently used as some sort of Lamarckian mutation (see [4]) to obtain competitive results for the TSP. To incorporate problem specific information in our EA, we use 2-edge exchange heuristic as a mutation operator.

A 2-edge exchange move is equivalent to removing a subpath from the tour and swapping it. In applying the move, we start with a randomly selected node and proceed clockwise. We consider removing the edge incident to the selected node and each of the remaining $n-2$ nonadjacent edges in the order they are given in the tour. One of the two subpaths is swapped and the tour is reconnected, introducing two new edges. The move is accepted immediately if it brings improvement. We then restart with a new randomly selected node. The total number of improvement checks is $(n-1)(n-2)/2$. Because an improving move is implemented immediately, resulting tours are not necessarily locally 2-opt. Given a tour and a starting node, this mutation is deterministic.

We have two versions of mutation. M1 applies 2-edge exchange only to the best offspring produced in every generation. In M2 all the offspring produced undergo mutation. Mutation operators are reported to have great influence on convergence behavior. Xiaoming et al. [10] prove that their genetic algorithm converges to global optima when only mutation operators are applied.

15.2.6 Other Settings of the First Evolutionary Algorithm

Our EA settings include chromosome representation, fitness function, initial population generation, parent selection and replacement, and stopping conditions.

1. *Chromosome representation:* Our literature review reveals various forms of vector representation such as path, adjacency, ordinal, and rank representations [7]. Among them, we choose the path representation for the TSP, which is the most natural and common one. In this representation, a number is assigned to every node and solutions are represented by the ordered sequence of nodes, as in our example. Unless the initial position is fixed, path representation of a given tour is not unique. Classical EA operators are not suitable for path representation, but a large number of operators are developed for this representation including PMX, CX, ER and EAX.
2. *Fitness function:* A TSP solution has a widely used fitness value, which is the tour length.
3. *Population size:* EAs converge more rapidly with smaller populations, but better results are obtained with larger populations. On the other hand, for large instances, the search space is so huge that increasing the population size further does not improve the solution quality significantly. We investigated the effect of the population size on solution quality and computation time and decided to use the population sizes of 50 and 100.
4. *Initial population generation:* Schmitt and Amini [9] report that, for various problem classes and sizes, a hybrid initial population tends to give superior results over a pure random initial population. We tried two settings to see the effect of initial population quality. In the first setting, the whole population is generated randomly. In the second one, about half of the population is still randomly generated and the other half is generated by using nearest neighbor, greedy and insertion heuristics with or without improvement heuristic 3-edge exchange. The percentages of different heuristics in the hybrid initial population are given in Table 15.2.

   In generating the hybrid population, insertion heuristic starts with a partial tour containing a few nodes (tour nodes), selects a non-tour node according to a particular criterion and then inserts that node in the tour. Selection and insertion are repeated until a tour containing all nodes is constructed. Nearest (farthest) insertion chooses the non-tour node whose minimal distance to the closest tour node is minimum (maximum). Both insert the selected node in the tour such that the increase in tour length is minimized. They run in $O(n^2)$ time. Cheapest insertion chooses the non-tour node to be inserted with the minimal increase in tour length and runs in $O(n^2 logn)$ time [23].
5. *Parent selection:* Parent selection, together with crossover and mutation operators, evolves the population toward higher fitness. If the selection pressure is too high, EA may converge to a local optimum since genetic diversity rapidly decreases. Our selection method, which is borrowed from [8], first forms a mating pool from the current population by replicating each chromosome twice. Then, it selects random pairs of parents from the pool without replacement. The crossover

operator generates one offspring from each pair. We use this method because we wish to isolate the effect of our evolutionary operators. We do not want the selection pressure to interfere with this effect, therefore, we prefer to use a neutral selection scheme.

6. *Replacement:* With newly generated offspring, the population size is temporarily doubled. For replacement, we sort parents and offspring together according to their fitness values. We then carry the best half of these chromosomes to the next generation.
7. *Stopping conditions:* We stop our EA if average fitness is exactly the same in two consecutive generations. In addition to this condition, we also use an upper bound of 500 on the number of generations, which is large enough when we consider convergence behavior of our EA in Figure 15.2 for an instance with 52 cities.

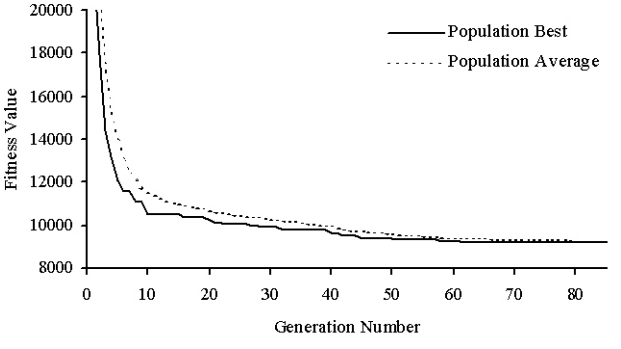


Fig. 15.2 Convergence behavior of the EA

Table 15.2 Share of different heuristics in hybrid initial population

Heuristic	%
Random	45
Nearest neighbor	10
Nearest neighbor + 3-edge exchange	10
Greedy + 3-edge exchange	5
Nearest insertion	5
Nearest insertion + 3-edge exchange	5
Cheapest insertion	5
Cheapest insertion + 3-edge exchange	5
Farthest insertion	5
Farthest insertion + 3-edge exchange	5

There is no one best configuration in terms of initial population generation, parent selection and parent replacement strategies. For instance, studies such as [8] and [24] report impressive results when the initial population is generated randomly. On

the other hand, [25], [26], and [27] are some of the studies who come up with very good solutions when the initial population is generated using a heuristic.

15.2.7 Computational Results for the TSP

Below we first describe our experimental settings including performance measures and test problems used. Then we discuss computational results for our operators with a focus on the effects due to conventional heuristics.

Our experimentation with the EA involves the following settings, which result in 18 combinations.

- Crossover operator: NNX, GX and combined use of the two.
- Mutation operator: No mutation (NoM), M1 and M2.
- Initial population (IP): Random (R) and hybrid (H).

We use the performance measures listed below in evaluating the EA.

- DB: Percent deviation of final population best from optimal solution.
- DA: Percent deviation of final population average from optimal solution.
- CT: Computation (or elapsed) time in seconds.
- NG: Number of generations until convergence.

DB is the percent deviation of the best solution found throughout the evolution in a single EA run. We also use DA to see whether or not the population has converged at the end of the run; if DB and DA are close, then convergence has been achieved. CT includes the initial population generation time. NG gives us an idea about convergence behavior of the EA. We can observe, for instance, if GX leads to faster convergence than NNX in terms of the number of generations.

We test 25 problem instances in total. They range in size from 52 to 1748. 24 problem instances selected from the TSPLIB have symmetric integer Euclidean distances. The additional problem tr81 includes the cities in Turkey. Note that only an upper bound is available for tr81, and therefore the percent deviation of its EA solutions is computed using this bound. We replicate our EA runs 30 times for small problems with $n \leq 226$ and 10 times for larger problems. The algorithm is coded in ANSI C and runs on a Pentium IV 1600 MHz machine with 256 MB RAM running RedHat Linux 8.0.

Table 15.3 includes the averages of performance measures over 30 replications of 10 problem instances with $n \leq 226$, where the initial population size p is 50. When we compare two crossovers, NNX yields better solution quality than GX and takes much shorter CT. Both M1 and M2 improve solution quality over no mutation case. M2 mutating all offspring improves NNX better compared to M1 mutating only the best. For GX, M2 performs only slightly better than M1. M2 coupled with NNX takes longer time than M1 as expected. With GX, however, M1 results in longer CT than M2 because of slower convergence of GX-M1 combination. Hybrid initial population leads to slightly better solution quality than random population. Edge preservation from the union graph is 97% for NNX and 92% for GX without mutation. Mutation reduces these figures by 2-5%. The largest NG values are

Table 15.3 Average results over 30 replications of the 10 small problem instances[†] where $p = 50$

Crossover	Mutation	IP	DB	DA	NG	CT
NNX	NoM	R	3.10	5.71	45.39	0.38
		H	4.82	5.74	33.52	2.95
	M1	R	1.67	2.30	40.21	0.62
		H	1.57	2.13	36.09	3.55
	M2	R	0.55	0.67	53.37	5.52
		H	0.55	0.73	43.53	8.11
GX	NoM	R	12.54	15.54	17.35	48.23
		H	7.19	12.58	16.37	54.27
	M1	R	4.36	7.48	60.44	208.70
		H	3.67	7.04	48.44	178.65
	M2	R	3.30	4.91	26.30	82.79
		H	3.01	4.80	25.83	90.58
50% NNX 50% GX	NoM	R	8.15	8.86	42.50	73.25
		H	5.53	7.99	38.47	75.67
	M1	R	1.92	2.90	66.04	113.81
		H	1.68	3.06	61.21	112.77
	M2	R	1.76	4.32	19.25	26.40
		H	1.61	4.16	20.68	34.19
90% NNX 10% GX	NoM	R	7.23	7.44	41.16	13.39
		H	5.19	6.03	34.93	14.95
	M1	R	1.84	2.35	55.60	19.14
		H	1.67	2.38	46.93	20.16
	M2	R	0.51	0.67	37.13	19.26
		H	0.48	0.63	37.24	21.95
95% NNX 5% GX	NoM	R	6.69	7.03	41.23	6.74
		H	5.06	5.98	33.04	8.93
	M1	R	1.77	2.39	52.62	10.03
		H	1.41	2.43	44.33	11.30
	M2	R	0.49	0.62	37.15	11.58
		H	0.44	0.58	36.19	14.88

[†] Problem instances: berlin52, tr81, eil101, bier127, ch130, pr136, ch150, u159, kroa200, and pr226.

observed when mutating only the best offspring. As expected, NG is higher for random initial population compared to hybrid one.

In implementing our multi-crossover approach, we tried mixing NNX and GX with various ratios. We started with 50% NNX and 50% GX and observed that both the solution quality and the computation time are in between those obtained with individual operators. As NNX yields better solution quality in shorter time compared to GX, we decided to increase the contribution of NNX. After experimenting with 90% and 95% NNX, we found that best results are obtained with the latter. With 95% NNX and 5% GX, M2, and hybrid initial population, the EA's results deviate

from the optimal by only 0.44%. The same figure is 0.55% with only NNX. This shows that using multiple crossover operators instead of a single one can indeed bring advantages in terms of solution quality. Our observation is also consistent with the results reported by Potvin and Bengio [22]. The results for the 10 instances are given in Table 15.4 for the best mixture of crossovers with $p = 100$. When p is increased from 50 to 100, the average deviation reduces to 0.34%, which is quite a satisfactory result in 25 seconds.

Table 15.4 Average results over 30 replications of the 10 small problem instances with 95% NNX and 5% GX where $p = 100$

Problem	IP	NoM				M1				M2			
		DB	DA	NG	CT	DB	DA	NG	CT	DB	DA	NG	CT
berlin52	R	1.22	1.91	35.9	0.7	0.07	0.45	39.0	0.8	0.00	0.00	14.6	0.8
	H	0.92	1.61	27.5	0.7	0.00	0.00	32.3	0.7	0.00	0.00	13.2	0.8
tr81	R	3.35	3.57	42.5	2.5	0.97	1.45	51.7	3.1	0.48	0.53	34.5	4.6
	H	3.48	3.80	34.2	2.7	0.84	1.21	50.6	3.6	0.47	0.48	37.7	5.8
eil101	R	6.60	6.75	62.1	12.5	2.10	2.64	83.8	9.0	0.93	1.06	69.9	15.6
	H	3.78	3.93	63.0	10.3	0.93	1.37	81.1	8.7	0.82	0.92	60.9	14.3
bier127	R	3.15	3.25	60.2	10.7	0.77	1.14	73.6	13.4	0.26	0.28	28.5	9.6
	H	3.72	3.81	53.0	12.1	0.62	0.93	71.5	15.3	0.28	0.28	27.9	12.0
ch130	R	5.49	5.68	45.8	20.0	2.14	2.49	69.0	23.8	0.76	0.90	53.4	22.6
	H	6.16	6.19	41.2	12.7	2.39	2.50	51.7	18.6	0.85	0.99	47.0	22.3
pr136	R	11.09	11.20	42.9	11.8	4.85	5.78	68.6	19.2	0.41	0.46	64.0	24.6
	H	5.92	5.92	40.4	14.2	2.87	3.52	55.3	18.4	0.37	0.49	67.1	35.2
ch150	R	3.18	3.26	44.0	15.3	0.28	0.42	58.8	21.1	0.26	0.40	26.3	15.8
	H	3.44	3.51	36.3	19.0	0.37	0.44	51.9	23.5	0.24	0.35	26.9	20.7
u159	R	6.45	6.80	42.6	14.9	0.81	0.97	6.5	23.5	0.04	0.14	31.6	19.6
	H	6.95	6.98	33.5	17.8	1.03	1.28	53.0	2.2	0.00	0.04	25.4	21.1
kroa200	R	8.37	8.43	42.6	33.3	1.68	1.78	85.7	72.4	0.33	0.43	64.1	84.3
	H	9.12	9.27	41.2	45.8	1.78	2.23	89.6	87.3	0.32	0.43	65.6	98.6
pr226	R	5.06	5.14	60.6	62.4	0.72	1.10	72.8	77.3	0.01	0.01	40.5	64.8
	H	4.26	4.27	54.4	76.4	0.79	1.21	70.5	93.0	0.01	0.07	40.2	21.6
Average	R	5.40	5.60	47.9	18.4	1.44	1.82	61.0	26.4	0.35	0.42	42.7	26.2
	H	4.78	4.93	42.5	21.2	1.16	1.47	60.8	27.1	0.34	0.41	41.2	25.2

To investigate performance of the EA for larger problems, we made a preliminary comparison test on four more problems from the TSPLIB, where $318 \leq n \leq 442$, using the best mixture of crossovers and only NNX. We also limit this test with random initial population since Table 15.4 shows that hybrid initial population does not improve consistently and brings unnecessary computational burden. Table 15.5 shows the averages of performance measures over 10 replications of these four problem instances, where p is 100. We observe that the deviation from the optimal solution and CT increase as n increases. The average deviations for the best mixture of crossovers

Table 15.5 Average results over 10 replications of the four larger problem instances† where $p = 100$

Crossover	Mutation	DB	DA	NG	CT
95% NNX 5% GX	NoM	8.43	8.46	74.5	494.0
	M1	3.90	4.66	120.9	972.4
	M2	2.01	2.74	105.2	964.0
100% NNX	NoM	4.73	7.03	92.4	4.5
	M1	3.25	4.61	133.8	11.7
	M2	2.13	3.05	102.1	206.3

† Problem instances: lin318, fl417, pr439 and pcb442.

are quite comparable with those for NNX only. Besides, the multi-crossover approach requires significantly longer computation times than NNX. Therefore, we decided to run the EA using only NNX and random initial population for solving larger test problems.

The final test bed includes 15 instances, where $318 \leq n \leq 1748$, including the four instances in the preliminary test. The average results over 10 replications are given in Table 15.6. Coupled with M1, NNX achieves an average deviation of 4.9% from the optimal in about 65 seconds. M2 requires more CT and slightly improves this deviation.

Table 15.6 Average results over 10 replications of the 15 larger problem instances with NNX only where $p = 100$

Problem	NoM			M1			M2		
	DB	DA	CT	DB	DA	CT	DB	DA	CT
lin318	4.44	6.67	3	1.87	2.60	8	2.01	2.93	105
fl417	4.93	6.67	5	2.93	4.73	14	1.83	3.34	210
pr439	4.38	6.76	4	3.44	3.68	10	1.48	1.56	240
pcb442	5.15	8.00	6	4.75	7.42	15	3.18	4.37	270
rat575	5.30	7.94	11	5.16	7.15	21	4.07	6.03	345
p654	7.47	10.50	11	3.09	7.68	27	2.22	4.70	360
d657	7.95	10.03	12	5.16	7.35	39	4.70	6.57	390
u724	6.35	8.29	15	5.14	6.64	39	3.87	5.42	720
rat783	8.40	10.38	21	5.84	6.75	46	5.33	7.46	480
u1060	9.56	13.81	38	6.68	9.06	93	7.44	10.19	1080
vm1084	10.01	12.55	24	5.77	10.58	65	6.52	9.19	1470
pcb1173	9.47	13.26	36	3.00	5.52	97	8.01	10.27	1230
nrw1379	10.60	13.26	63	7.35	10.41	181	6.65	9.09	1800
u1432	10.79	13.08	65	6.89	9.95	117	6.08	8.12	3030
vm1748	9.31	11.16	65	7.05	9.99	203	7.09	9.78	4215
Average	7.61	10.16	25.3	4.94	7.30	65.0	4.70	6.60	1063.0

Finally, we present Table 15.7 for the comparison of the EA with the two metaheuristics in the literature, Meta-RaPS [6] and ESOM [5]. It can be seen from the table that the EA outperforms both heuristics on the 10 benchmark TSPs. The table also reports CPU times for each competitor.

Table 15.7 Results for the first EA and two metaheuristics in the literature†

Problem	EA-M1		EA-M2		Meta-RaPS		ESOM3	
	DB	CT[1]	DB	CT[1]	DB	CT[2]	DB	CT[3]
eil101	0.93	8.7	0.82	14.3	NA	NA	3.43	NA
bier127	0.62	15.3	0.28	12.0	0.90	48	1.70	NA
pr136	2.87	18.4	0.37	35.2	0.39	73	4.31	NA
kroa200	1.78	87.3	0.32	98.6	1.07	190	2.91	NA
pr226	0.79	93	0.01	21.6	0.23	357	NA	NA
lin318	1.87	8	2.01	105	NA	NA	2.89	NA
pr439	3.44	10	1.48	240	3.30	2265	NA	NA
pcb442	4.75	15	3.18	270	NA	NA	7.43	NA
pcb1173	3.00	97	8.01	1230	NA	NA	9.87	200
vm1748	7.05	203	7.09	4215	NA	NA	7.27	475

[1] Pentium 4 1.6 GHz.
[2] AMD Athlon 900 MHz.
[3] SUN Ultra 5/270.

† NA: Not available. EA results are the best results given in Table 15.4 (15.6) for small (larger) problem instances.

15.3 The Second Evolutionary Algorithm for the TSP and the TSPB

The second EA can be perceived as an enhancement of the first algorithm based on our previous experimental results. Combined use of NNX and GX slightly improves the solution quality, but NNX is much faster than GX. Hence, we focus on NNX and want to explore generating multiple offspring from more than two parents in an attempt to preserve parental edges while keeping the population diverse. Also, considering that the 2-edge exchange mutation limits diversity and takes significant computation time, we propose faster mutation operators that will increase population diversity.

15.3.1 More Than Two Parents and Multiple Offspring

Mühlenbein [28] introduces more than a pair of parents to EAs. Although his voting recombination operator does not resemble NNX, the idea of multi-sexual crossover is applicable with NNX, as there is no limitation on the number of edges included in the union graph. Using more than two parents can increase edge variety and improve solution quality. However, the effect of more than two parents cannot be

easily judged, because the performance of NNX is determined by the number of good or optimal edges acquired from the complete graph as opposed to the union graph.

In an attempt to explore this idea, we form the union graph using the edges of more than two parents. We generate multiple offspring by applying NNX to this union graph. Each offspring is generated by randomly choosing a different node as the starting point of NNX. The number of offspring is expected to play an important role in exploring the search space. [8] reports favorable results with multiple offspring.

In our initial experiments with multiple parents and offspring, we use eight small instances with $n \leq 226$. Population size is set to n and initial population is generated randomly. We employ steady-state evolution instead of a generational approach. We try random and rank based parent selection with low and high selection pressures. Rank based parent selection is implemented as follows. Population members are ranked according to fitness such that the best member has rank $i = 1$ and the worst has $i = p$ where p is the population size. Then, parent selection probabilities are assigned as $pr_i = \frac{1}{p}[\eta - 2(\eta - 1)(\frac{i-1}{p-1})]$ for $i = 1, \ldots, p$. Parameter η defines the selection pressure. We use $\eta = 1.5$ ($\eta = 2.0$) for low (high) selection pressure, where $\eta = 1$ implies random selection. The best offspring replaces either the worst population member (RM) or the worst parent (RP) if it has better fitness. Otherwise it is discarded. The stopping condition is 10,000 generations for small problem instances.

We experimented with four parameters of the algorithm.

- Number of parents (P): 2, 3, 6.
- Number of offspring (C): 1, 10.
- Parent selection: Random, low selection pressure, high selection pressure.
- Parent replacement: RM, RP.

All combinations of the above parameters are replicated 30 times for each problem. Analysis of variance results show that all parameters and two-way interactions (except PxC interaction) have a significant effect at $\alpha = 0.05$. According to the two-way interaction plots, the best settings are P=2, C=10, random parent selection and RP replacement. It is interesting that, although generating multiple offspring brings a significant improvement, using more than two parents has an adverse effect. This is probably because as the number of parents increases the union graph resembles the complete graph. The tours NNX generates on this graph become similar to those that would be generated with the deterministic nearest neighbor heuristic, resulting in premature convergence. For generating diversified offspring, using two parents seems to provide a good balance between the edges inherited from the parents and the edges borrowed from the complete graph.

We have also tried generating initial population with the nearest neighbor heuristic and selecting edges in NNX by alternating between the parents (similar to the AB cycles used in EAX [8]) with no further improvements.

15.3.2 Improved Mutation Operators

NNX is useful for exploiting the edges in the union graph, but it is limited in introducing new edges from the complete graph. Three new mutation operators are developed to increase the exploration power of the second EA. These operators are based on edge exchange and node insertion, as these are the two fundamental improvement moves for the TSP.

Longest edge mutation (LEM) aims at eliminating the longest edges in an offspring based on 2-edge exchange moves. LEM exchanges the longest edge and one of the remaining nonadjacent edges with two new edges if the exchange brings an improvement. It then tries to remove the next longest edge. The motivation behind LEM is that nearest neighbor tours contain a few long edges due to "forgotten" nodes and eliminating them may bring significant savings. LEM runs in O(n) time for a single move. To reduce the computation time, the number of edge exchange moves is limited by 15. LEM1 is applied until the first improvement is found within the limit of 15 moves, LEM2 examines all 15 longest edges.

We have also tried random edge mutation (REM), which is similar to LEM with the exception that the edges to be exchanged are selected at random. REM also runs in O(n) time for a single move and has REM1 and REM2 versions. This operator does not improve the solution quality for small problem instances, but proves to be useful for larger ones.

Node insertion mutation (NIM) selects a node randomly and inserts it at the point that provides the maximum saving in the tour length. NIM that runs in O(n) time for an insertion move is also limited by 15 insertion moves. NIM1 stops at the first improvement, NIM2 attempts all 15 insertions. NIM is used in order to introduce new edges to the offspring generated by NNX.

In our experiments with mutation, we use eight small instances with $n \leq 226$. Population size is set to n and initial population is generated randomly. Parents are selected at random. Four combinations of LEM and NIM operators are tried. For example, for LEM1 and NIM2 combination, an offspring is mutated with either LEM1 or NIM2 with equal probabilities. Best half of offspring replace worst half of parents (RH) or offspring may replace more than half of parents as long as they have better fitness than parents (RMH). The stopping condition is again 10,000 generations.

We have experimented with four parameters of the algorithm.

- Number of parents and offspring (P=C): 2, 4, 6, 12.
- LEM operator: LEM1, LEM2.
- NIM operator: NIM1, NIM2.
- Parent replacement: RH, RMH.

Analysis of variance results indicate that P=C and LEM have significant effect on solution quality as well as their two-way interaction at $\alpha = 0.01$. According to the two-way interaction plots, the best settings are P=C=2, LEM2/NIM2 combination, and RMH replacement. Note that RMH with two parents is equivalent to RP with two parents given in Sect. 15.3.1.

15.3.3 Computational Results for the TSP

Results of the best settings in Sect. 15.3.1 (before mutation) and Sect. 15.3.2 (after mutation) are given in Table 15.8 for small problems. DBR used in the table is the percent deviation of the best of 30 replications from the optimal solution, and DAR is the average percent deviation of the best of each replication over 30 replications. Solution quality is significantly improved with the settings of Sect. 15.3.2 after incorporating the mutation operators. We can conclude that mutation operators are very effective even when only two offspring are generated. CTs for the small problem instances vary between 6.0 and 39.0 seconds before mutation, and between 3.7 and 16.8 seconds after mutation.

We have conducted a comprehensive convergence analysis for larger problems and experimented further with parameter settings. Using more than two parents at a time does not pay. Therefore, we construct the union graph using P=2, but repeat this with up to 10 different pairs of parents in the final algorithm. Accepting offspring that bring no improvement over their parents does not have a significant contribution to diversity. Therefore, we generate a maximum of 10 offspring until finding an improved offspring. Mutating only those offspring that are better than their parents saves CT. For larger problems, using REM2 instead of LEM2 proves to be useful in preserving population diversity. Therefore, if an offspring is better than its worse parent, then either REM2 or NIM2 is applied at random. The final form of the algorithm, reached after these experiments, is given in Figure 15.3.

```
    Set population size to n for small problems, to 200 for larger problems
    Generate initial population randomly
    for generation = 1 to NG
      for parent = 1 to 10
        Select two parents at random
        for offspring = 1 to 10
          Generate an offspring using NNX on union graph of parents
          If the offspring is better than its worse parent, go to M
        end for
      end for
M     Apply REM2 or NIM2 with equal probabilities to the offspring
      Replace the worse parent with the mutated offspring
    end for
```

Fig. 15.3 The pseudocode of the second EA

This algorithm is run for larger problem instances and their average results over 10 (5) replications for instances with $n \leq 2000$ ($n > 2000$) are summarized in Table 15.9. The algorithm achieves an average deviation of 2.3% from the optimal. As expected, CT increases as n increases. However, DBR and DAR values do not increase monotonously. Instance characteristics seem to make a difference, and problems nrw1379, u2152 and pr2392 prove to be more difficult to solve.

Table 15.8 Results of the second EA over 30 replications for the eight small problem instances

Problem	NoM with RP		LEM2/NIM2 with RMH	
	DBR	DAR	DBR	DAR
berlin52	0.00	1.24	0.00	0.10
eil101	0.79	4.32	0.00	1.60
bier127	0.61	1.74	0.28	0.43
ch130	1.64	3.12	0.49	1.08
ch150	1.23	1.95	0.32	0.46
u159	1.30	5.09	0.00	0.15
kroa200	1.25	1.83	0.14	0.63
pr226	0.86	1.44	0.35	0.79
Average	0.96	2.59	0.20	0.65

Table 15.9 Results of the second EA for the 17 larger problem instances†

Problem	NG	DBR	DAR	CT
lin318	10,000	0.92	1.36	33.1
fl417	10,000	0.95	2.03	43.0
pr439	10,000	1.33	1.48	33.9
pcb442	10,000	2.58	3.44	66.7
rat575	10,000	1.64	2.23	93.4
p654	10,000	0.91	1.19	92.7
d657	10,000	3.32	5.17	127.5
u724	10,000	2.33	3.52	118.8
rat783	10,000	2.07	3.35	140.7
u1060	40,000	1.92	2.78	1333.8
vm1084	40,000	2.53	3.29	993.2
pcb1173	40,000	3.88	4.48	1191.9
nrw1379	40,000	3.81	6.63	4775.3
u1432	40,000	2.25	3.17	1999.7
vm1748	40,000	3.13	3.59	1710.4
u2152	60,000	1.52	1.98	7623.5
pr2392	60,000	4.67	7.31	13242.6
Average	26471	2.34	3.35	1977.7

† Over 10 (5) replications for instances with $n \leq 2000$ ($n > 2000$).

15.3.4 Computational Results for the TSPB

The TSPB does not have a well-defined benchmark problem set. Therefore, we generate the TSPB instances randomly using the method proposed by Gendreau et al. [18]. The same method is also used by Mladenović and Hansen [20] and Ghaziri and Osman [21], thus all these results are comparable in terms of average results

of the test instances. We generate 30 instances for each (n,r) pair, where r is the backhaul fraction. Note that $r = 0.1$ indicates an instance with 10% backhauls.

We follow [17] to ensure that all linehauls precede all backhauls. In transforming TSPB to TSP we add a very large value to the distances between each pair of linehaul and backhaul customers. Using the modified distance matrix enforces the second EA to visit all linehauls prior to all backhauls.

Table 15.10 presents the averages over 30 randomly generated instances. Column GENI gives the results of the construction heuristic, column GENIUS presents

Table 15.10 Average results of the second EA over five replications and various metaheuristics in the literature†

n	r	GENI	GENIUS	G+VNS	SOFM	SOFM*	EA DBR	EA DAR
100	0.1	1012.5	994.12	987.11	1043.56	996.13	994.18	994.57
	0.2	1068.7	1047.01	1044.66	1072.3	1052.71	1052.06	1052.30
	0.3	1109.66	1088.09	1085.34	1108.54	1092.07	1089.04	1089.85
	0.4	1125.63	1106.69	1102.29	1131.83	1106.97	1100.51	1101.02
	0.5	1133.87	1114.34	1108.68	1123.29	1112.37	1103.16	1104.31
Average		1090.07	1070.05	1065.62	1095.90	1072.05	1067.79	1068.41
200	0.1	1418.63	1387.22	1378.8	1436.12	1381.15	1381.17	1381.85
	0.2	1498.83	1470.95	1464.88	1489.91	1462.32	1467.57	1467.85
	0.3	1550.52	1525.26	1519.93	1545	1523.71	1502.41	1502.99
	0.4	1585.76	1555.26	1548.73	1554.69	1551.48	1521.97	1522.78
	0.5	1586.93	1554.13	1546.97	1553.9	1549.61	1534.64	1535.90
Average		1528.13	1498.56	1491.86	1515.92	1493.65	1481.55	1482.27
300	0.1	1720.82	1683.76	1675.82	1702.6	1680.93	1668.96	1671.28
	0.2	1824.62	1784.8	1782.62	1787.14	1784.9	1773.81	1774.43
	0.3	1886.48	1854.86	1849.05	1877.15	1854.3	1830.29	1832.32
	0.4	1903.29	1874.43	1865.75	1876.49	1866.84	1868.45	1869.54
	0.5	1927.34	1892.2	1887.35	1891.39	1888.92	1873.38	1874.08
Average		1852.51	1818.01	1812.12	1826.95	1815.18	1802.98	1804.33
500	0.1	2197.16	2158.79	2156.61	2168.59	2161.07	2139.06	2142.44
	0.2	2342.99	2297.11	2292.04	2310.7	2297.35	2279.95	2285.57
	0.3	2409.8	2370.45	2363.16	2398.49	2376.73	2342.80	2347.55
	0.4	2443.12	2399.35	2388.07	2441.94	2397.06	2392.52	2398.11
	0.5	2464.11	2418.2	2405.55	2428.72	2410.81	2405.93	2411.29
Average		2371.44	2328.78	2321.09	2349.69	2328.60	2312.05	2316.99
1000	0.1	3099.17	3042.6	3029.76	3083.59	3048.69	3035.92	3044.57
	0.2	3281.34	3232.65	3213.61	3279.02	3228.44	3212.79	3216.77
	0.3	3366.07	3314.8	3302.93	3327.04	3312.38	3323.03	3328.28
	0.4	3451.02	3387.43	3366.23	3392.17	3371.28	3358.36	3366.75
	0.5	3455.69	3388.16	3379.67	3408.41	3386.50	3398.27	3403.74
Average		3330.66	3273.13	3258.44	3298.05	3269.46	3265.67	3272.02
Overall		2034.56	1997.71	1989.83	2017.30	1995.79	1986.01	1988.81

† Over 30 instances for each (n,r) pair.

Table 15.11 Average CPU times in seconds for the second EA and various metaheuristics in the literature†

n	r	GENIUS[1]	G+VNS[1]	SOFM[2]	SOFM*[2]	EA[3]
100	0.1	4.7	5.4	23.5	23.7	20.5
	0.2	4.8	5.8	22.1	22.8	18.5
	0.3	4.8	5.5	21.2	21.6	17.9
	0.4	5.1	5.4	27.6	28.0	17.3
	0.5	4.4	5.6	23.1	23.8	17.2
Average		4.8	5.5	23.5	24.0	18.3
200	0.1	36.3	31.6	61.1	61.7	38.0
	0.2	32.4	35.9	63.6	64.3	38.7
	0.3	31.7	30.7	71.3	71.9	35.8
	0.4	31.2	38.8	72.9	73.8	35.6
	0.5	39.6	43.1	62.1	62.6	35.2
Average		34.2	36.0	66.2	66.9	36.7
300	0.1	106.4	109.3	237.9	239.2	68.5
	0.2	105.9	87.2	278.3	283.6	65.4
	0.3	70.9	100.1	286.3	287.8	68.8
	0.4	69.6	105.6	365.0	371.0	68.7
	0.5	72.3	101.1	354.0	360.3	66.2
Average		85.0	100.7	304.3	308.4	67.5
500	0.1	325.6	343.6	732.0	749.7	317.4
	0.2	289.5	248.1	729.0	751.9	300.5
	0.3	317.7	383.3	798.0	821.5	321.4
	0.4	374.0	326.1	802.0	834.2	343.1
	0.5	405.9	472.2	852.0	872.0	295.4
Average		342.5	354.7	782.6	805.9	315.6
1000	0.1	1130.3	1417.9	1398	1428.1	1933.7
	0.2	1211.1	1637.2	1423	1495.3	1921.4
	0.3	1019.8	1643.1	1412	1432.1	1829.1
	0.4	1302.8	1898.3	1435	1470.5	1776.9
	0.5	1324.6	1762.6	1402	1440	1832.9
Average		1197.7	1671.8	1414	1453.2	1858.8
Overall		332.9	433.7	518.1	531.7	459.4

[1] SUN SPARC 10.
[2] Pentium MMX 233 MHz.
[3] AMD Turion 64 1.6 GHz.

† Over 30 instances for each (n, r) pair.

the results when US improvement is applied after GENI, and column G+VNS is for the variable neighborhood search applied on GENIUS [20]. The neighborhood is formed by node exchange moves, a node is deleted from the tour and inserted at a point that improves the tour length. Columns SOFM and SOFM* display the results of self-organizing feature map type neural network algorithms in [21].

SOFM* corresponds to the procedure where the SOFM solutions are improved with 2-opt.

The last two columns represent the results of our second EA in its final form. The EA is run for 10,000 generations for problem instances n =100, 200, 300, and for 20,000 (30,000) generations for instances with n = 500 (1000). Column EA DBR in Table 15.10 presents the average of the best of five replications over 30 problem instances, and EA DAR is the average of the best of each replication over five replications of 30 instances.

A paired t-test on the difference between EA DAR versus GENI, GENIUS, SOFM, SOFM* and G+VNS indicates that EA is better than the first four competitors at $\alpha = 0.01$. The overall average is better than that of G+VNS, but the difference is relatively small to derive a statistically significant result (p-value 0.122). When $n \leq 500$, the test on the difference between EA DAR versus GENI, GENIUS, SOFM, SOFM* and G+VNS indicates that EA is statistically the best algorithm among all alternatives. The p-value for G+VNS versus EA DAR comparison is 0.011.

The results of the random TSPB instances indicate that our second algorithm with REM2/NIM2 mutation is better than the three competitors and is comparable to G+VNS. The application of EA is simple and the constraint handling is eliminated as the algorithm can effectively find good solutions just by making the necessary modifications in the distance matrix.

Table 15.11 reports CPU times for various metaheuristics. Although each competitor uses a different machine, there are no significant differences in CPU times for problems of certain size.

15.4 Conclusion

We presented two EAs using conventional heuristics to solve TSPs. The first EA uses nearest neighbor and greedy heuristics as crossover operators. Their application on the union graph resembles the implementation of classical heuristics on a candidate graph of k-nearest neighbors of each node. The mutation operator makes use of the well known 2-edge exchange heuristic. The two crossovers are also used in a combined manner and performed better than a single operator alone, but required more computation time for larger problems. The EA solutions are significantly better than those obtained by conventional heuristics and by some recent metaheuristics in the literature. The second EA uses only NNX and three new mutation operators based on longest or random edge elimination and node insertion. Solution quality is further improved with this EA in reasonable computation times.

Considering the TSP materializes with complicating side constraints in practice, we tested our second EA on TSPB benchmark sets generated randomly. To the best of our knowledge there is no published results on the TSPB solved using an EA. Our EA in general outperforms other metaheuristics for the TSPB. Our experience shows that it might be easier to incorporate constraint handling into this form of EAs than the specialized and sophisticated classical TSP algorithms.

For future research, NNX can be further improved to become faster in solving large problems in shorter time. When offspring construction on the union graph fails, search of the complete graph for the shortest edge can be improved. A *k*-nearest neighbor candidate graph can be used to speed up the search as suggested by Yang [26]. We believe that these efforts will take us one step closer to bridging the gap between operations research and EAs because the proposed approach can also be generalized for solving other expensive combinatorial optimization problems.

References

1. Traveling Salesman Problem, http://www.tsp.gatech.edu/ (last access: July 2009)
2. Gutin, G., Punnen, A.P.: The Traveling Salesman Problem and Its Variations (Combinatorial Optimization). Kluwer Academic, Dordrecht (2002)
3. Jünger, M., Reinelt, G., Rinaldi, G.: The Traveling Salesman Problem. In: Monma, C.L., Ball, M.O., Magnanti, T., Nemhauser, G. (eds.) Network Models. Handbook on Operations Research and Management Science, vol. 7, pp. 225–230. Elsevier Science, Amsterdam (1995)
4. Johnson, D.S., Mcgeoch, L.A.: The Traveling Salesman Problem: A Case Study in Local Optimization. In: Aarts, E.H.L., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 215–310. John Wiley, New York (1997)
5. Leung, K.S., Jin, H.D., Xu, Z.B.: An Expanding Self-Organizing Neural Network for the Traveling Salesman Problem. Neurocomputing 62, 267–292 (2004)
6. DePuy, G.W., Moraga, R.J., Whitehouse, G.E.: Meta-RaPS: A Simple and Effective Approach for Solving Traveling Salesman Problem. Transportation Research Part E 41(1), 115–130 (2005)
7. Michalewicz, Z., Fogel, D.B.: How to Solve It: Modern Heuristics. Springer, New York (2000)
8. Nagata, Y., Kobayashi, S.: Edge Assembly Crossover: A High-power Genetic Algorithm for the Traveling Salesman Problem. In: Bäck, T. (ed.) Proceedings of the Seventh International Conference on Genetic Algorithms, pp. 450–457. Morgan Kaufmann, San Mateo (1997)
9. Schmitt, L.J., Amini, M.M.: Performance Characteristics of Alternative Genetic Algorithmic Approaches to the Traveling Salesman Problem Using Path Representation: An Empirical Study. European Journal of Operational Research 108(3), 551–570 (1998)
10. Xiaoming, D., Runmin, Z., Rong, S., Rui, F., Shao, H.: Convergence Properties of Non-Crossover Genetic Algorithm. In: Proceedings of the Fourth World Congress on Intelligent Control and Automation, pp. 1822–1826 (2002)
11. Potvin, J.J.: Genetic Algorithms for the Traveling Salesman Problem. Annals of Operations Research 63, 339–370 (1996)
12. Reinelt, G.: TSPLIB-A Traveling Salesman Problem Library. INFORMS Journal on Computing 3(4), 376–384 (1991)
13. Chen, S.Y.: Is the Common Good? A New Perspective Developed in Genetic Algorithms. PhD thesis, Carnegie Mellon University, Pittsburgh, USA (1999)
14. Jung, S., Moon, B.R.: Toward Minimal Restriction of Genetic Encoding and Crossovers for the Two-Dimensional Euclidean TSP. IEEE Transactions on Evolutionary Computation 6(6), 557–565 (2002)

15. Merz, P.: A Comparison of Memetic Recombination Operators for the Traveling Salesman Problem. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 472–479. Morgan Kaufmann, San Francisco (2002)
16. Ray, S.S., Bandyopadhyay, S., Pal, S.K.: New Operators of Genetic Algorithms for Traveling Salesman Problem. In: Proceedings of the 17th International Conference on Pattern Recognition, pp. 497–500. IEEE Computer Society, Washington (2004)
17. Chisman, J.A.: The Clustered Traveling Salesman Problem. Computers and Operations Research 22, 115–119 (1975)
18. Gendreau, M., Hertz, A., Laporte, G.: The Traveling Salesman Problem with Backhauls. Computers and Operations Research 23(5), 501–508 (1996)
19. Gendreau, M., Hertz, A., Laporte, G.: New Insertion and Post-Optimization Procedures for the Traveling Salesman Problem. Operations Research 40, 1086–1094 (1992)
20. Mladenović, N., Hansen, P.P.: Variable Neighborhood Search. Computers and Operations Research 24, 1097–1100 (1997)
21. Ghaziri, H., Osman, I.H.: A Neural Network Algorithm for the Traveling Salesman Problem with Backhauls. Computers and Industrial Engineering 44(2), 267–281 (2003)
22. Potvin, J.J., Bengio, S.: The Vehicle Routing Problem with Time Windows. Part II: Genetic search. INFORMS Journal on Computing 8, 619–632 (1996)
23. Reinelt, G.: The Traveling Salesman Problem, Computational Solutions for TSP Applications. Springer, Berlin (1994)
24. Baraglia, R., Hidalgo, J.I., Perego, R.: A Hybrid Heuristic for the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation 5(6), 613–622 (2001)
25. Merz, P., Freisleben, B.: Genetic Local Search for the TSP: New Results. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 159–164 (1997)
26. Yang, R.: Solving Large Traveling Salesman Problems with Small Population. In: Genetic Algorithms in Engineering Systems: Innovations and Applications Conference, pp. 157–162 (1997)
27. Tsai, H.K., Yang, J.M., Tsai, Y.F., Kao, C.Y.: A Heterogeneous Selection Genetic Algorithm for Traveling Salesman Problems. Engineering Optimization 35(3), 297–311 (2003)
28. Mühlenbein, H.: Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization. In: Proceedings of the Third International Conference on Genetic algorithms, pp. 416–421. Morgan Kaufmann, San Francisco (1989)

Chapter 16
Towards Efficient Multi-objective Genetic Takagi-Sugeno Fuzzy Systems for High Dimensional Problems

Marco Cococcioni, Beatrice Lazzerini, and Francesco Marcelloni

Abstract. Multi-objective genetic Takagi-Sugeno (TS) fuzzy systems use multi-objective evolutionary algorithms to generate a set of fuzzy rule-based systems of the TS type with different trade-offs between, generally, complexity/interpretability and accuracy. The application of these algorithms requires a large number of TS system generations and evaluations. When we deal with high dimensional data sets, these tasks can be very time-consuming, thus making an adequate exploration of the search space very problematic. In this chapter, we propose two techniques to speed up generation and evaluation of TS systems. The first technique aims to speed up the identification of the consequent parameters of the TS rules, one of the most time-consuming phases in TS generation. The application of this technique produces as a side-effect a decoupling of the rules in the TS system. Thus, modifications in a rule do not affect the other rules. Exploiting this property, the second technique proposes to store specific values used in the parents, so as to reuse them in the offspring and to avoid wasting time. We show the advantages of the proposed method in terms of computing time saving and improved search space exploration through two examples of multi-objective genetic learning of compact and accurate TS-type fuzzy systems for a high dimensional data set in the regression and time series forecasting domains.

16.1 Introduction

Multi-objective Genetic Fuzzy Systems (MGFSs) [18] are interesting multi-objective computational intelligent methods successfully employed in regression [3, 4, 8, 9, 10], classification [19, 20, 21], data mining [7, 22] and control [32], which are receiving increasing attention in the research community [12]. MGFSs extend Genetic

Marco Cococcioni · Beatrice Lazzerini · Francesco Marcelloni
Dipartimento di Ingegneria dell'Informazione, University of Pisa,
Largo Lucio Lazzarino 1, 56122 Pisa, Italy
e-mail: m.cococcioni@iet.unipi.it, b.lazzerini@iet.unipi.it, f.marcelloni@iet.unipi.it

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 397–422.
springerlink.com

Fuzzy Systems (GFSs) [16] in that more than one objective can be optimized; this provides a set of equally optimal solutions which approximate the Pareto optimal front. By visually inspecting the obtained front the user can select his preferred trade-off among the different objectives (an example of a typically used pair of objectives is complexity/interpretability and accuracy).

In most existing MGFSs, Fuzzy Rule-Based Systems (FRBSs) are optimized through a multi-objective evolutionary algorithm. While such systems have been widely studied and applied to low dimensional data sets, more research is still required to make them practical for high dimensional data sets [16, 25]. By High Dimensional Data Sets (HDDSs) we mean data sets having a high number of input features and, sometimes, a huge number of training data. In this study we will consider only FRBSs of the Takagi-Sugeno (TS) type (denoted TS systems in the following), since they are known to be very powerful in regression and control tasks. However, their use is particularly time consuming, since they consist of if-then rules where the consequent parts involve crisp functions (typically linear), whose parameters have to be estimated through a sequence of pseudoinversions. When dealing with HDDSs, the most time consuming task in the evolutionary multi-objective optimization of TS systems resides in the fitness evaluation, and, in particular, in consequent parameters estimation [11].

Most available designing methods of TS-type MGFSs are impractical for HDDSs. Moreover, it is very unlikely that a single strategy can make the optimization efficient. Conversely, we think that only a combination of different methods can overcome computational bottlenecks. In the following we list some of the methods that, once appropriately combined, can achieve significant improvements in the efficient implementation of TS-type MGFSs for HDDSs: i) fast identification of the TS systems [11] through fitness approximation [26], ii) reuse of previously computed quantities in evaluating the fitness of a TS system (activation degrees, etc ...), iii) fitness inheritance [6, 14], iv) landscape modeling to reduce the number of fitness evaluations [27], v) parallel Multi-Objective Evolutionary Algorithms (MOEAs) on parallel/multicore/distributed machines [5, 33, 34, 35].

This work aims to extend the results obtained in [11] by considering the integration of the first two methods among those listed above: a fast identification of consequents of the TS systems and the reuse of previously computed quantities. The first technique is described in [11] and will be briefly discussed in this chapter. As regards the reuse of previously computed quantities, our simple idea is to store, for all the rules in the current population, their activation degrees and their unnormalized weighted outputs. Since fast identification of consequents is based on a decoupling of rules, modifications in a rule do not affect the consequents of the other rules. Thus, thanks to the fast identification, we can avoid re-estimating consequent parameters for all the rules that are not modified. For instance, the re-estimation can be avoided when we apply crossover followed by no mutation: if the crossover point is between rules and not within rules, no re-estimation of consequent parameters is needed for the offspring. Similarly to crossover, we can completely avoid re-estimation when we apply mutation operators which remove rules from the rule base. Further, we can limit the re-estimation only to the added or modified rules

when we apply mutation operators which, respectively, add or modify rules. Finally, we also discuss how the reuse of previously computed parameters can speed up the evaluation phase of the TS systems.

In the experimental part, we show the benefits of the combined use of the two methods through two examples of multi-objective genetic learning of compact and accurate TS systems for HDDSs in a regression problem and a chaotic time series forecasting problem, respectively. We evaluate the saved time with respect to the application of the MGFS without the adoption of the two methods and discuss the advantages that this saved time provides in terms of search space exploration.

The rest of the chapter is organized as follows. Section 16.2 introduces the TS systems. In Section 16.3, we compute the asymptotical time complexity associated with the identification of the consequent parameters, and with the evaluation of the TS systems. Section 16.4 introduces the technique of the fast identification of consequents. In Section 16.5, we discuss how the reuse of computed parameters can speed up both the identification and the evaluation of the TS systems. Section 16.6 describes the MOEA used in the experiments, which are shown in Section 16.7. Finally, Section 16.8 draws some conclusions.

16.2 The Takagi-Sugeno Fuzzy Systems

Let $\{X_1,\ldots,X_F\}$ be a set of variables and $\mathbf{x} = [x_1,\ldots,x_F]^T$ a generic input vector, with $x_1,\ldots, x_F \in \Re$. Let U_f, $f = 1,\ldots,F$, be the universe of variable X_f and let $P_f = \left\{A_{f,1},\ldots,A_{f,Q_f}\right\}$, $f = 1,\ldots,F$, be a fuzzy partition of U_f made of Q_f fuzzy sets. To simplify the discussion, in the following we will use $Q_f = Q$, i.e., the same granularity for all inputs. The m^{th} rule of a first-order TS-type FRBS [2] has the form:

$$r_m : \textbf{IF } X_1 \textbf{ is } A_{1,j_{m,1}} \textbf{ AND } \ldots \textbf{ AND } X_F \textbf{ is } A_{F,j_{m,F}} \textbf{ THEN}$$
$$y_m = p_{m,0} + p_{m,1}\cdot x_1 + \ldots + p_{m,F}\cdot x_F,$$

where the generic element $j_{m,f}$ indicates that the $j_{m,f}^{th}$ fuzzy set of partition P_f has been selected for variable X_f in rule r_m, with $m = 1,\ldots,M$. Indexes $j_{m,f}$ can be viewed as entries of a matrix $\mathbf{J}$ of natural numbers having size $M \times F$. The $(F+1)$ parameters constituting the **THEN** part, called *consequent parameters*, can be stored in a column vector $\mathbf{p}_m = [p_{m,0}, p_{m,1},\ldots,p_{m,F}]^T$. The output of the m^{th} rule can thus be computed as: $y_m(\mathbf{x}) = \hat{\mathbf{x}}^T \cdot \mathbf{p}_m$, where $\hat{\mathbf{x}} = \left[1,\mathbf{x}^T\right]^T$. Once a pattern $\mathbf{x}$ is fed to a TS system, the activation degrees $w_m(\mathbf{x})$, $m = 1,\ldots,M$, are computed for each rule as: $w_m(\mathbf{x}) = \prod_{f=1}^{F} A_{f,j_{m,f}}(x_f)$. Then, the output of the system is calculated as the weighted average of the outputs of each rule weighted by the corresponding activation degree: $y(\mathbf{x}) = \sum_{m=1}^{M} \left(w_m(\mathbf{x}) \big/ \sum_{h=1}^{M} w_h(\mathbf{x})\right) \cdot y_m(\mathbf{x})$. By defining

$$v_m(\mathbf{x}) = w_m(\mathbf{x}) \Big/ \sum_{h=1}^{M} w_h(\mathbf{x}) \tag{16.1}$$

we can also express the output as:

$$y(\mathbf{x}) = \sum_{m=1}^{M} v_m(\mathbf{x}) \cdot \hat{\mathbf{x}}^T \cdot \mathbf{p}_m. \tag{16.2}$$

All the M consequent vectors $\mathbf{p}_m$ can be stored in a matrix $\mathbf{P} \in \Re^{(F+1)\times M}$: $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_M]$. It is now clear that under our assumption of static (not tuned during the optimization process) membership functions (MFs), the identification of a TS system reduces to the choice of the number of rules M and the identification of matrices $\mathbf{J}$ and $\mathbf{P}$.

A simple way to obtain M and $\mathbf{J}$ is to adopt grid partitioning, i.e., all possible rules obtained by all combinations of fuzzy sets belonging to different input variables. In this case $M = \underbrace{Q \cdot \ldots \cdot Q}_{F\ times} = Q^F$. Alternatively, matrix $\mathbf{J}$ can be evolved by a single or multiple objective evolutionary algorithm, with variable length integer encoding (matrix $\mathbf{J}$ is coded row by row). In this case, the number of rules M corresponds to the number of rows in $\mathbf{J}$. As regards the estimation of consequent parameters $\mathbf{p}_m$, they can be estimated in different ways once matrix $\mathbf{J}$ has been provided.

Given a set of N input-output data pairs $\{\mathbf{x}_n, o_n\}_{n=1}^{N}$ ($\mathbf{x}_n \in \Re^F$, $o_n \in \Re$) constituting the training set, the optimal estimation of the consequent parameters can be done either locally or globally [1, 36]. However, local learning gives more locally meaningful consequent parameters and leads to better conditioned least squares estimations. In the local estimation approach, the optimal consequent parameters can be found using pseudoinversion [1]:

$$\mathbf{p}_m = [\hat{\mathbf{X}}^T \cdot \mathbf{V}_m \cdot \hat{\mathbf{X}}]^{-1} \cdot \hat{\mathbf{X}}^T \cdot \mathbf{V}_m \cdot \mathbf{o}, \quad m = 1, \ldots, M, \tag{16.3}$$

where $\hat{\mathbf{X}} \in \Re^{N\times(F+1)}$ is the matrix defined as $\hat{\mathbf{X}} = [\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_N]^T$, $\hat{\mathbf{x}}_n = [1, x_{n,1}, \ldots, x_{n,F}]^T$, $n = 1, \ldots, N$, $\mathbf{V}_m \in \Re^{N\times N}$ is a diagonal matrix having zeros in non-diagonal entries and $v_m(\mathbf{x}_n)$ along the diagonal ($\mathbf{V}_m = diag\{v_m(\mathbf{x}_1), \ldots, v_m(\mathbf{x}_N)\}$), and $\mathbf{o}$ is the column vector with the desired outputs: $\mathbf{o} = [o_1, \ldots, o_N]^T$.

16.3 Time Complexity

To determine the fitness of each solution, we have first to identify the consequent parameters of each rule and then to evaluate the output of the corresponding TS system for each pattern in the training set. In [11], we have already discussed how the asymptotical time complexity associated with both the identification of the consequent parameters and the TS system evaluation can be determined by using the big O notation. All the computations were performed by assuming that the complexity of multiplying two matrices $\mathbf{Z}_1$ and $\mathbf{Z}_2$, ($\mathbf{Z}_1 \in \Re^{N_1\times N_2}$, $\mathbf{Z}_2 \in \Re^{N_2\times N_3}$) is $O(N_1 \cdot N_2 \cdot N_3)$. Actually, this is the number of operations needed in a straightforward implementation in Fortran or C/C++ using "for" loops. In [11], we highlighted that, although more efficient algorithms exist for computing the product between square [13] and

rectangular [17] matrices, they are generally faster only asymptotically and quite difficult to implement. On the other hand, the formulas in [11] are still valid even when using fast multiplication algorithms, but with a lower impact.

In the following, we will briefly recall the procedure and the results shown in [11]. Without considering the time required to compute $\mathbf{V}_m$ (which will be analyzed later), it is well known that the time complexity, independently of the technique used to solve the weighted least squares problem, associated with each pseudoinversion is $O(N \cdot (F+1)^2)$. Estimating the consequent parameters of a rule base made of M rules will therefore require $O(M \cdot N \cdot (F+1)^2)$ operations.

As regards the time complexity associated with the evaluation of a TS system, assuming that $\mathbf{J}$ and $\mathbf{P}$ are already known, we first compute the M vectors of activation degrees $\mathbf{w}_m$, ($\mathbf{w}_m = [w_m(x_1), \ldots, w_m(x_N)]^T$, $m = 1, \ldots, M$), and then we derive the normalized vectors $\mathbf{v}_m$. Often, it might be more efficient to work with vectors and matrices instead of using scalar quantities (for instance, when adopting Matlab [15]). To this aim, it is helpful to first define vectors $\mathbf{a}_{f,q}$, ($\mathbf{a}_{f,q} = [A_{f,q}(x_{1,f}), \ldots, A_{f,q}(x_{N,f})]^T$, $f = 1, \ldots, F$, $q = 1, \ldots, Q$), to represent the membership degrees of each input variable to the pertinent fuzzy set for the whole training set. Then we compute all the $\mathbf{w}_m$ from $\mathbf{a}_{f,q}$ using the vectorized form:

$$\mathbf{w}_m = \prod_{f=1}^{F} \mathbf{a}_{f,j_{m,f}}. \tag{16.4}$$

The complexity for computing each $\mathbf{a}_{f,q}$ is $O(N)$, assuming that the complexity for computing each $A_{f,q}(x_{n,f})$ is constant, i.e., $O(1)$. For instance, the latter assumption is true when using triangular, trapezoidal or generalized Bellman MFs. The use of gaussian MFs may require significantly higher computations [30], since it is not a rational function. Nevertheless, even for this case, efficient approximated algorithms exist, which exploit the internal representation of floating point numbers [31]. In the considered application, the evaluation of the MFs will never be a bottleneck, since in Multi-objective Genetic Rule Learning (MGRL) each $\mathbf{a}_{f,q}$ can be computed just once at the beginning and reused during the optimization process. We can observe now that computing each $\mathbf{w}_m$ from $\mathbf{a}_{f,q}$ takes $O(N \cdot F)$ time. As regards $\mathbf{v}_m$, an efficient way to compute them in a vectorized form from $\mathbf{w}_m$ can be obtained by first building the new matrix $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_M]$ ($\mathbf{W} \in \Re^{N \times M}$) and then computing the vector $\mathbf{s} = [s_1, \ldots, s_N]^T$, in $O(M \cdot N)$ time, where $s_n = 1 / \sum_{m=1}^{M} w_m(\mathbf{x}_n)$. Second, we build matrix $\tilde{\mathbf{S}} = \underbrace{[\mathbf{s}, \ldots, \mathbf{s}]}_{M \text{ times}} \in \Re^{N \times M}$ and finally compute $\mathbf{V} = \mathbf{W} \circ \tilde{\mathbf{S}}$ in $O(M \cdot N)$ time, $\mathbf{V} \in \Re^{N \times M}$, where $\circ$ is the entrywise product and $\mathbf{v}_m$ coincide with columns of $\mathbf{V}$. In summary, computing all the M normalized activation degrees $\mathbf{v}_m$ takes $O(M \cdot N)$ time. We can address now the complexity of computing the output vector and the Mean Squared Error (MSE). First note that the vectorized output $\mathbf{y}_m = [y_m(x_1), \ldots, y_m(x_N)]^T$ of the m^{th} rule can be written in matrix form as: $\mathbf{y}_m = \hat{\mathbf{X}} \cdot \mathbf{p}_m$. The complexity of computing each $\mathbf{y}_m$ is therefore $O(N \cdot (F+1))$ starting from $\mathbf{p}_m$ and $\hat{\mathbf{X}}$. The complexity for computing all the M outputs $\mathbf{y}_m$ is therefore $O(M \cdot N \cdot (F+1))$.

The computation of $\mathbf{y} = \sum_{m=1}^{M} \mathbf{v}_m \circ \mathbf{y}_m$ first requires to compute each $\mathbf{v}_m \circ \mathbf{y}_m$, which has complexity $O(N)$ when starting from $\mathbf{v}_m$ and $\mathbf{y}_m$. Thus $\mathbf{y}$ is just the sum of the M vectors $\mathbf{v}_m \circ \mathbf{y}_m$ and has complexity $O(M \cdot N)$. Globally, the complexity to evaluate the output vector $\mathbf{y}$ starting from $\mathbf{a}_{f,q}$ (without considering the time for computing $\mathbf{p}_m$) is $O(M \cdot N \cdot (F+1))$. Finally, we observe how computing the MSE between $\mathbf{y}$ and $\mathbf{o}$ has complexity $O(N)$, since MSE $= mean(\mathbf{e})$, and computing $\mathbf{e} = (\mathbf{y} - \mathbf{o}) \circ (\mathbf{y} - \mathbf{o})$ has complexity $O(N)$.

16.4 Fast Identification of Consequent Parameters

In the previous section we have found out that the most time-consuming task concerns the estimation of the M consequent vectors $\mathbf{p}_m$ ($O(M \cdot N \cdot (F+1)^2)$, while the TS system evaluation requires $O(M \cdot N \cdot (F+1))$ operations. In [11] we have proposed an approach to speed up the identification of consequent parameters. The approach is based on the following observation: since gaussian functions have thin tails and are multiplied by each other, it is likely that most of the patterns involved in each pseudoinversion receive a low weight and thus they loosely affect the estimation of consequent parameters. From this observation we have introduced the simple idea of giving maximum weights only to patterns belonging to the specific cell of the grid associated with the m^{th} rule and zero weights to all the others. We have implemented this solution by defining, for each input variable, a new partition $\bar{P}_f = \{\bar{A}_{f,1}, \ldots, \bar{A}_{f,Q}\}$ built upon the original gaussian partition P_f and made of non overlapping and adjacent rectangular MFs. Each rectangular MF begins and ends, respectively, at the consecutive intersections of two consecutive gaussian functions. The new MF vectors and the new weighting vectors are therefore $\bar{\mathbf{a}}_{f,q} = [\bar{A}_{f,q}(x_{1,f}), \ldots, \bar{A}_{f,q}(x_{N,f})]^T$, $f = 1, \ldots, F$, and $\bar{\mathbf{w}}_m = \prod_{f=1}^{F} \bar{\mathbf{a}}_{f,j_{m,f}}$, $m = 1, \ldots, M$, respectively.

It is straightforward to note that in this case $\bar{\mathbf{v}}_m = \bar{\mathbf{w}}_m$, i.e., weights $\bar{\mathbf{w}}_m$ result to be already normalized. We wish to point out that we use $\bar{\mathbf{v}}_m$ in place of $\mathbf{v}_m$ only for estimating consequents, while in evaluating the output of the TS system we continue to use $\mathbf{v}_m$, since gaussian MFs assure better approximation capabilities than rectangular ones. The new formula for computing the consequent parameters is ($m = 1, \ldots, M$):

$$\mathbf{p}_m = [\hat{\mathbf{X}}^T \cdot \bar{\mathbf{V}}_m \cdot \hat{\mathbf{X}}]^{-1} \cdot \hat{\mathbf{X}}^T \cdot \bar{\mathbf{V}}_m \cdot \mathbf{o}, \tag{16.5}$$

where $\bar{\mathbf{V}}_m = diag\{\bar{v}(\mathbf{x}_1), \ldots, \bar{v}(\mathbf{x}_N)\}$. Formula (5) is identical to formula (3), but formula (5) can be implemented in a faster way. Let $\mathbf{i}_m$ be the vector containing the indexes of non-zero entries in $\bar{\mathbf{v}}_m$. Consequent parameters can now be computed using unweighted least squares on a subset of training data ($m = 1, \ldots, M$):

$$\mathbf{p}_m = [\hat{\mathbf{X}}_{\mathbf{i}_m}^T \cdot \hat{\mathbf{X}}_{\mathbf{i}_m}]^{-1} \cdot \hat{\mathbf{X}}_{\mathbf{i}_m}^T \cdot \mathbf{o}_{\mathbf{i}_m}, \tag{16.6}$$

where $\hat{\mathbf{X}}_{\mathbf{i}_m}$ represents the matrix extracted from $\hat{\mathbf{X}}$ by considering only rows having indexes $\mathbf{i}_m$, and $\mathbf{o}_{\mathbf{i}_m}$ represents the entries $\mathbf{i}_m$ of the output vector $\mathbf{o}$. It is now evident how the new complexity is $N_m \cdot (F+1)^2$, N_m being the length of vector $\mathbf{i}_m$. The

complexity associated with the estimation of all the consequent parameters, therefore, is $\sum_{m=1}^{M} N_m \cdot (F+1)^2$. Observe that now $\sum_{m=1}^{M} N_m \leq N$, and thus the new complexity $\sum_{m=1}^{M} N_m \cdot (F+1)^2$ is lower than, or equal to, $N \cdot (F+1)^2$, which is, on its turn, lower than $M \cdot N \cdot (F+1)^2$. However, we have to include the complexity associated with computing the vectors $\mathbf{i}_m$, which consists of $M \cdot N \cdot F$ operations. Globally, the complexity associated with the estimation of consequent parameters in the fast approach is: $M \cdot N \cdot F + \sum_{m=1}^{M} N_m \cdot (F+1)^2$, which is lower than $M \cdot N \cdot (F+1)^2$.

Real data patterns are not uniformly distributed over the input space; as a consequence, some cells could receive fewer training data points than others or no training points at all. However, this situation can be managed by removing rules with insufficient number of patterns (for each cell we need to estimate $(F+1)$ parameters and thus we could require at least a number of patterns equal to $4 \cdot (F+1)$). In MGRL it can be meaningful not to consider a rule if few data points are directly related to it. We have already shown in [11] that this approach considerably speeds up the identification of the consequent parameters, without particularly deteriorating the modeling capabilities of the TS system.

16.5 Reuse of Computed Parameters

In this section, we will discuss how the reuse of previously computed parameters can further reduce the time complexity of the identification of the consequent parameters when we apply the mating operators during the evolutionary process. Then, we focus on speeding up the evaluation of the outputs. Indeed, when we apply the fast identification of the consequent parameters and the reuse of components during the evolutionary process, the complexity of the identification of consequent parameters could considerably diminish, thus making the search for reduction of the time needed for output evaluation attractive. Finally, we focus on speeding up the computation of the activation degrees by using reuse.

All the time complexity reduction techniques proposed in this section are based on storing and reusing previously computed parameters and therefore can suffer from the drawback of memory occupation. However, the constant increase of the available physical memory even in desktop computers makes this drawback less relevant than time complexity in many applications. The choice of the best trade-off between time complexity and memory occupation is obviously application-dependent.

16.5.1 Reuse in the Application of Mating Operators

We will take into account different scenarios which typically occur when we apply an MOEA to generate a set of TS systems with a good trade-off between complexity and accuracy. The first scenario considers the application of a crossover operator during the evolutionary process. The second, third and fourth scenarios consider the application of generic mutation operators, which can add, remove and modify rules, respectively.

16.5.1.1 Application of the Crossover Operator

We focus on no specific crossover operator. Actually, the scenario is suitable for any crossover operator which acts at the rule level, that is, does not exchange parts of rules, but rather exchanges only rules. For instance, if we apply the one-point crossover, one of the most used crossover operators, the crossover point has to be chosen between two rules and not within a rule. When we apply the crossover operator, the offspring generated from the two parents inherit rules from both the parents. In the case of the classical approach, the matrix $\mathbf{P}$ of the consequent parameters has to be recomputed for both the offspring. Indeed, the values of the vectors $\mathbf{V}_m$ used in the computation of the pseudoinversion (see formula 16.3) depend on the activation degrees of all the rules contained in the rule base (see formula 16.1).

On the contrary, in the fast method, rules do not depend on each other. Indeed, indexes $\mathbf{i}_m$ used in the computation of the pseudoinversion (see formula 16.6) do not depend on the activations of the other rules, but only on the activation of the m^{th} rule. Thus, in the fast method, no computation of the consequent parameters is needed after applying a crossover operator. This allows considerably speeding up the evolutionary process. We recall that the estimation of the consequent parameters of a rule base composed of M rules requires $O(M \cdot N \cdot (F+1)^2)$ operations. Thus, due to an interesting property of the fast identification method, the application of the crossover operator does not require the execution of the most time consuming task in TS identification.

16.5.1.2 Removing Rules by Mutation

As we have already discussed above, when we apply an operator which changes the rule base, the overall matrix $\mathbf{P}$ of the consequent parameters has to be recomputed. This occurs also when we apply a mutation operator which removes one or more rules from the rule base. In the case of the fast identification method, since rules are independent of each other, no computation is needed. Indeed, the consequent parts of the survived rules have not to be updated. Thus, also the application of the mutation operator, which removes rules, does not require the execution of the most time consuming task in TS identification.

16.5.1.3 Adding Rules by Mutation

When we apply a mutation operator which adds rules to the rule base, the overall matrix $\mathbf{P}$ of the consequent parameters has to be recomputed in the classical approach. Indeed, since the rule base is changed, the consequent parameters of all the rules have to be updated. On the contrary, when we adopt the fast identification method, only the consequent parameters of the added rules have to be computed. Indeed, the rules already existing in the rule base have not to be updated. Thus, the application of the mutation operator, which adds rules, requires the execution of the most time consuming task in TS identification only for the added rules.

16.5.1.4 Modifying Rules by Mutation

When we apply a mutation operator, which modifies some rules in the rule base, the overall matrix $\mathbf{P}$ of the consequent parameters has to be recomputed in the classical approach. Indeed, since the rule base is changed, the consequent parameters of all the rules have to be updated. On the contrary, when we adopt the fast identification method, only the consequent parameters of the modified rules have to be computed. Indeed, the unmodified rules have not to be updated. Thus, the application of the mutation operator, which modifies rules, requires the execution of the most time consuming task in TS identification only for the modified rules.

16.5.2 Speeding Up the Calculus of the Output through Reuse

The fast identification of the consequent parameters and the reuse of these parameters during the application of the mating operators significantly reduce the time spent to estimate the consequent parameters during the evolutionary process, thus making the analysis of the complexity reduction for other computations attractive. In particular, the computation of the output $\mathbf{y}$, which had been judged to be negligible and therefore not deserving attention, can now represent a significant fraction of the overall fitness evaluation time.

In Section 16.3, we have determined that, once computed the $\mathbf{v}_m$ vectors, the time complexity needed to calculate the output $\mathbf{y}$ of the system is $O(M \cdot N \cdot (F+1))$. Unfortunately, since we use directly the $\mathbf{v}_m$ vectors to compute $\mathbf{y}$, no reuse of previously computed parameters is possible. Indeed, as described in the previous subsections, vectors $\mathbf{v}_m$ have to be recomputed whenever the rule base changes. On the other hand, we decided to use vectors $\mathbf{v}_m$ in place of vectors $\bar{\mathbf{v}}_m$ because vectors $\mathbf{v}_m$ guarantee a better fitting.

Nevertheless, by introducing a different method to compute $\mathbf{y}$, we can exploit reuse also in the output evaluation. Such new expression for $\mathbf{y}$ is based on the computation of vectors $\mathbf{u}_m \in \Re^N$, defined as $\mathbf{u}_m = \mathbf{w}_m \circ (\hat{\mathbf{X}} \cdot \mathbf{p}_m)$, which can be computed in $O(N \cdot (F+1))$ time, once $\mathbf{w}_m$ and $\mathbf{p}_m$ are available (please notice that $\mathbf{u}_m$ is the unnormalized weighted output of rule r_m). The output $\mathbf{y}$ can be calculated as $\mathbf{y} = \left(\sum_{m=1}^{M} \mathbf{u}_m\right) \circ \mathbf{s}$ in $O(M \cdot N)$ time, starting from $\mathbf{u}_m$ and $\mathbf{s}$, where $\mathbf{s}$ is defined in Section 16.3 as the element-wise inverse of the sum of $\mathbf{w}_m$ vectors. Computing vector $\mathbf{s}$ has complexity $O(M \cdot N)$ when starting from $\mathbf{w}_m$, and $O(M \cdot N \cdot F)$ when starting from $\mathbf{a}_{f,q}$. Thus, the complexity associated with computing $\mathbf{y}$ by starting from $\mathbf{v}_m$ and $\mathbf{p}_m$, or from $\mathbf{u}_m$ and $\mathbf{s}$ is the same and equal to $O(M \cdot N \cdot (F+1))$.

However, in the latter case, the $\mathbf{u}_m$ vectors do not depend on each other, and thus they can be stored and reused. Reusing the $\mathbf{u}_m$ allows saving computational time, since the complexity for computing $\mathbf{u}_m$ is $N \cdot (F+1)$. In the best case, when 100% of the $\mathbf{u}_m$ are reused (in case of crossover followed by no mutation and crossover followed by mutation that removes rules), the complexity of computing $\mathbf{y}$ descreases to $O(M \cdot N)$, once the $\mathbf{w}_m$ are available. Since $\mathbf{w}_m$ can be stored and reused as well (as discussed in the next subsection), this is also the total complexity. This means

that in such cases we can save a factor $(F+1)$ of time, which can be very important for high dimensional problems.

16.5.3 Speeding Up the Calculus of the Activation Degrees through Reuse

After speeding up also the computation of output $\mathbf{y}$, the time required to compute the degrees of activation $\mathbf{w}_m$ might not be negligible at all. Again, we can think to adopt the same trick used in the previous subsections: vectors $\mathbf{w}_m$ of the parents can be stored and reused in the offspring. Thus, we could save $N \cdot F$ operations for each reused $\mathbf{w}_m$.

When we generate a TS fuzzy system, rules could have some *don't care* conditions in the antecedent part. These conditions reduce the complexity for estimating the vectors $\mathbf{w}_m$ when they cannot be reused. In particular, if we fix the maximum possible number of conditions different from *don't care* conditions to a value $\Gamma_{max} \leq F$, then the complexity associated with the calculus of $\mathbf{w}_m$ is $O(M \cdot N \cdot \Gamma_{max})$. The same complexity reduction occurs for indexes $\mathbf{i}_m$.

On the other hand, the authors in [23] had already observed that, fixed the number of rules, the introduction of the *don't care* conditions allows covering a larger portion of the input space. Indeed, when no *don't care* conditions are included, the fraction covered by each fuzzy rule is exponentially decreased by the increase in the dimensionality of the input space. Since fuzzy rules with several *don't care* conditions can cover a large portion of the input space, the overall input space can be covered by a small number of general fuzzy rules. When we fix Γ_{max} to small values, we force the identification of TS rules with a high number of *don't care* conditions. Thus, we achieve a higher coverage of the input space, which also corresponds to a lower MSE.

16.6 The Used MOEA to Learn TS Rules

We performed multi-objective genetic rule learning by using the (2+2)M-PAES, a modified version of the classical (2+2)PAES proposed in [28]. We have successfully used (2+2)M-PAES in several studies [9, 10], and we have shown its good behavior if compared with classical PAES and NSGA-II [10]. Unlike classical (2+2)PAES, which uses only mutation to generate new candidate solutions, (2+2)M-PAES exploits the one-point crossover and three appropriately defined mutation operators. We experimentally verified that crossover helps create an approximation of the Pareto front where solutions are uniformly distributed along the front [10]. Each solution is described through a chromosome composed of $M \cdot F$ natural numbers, obtained by concatenating rows of matrix $\mathbf{J}$ (consequent parameters are not included in the chromosome, since they are optimized on the fly).

Let c_1 and c_2 be two solutions. The one-point crossover operator cuts the chromosomes c_1 and c_2 at some chosen common gene and swaps the resulting subchromosomes. The common gene is chosen by extracting randomly a number in

$[M_{min}, \rho_{min}]$, where M_{min} is the minimum number of rules, which must be present in a rule base, and ρ_{min} is the minimum number of rules in c_1 and c_2, and multiplying this number by $(F+1)$.

The first mutation operator adds γ rules to the rule base, where γ is randomly chosen in $[1, \gamma_{max}]$. The upper bound γ_{max} is fixed by the user. If $\gamma + M > M_{max}$, where M_{max} is the maximum possible number of rules in a generated TS system, then $\gamma = M_{max} - M$. For each rule r_m added to the chromosome, we generate a random number t, $t \in [1, \Gamma_{max}]$, which indicates the number of input variables used in the antecedent of the rule. Then, we generate t natural random numbers between 1 and F to determine the input variables which compose the antecedent part of the rule. Finally, for each selected input variable f, we generate a random natural number $j_{m,f}$ between 1 and Q_f , which determines the fuzzy set $A_{f,j_{m,f}}$ to be used in the antecedent of rule r_m.

The second mutation operator randomly changes δ elements of matrix $\mathbf{J}$. The number δ is randomly generated in $[1, \delta_{max}]$. The upper bound δ_{max} is fixed by the user. For each element to be modified, a number is randomly generated in $[0, Q_f]$, where f is the input variable corresponding to the selected matrix element (when the element 0 is selected, the condition corresponds to *don't care*). The element is modified only if the constraint on the maximum number of input variables Γ_{max} for each rule is satisfied; otherwise, the element maintains its original value.

The mutation operator removes κ rules from the rule base, where κ is randomly chosen in $[1, \kappa_{max}]$. In the experiments, we used $\kappa_{max} = min(\phi_{max}, M - M_{min})$, where ϕ_{max} is fixed by the user, and M and M_{min} are, respectively, the number of rules of the individual and the minimum number of rules allowed for all individuals.

We start with two randomly generated solutions. At each iteration, the application of crossover and mutation operators produces two new solutions z_1 and z_2 from two solutions c_1 and c_2 randomly picked from the archive. If the archive contains a unique solution, c_1 and c_2 correspond to this unique solution. We experimentally verified that the random extraction of the current solutions from the archive allows us to extend the set of non-dominated solutions contained in the archive so as to obtain a better approximation of the Pareto front. In this paper, we have used two alternative stopping criteria, based on the number of epochs G and on elapsed time ET_{tot}, respectively. Figure 16.1 shows the flow-chart of the (2+2)M-PAES which uses the fast identification of TS-type FRBSs and the reuse. Sometimes in the following we will shortly call the fast method with reuse as "fast" method. When the reuse is not exploited, we will always refer to it as "fast with no reuse" method.

16.7 Experimental Results

We have compared the proposed technique on two datasets, namely, a regression problem and a chaotic time series forecasting problem. On each dataset, we have compared the performance of the classical identification, the fast identification with no reuse and the fast identification (with reuse) carrying out two experiments: the first was on equal execution times basis, while the second on equal optimization

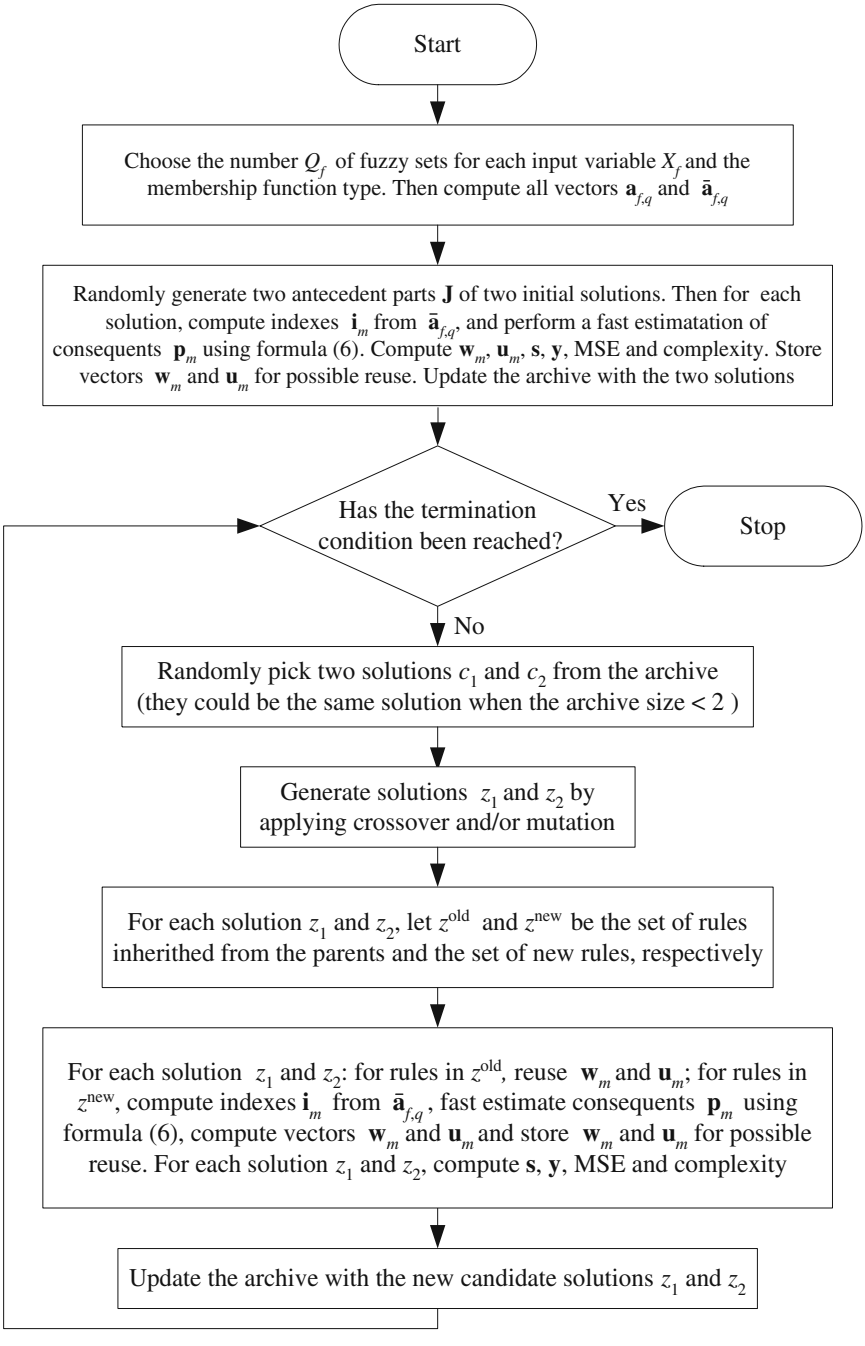


Fig. 16.1 Flow-chart of the (2+2)M-PAES which uses fast identification and reuse

epochs basis. Each of the four experiments has been repeated for eight times, changing the seed of the random number generator, thus causing the generation of different random training and test sets and different evolutions of the MOEA.

16.7.1 Regression Problem

As regards the regression problem, we have considered an artificial dataset, based on a benchmark used in [24]. Such benchmark is a three-variable real function defined as: $o = \left(1 + {X_1}^{0.5} + {X_2}^{-1} + {X_3}^{-1.5}\right)^2$. The training data are generated by uniformly sampling the input variables in $[1,6]^3$, while the testing data points are generated by uniformly sampling the input variables in $[1.5,5.5]^3$. In our experiments, we have added 7 input variables $X_4,\ldots,X_{10}$, uniformly sampled in the domains $[1,6]^7$ and $[1.5,5.5]^7$ for the training and test sets, respectively. Obviously, the seven added input variables constitute, to some extent, a sort of noise with respect to the function since the output o does not depend on them. We have generated $N = 50,000$ and $5,000$ training and test data patterns, respectively. We have used a number of fuzzy sets equal to 4 for all inputs ($Q_f = Q = 4$) and uniform partitions composed by gaussian membership functions.

We observe that for this dataset we are able to generate the most complete and non-redundant achievable TS system. The rule base of this TS system consists of all combinations of fuzzy sets belonging to the partitions of the only three input variables X_1, X_2, X_3 which influence the output. Let us denote such rule base as $\mathbf{J}_{64}$. $\mathbf{J}_{64}$ is composed of $4^3 = 64$ rules, with *don't care* conditions for all variables except for X_1, X_2, X_3. Obviously, the MSE computed for this TS system can be considered as a sort of lower bound, at least for the training set, and can provide us with some indication on how much the evolutionary process is far from optimal solutions. Table 16.1 shows the MSEs computed on the training and test sets on $\mathbf{J}_{64}$ rule base. In the following, we will show that the fast method is able to achieve very accurate solutions much faster than the classical method.

In the first experiment for the dataset at hand, we show how, on equal execution time, the fast method obtains Pareto fronts which dominate the ones obtained by the classical method. In the second experiment, we point out how, on equal number of epochs, the fast method achieves Pareto fronts comparable with the ones obtained by the classical method, but saving approximately 90% of time.

In both the experiments we have used, as objectives, the total number of conditions different from *don't care* conditions as a measure of complexity, and the MSE

Table 16.1 MSE for $\mathbf{J}_{64}$ rule base

	MSE
Training set	0.117
Test set	0.084

Table 16.2 Parameters used in the (2+2)M-PAES execution for the regression problem

Parameter	Value
archive size	50
δ_{max}	10
γ_{max}	20
M_{min}	1
M_{max}	64
Γ_{max}	3
ϕ_{max}	5
crossover probability	1
mutation probability	0.4

as a measure of accuracy. Table 16.2 summarizes the parameters used for the execution of the (2+2)M-PAES. As regards mutation probability, the first, second and third mutation operators are applied with 0.8, 0.05 and 0.15 probabilities, when the mutation is applied. Here, there is a bias towards rule adding, since the probability to generate a solution which will be added to the archive is higher when removing than when adding rules.

16.7.1.1 Comparisons under Equal Execution Times on the Regression Problem

In the first experiment, we have used the execution time as stopping criterion. We have set the maximum amount of time to 1800 sec (30 min). We have repeated the experiment for eight trials using different randomly extracted training and test sets and different MOEA evolutions. Figures 16.2 and 16.3 show the trend of the Pareto fronts after approximately 450, 900, 1350, and 1800 seconds on the training set for the classical (figure on the left) and fast (figure on the right) methods for two randomly selected trials.

For the sake of brevity, we do not show the trends for all the trials. On the other hand, these trends are similar to the ones reported in Figs. 16.2 and 16.3. First of all, we can observe that, independently of the trial, the fast method executes a much higher number of epochs on equal time, thus achieving good Pareto fronts. Actually, at each epoch considered in the figures, the approximated Pareto fronts obtained by the fast method dominate the ones achieved by the classical approach. Further, the MSEs of the most accurate solutions generated by the fast method are quite close to the MSEs shown in Table 16.1, thus suggesting that the fast method achieves very good MSEs just after 1800 seconds. On the other hand, the MSEs of the most accurate solutions generated by the classical method are quite far from the MSEs shown in Table 16.1. Finally, we observe that the intervals of complexity of the Pareto fronts generated by the classical method are much wider than the ones generated by the fast method. This is quite normal since we start the execution of the (2+2)M-PAES from two solutions that contain the maximum number of rules and conditions.

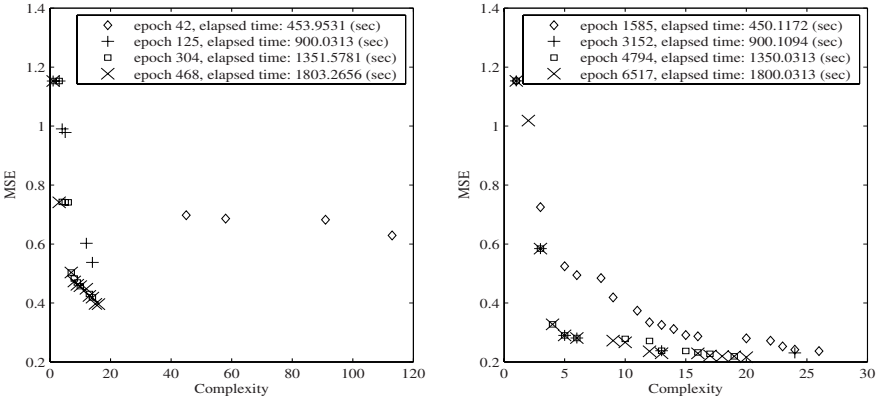


Fig. 16.2 Trends of the approximated Pareto fronts obtained within 30 minutes on the training set using classical (left) and fast (right) methods for a sample trial (regression problem)

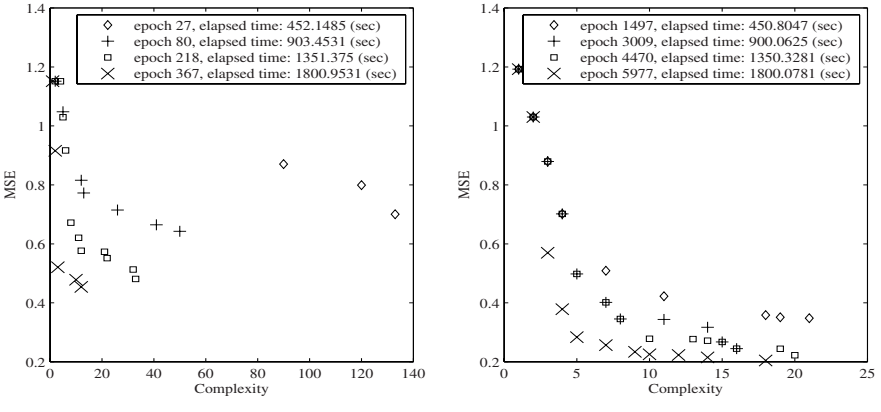


Fig. 16.3 Trends of the approximated Pareto fronts obtained within 30 minutes on the training set using classical (left) and fast (right) methods for another sample trial (regression problem)

During the evolutionary process, the complexity of the rules decreases, as testified by the fast method and by the second experiment on the dataset at hand. Figure 16.4 shows the final Pareto fronts achieved on the test set by the classical (figure on the left) and fast (figure on the right) methods on all the eight trials. We can observe that the Pareto fronts obtained by the fast method outperform the Pareto fronts obtained by the classical method, thus testifying the good generalization properties of the TS systems generated by the fast method.

Table 16.3 shows the average results obtained by the classical method and the fast method executed both without exploiting and by exploiting reuse. Here, $\overline{G}$ is the average number of epochs on the eight trials, $\overline{M}_{tot}$ is the average total number

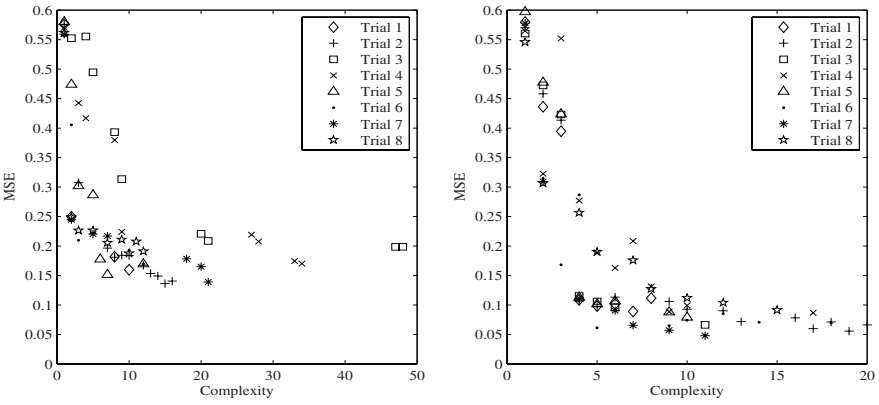


Fig. 16.4 Final Pareto fronts for each trial on the test set obtained using classical (left) and fast (right) methods after 30 minutes (regression problem)

Table 16.3 Results averaged on the eight trials after 30 minutes on the regression problem

	$\overline{G}$	$\overline{M}_{tot}$	$\overline{ET}_{rule}(sec)$	$\overline{MSE}_{TR}^{best}$	$\overline{MSE}_{TS}^{best}$
Classical	466.0	10,309.5	0.17475	0.43340	0.16404
Fast with no reuse	1,950.5	45,566.4	0.03777	0.28819	0.08163
Fast (with reuse)	6,066.6	115,390.4	0.01375	0.25273	0.07209

of evaluated rules (which corresponds to the total number of consequent parameters estimated) on the eight trials, $\overline{ET}_{rule}$ is the average elapsed time per rule, computed as the total elapsed time divided by the total number of rules (and averaged over the eight trials), $\overline{MSE}_{TR}^{best}$ and $\overline{MSE}_{TS}^{best}$ are, respectively, the average lowest MSEs obtained in the final fronts on training and test sets, respectively. We can observe that the fast identification of the consequent parameters allows increasing the average number of rules generated and evaluated during the 30 minutes from 10,309.5 of the classical method to 45,566.4 of the fast method. Further, the adoption of the reuse increases this average number of rules until 115,390.4. The tangible result of this decrease in time needed to generate and evaluate the rules is the increase in accuracy achieved by the fast method thanks to the higher number of epochs. Concluding, this experiment has pointed out how speeding up the generation and evaluation of the TS systems allows executing a larger number of epochs, thus improving accuracy of the solutions.

16.7.1.2 Comparisons under Equal Number of Epochs on the Regression Problem

In the second experiment on the regression dataset, we have used the number of epochs as stopping criterion. We have set the maximum number G of epochs to 2500

and have repeated the experiment for eight trials using different randomly extracted training and test sets. Figures 16.5 and 16.6 show the trend of the Pareto fronts after 625, 1250, 1875, and 2500 epochs on the training set for the classical (figure on the left) and fast (figure on the right) methods and for two randomly selected trials, respectively. We can observe that, independently of the trial, the accuracies of the solutions in the Pareto fronts obtained by the classical and the fast methods are quite similar. This points out that the use of the fast method does not affect the performance of the generated TS. After the same number of epochs the solutions in the corresponding Pareto fronts do not differ from each other considerably. Of course, the Pareto fronts of the fast method have been obtained in a much shorter time.

Figure 16.7 shows the final Pareto fronts achieved on the test sets of the eight trials by the classical (figure on the left) and fast (figure on the right) methods. We can observe that the Pareto fronts obtained by the fast method and by the classical method are comparable, thus again testifying the good generalization properties of the TS systems generated by the fast method.

Table 16.4 shows the average results obtained by the classical method and the fast method executed both without exploiting and by exploiting reuse. Here, $\overline{ET}_{tot}$ is the average elapsed time after 2500 epochs. We can observe that the fast identification of the consequent parameters allows considerably reducing the average elapsed times from 6580 seconds of the classical method to 2166 seconds of the fast method with no reuse. Further, the adoption of the reuse decreases the average elapsed time until 643. It is interesting to observe that the average MSE of the best solutions achieved by the fast method is lower than the average MSE of the best solutions generated by the classical method, thus further testifying that the techniques proposed to speed up the fitness computation do not affect the accuracies of the final solutions. Concluding, this second experiment has pointed out how speeding up the generation and evaluation of the TS systems allows reducing the execution times without deteriorating the accuracy of the solutions of an amount of $\cong 90\%$.

16.7.2 Time Series Forecasting Problem

As regards the time series forecasting problem we have generated a chaotic time series by the Mackey-Glass delayed differential equation [29]:

$$\frac{dx(t)}{dt} = \frac{a \cdot x(t-\tau)}{1+x(t-\tau)^{10}} - b \cdot x(t).$$

In our experiments, we used $a = 0.2$, $b = 0.1$ and $\tau = 17$, as generally done (see, e.g., [24]). We considered the time series at discrete temporal times, with time step $\Delta t = 0.01$ ($x[k] = x(t = k \cdot \Delta t)$), for 67,499 integration steps ($k = 1, \ldots, 67{,}499$). The value of the discrete series has been obtained by using the 4^{th} order Runge-Kutta method:

$$x[k+1] = x[k] + \frac{K_1}{6} + \frac{K_2}{3} + \frac{K_3}{6} + \frac{K_4}{6}$$

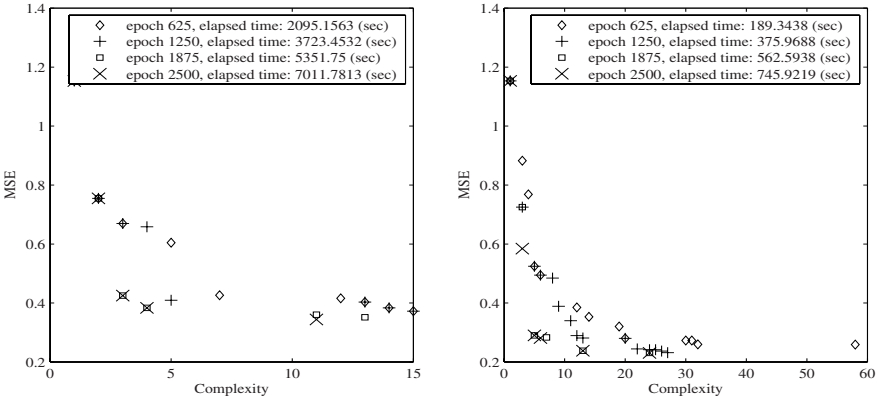


Fig. 16.5 Approximated Pareto fronts obtained on the training set using classical (left) and fast (right) methods after 625, 1250, 1875, and 2500 epochs for a sample trial (regression problem)

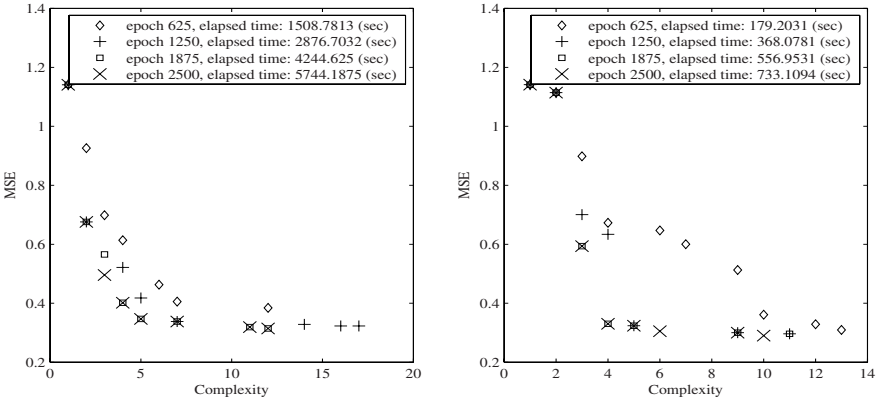


Fig. 16.6 Approximated Pareto fronts obtained on the training set using classical (left) and fast (right) methods after 625, 1250, 1875, and 2500 epochs for another sample trial (regression problem)

where K_1, K_2, K_3 and K_4 are the Runge-Kutta coefficients. The method has been applied starting from an initial condition x[0] randomly generated within the interval [0, 1] and considering x(t) = 0 for all $t < 0$.

Once the 67,499 samples have been generated, according to [24] we have removed the first 10,099 data points to avoid the transient portion of the data, thus obtaining 57,400 values. From these values, we have generated a dataset of 55,000 samples of the format $(x[k-1800], x[k-1200], x[k-600], x[k], x[k+600])$ in order to predict $x[k+600]$ from the past values $x[k-1800]$, $x[k-1200]$, $x[k-600]$, and $x[k]$ (2400 samples have been discarded to completely separate the test set from the training set). The first 50,000 samples have been used for training while the

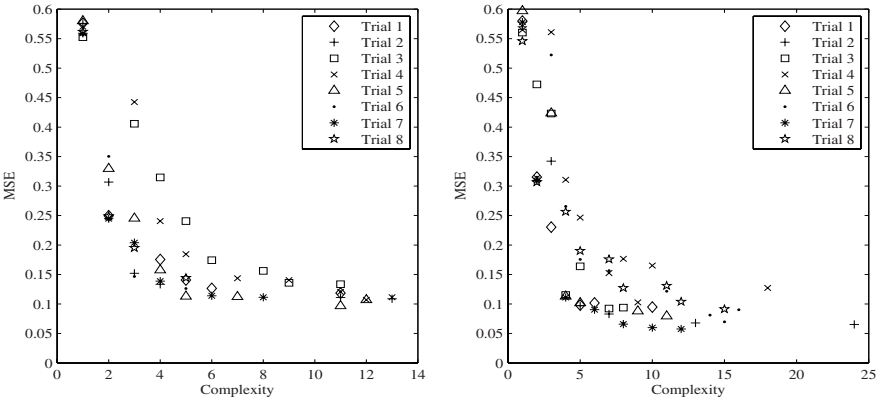


Fig. 16.7 Final Pareto fronts for each trial on the test set obtained using classical (left) and fast (right) methods after 2500 epochs (regression problem)

Table 16.4 Results averaged on the eight trials after 2500 epochs on the regression problem

Method	$\overline{ET}_{tot}(sec)$	$\overline{M}_{tot}$	$\overline{ET}_{rule}(sec)$	$\overline{MSE}_{TR}^{best}$	$\overline{MSE}_{TS}^{best}$
Classical	6,580	40,359	0.16306	0.3346	0.1180
Fast with no reuse	2,166	55,497	0.03904	0.2753	0.0762
Fast (with reuse)	643	55,497	0.01159	0.2753	0.0762

remaining 5,000 for test. The problem at hand can be viewed as a regression problem of the form $o = f(X_1, X_2, X_3, X_4)$, where $X_1 = x[k-1800]$, $X_2 = x[k-1200]$, $X_3 = x[k-600]$, $X_4 = x[k]$, and $o = x[k+600]$.

Unlike in [24], we have added six more input variables $X_5, \ldots, X_{10}$, uniformly sampled in the domains $[0,1]^6$ for the training and test sets, in order to make the dataset more difficult to deal with, thus obtaining the problem: $o = f(X_1, \ldots, X_{10})$.

Again, the six added input variables constitute a sort of noise with respect to the function since the output does not depend on them. We have used a number of fuzzy sets equal to seven for all inputs ($Q_f = 7$) and uniform partitions composed by gaussian membership functions. Such dataset is similar to the one used in [24], but with some differences, namely: a higher sampling rate (1 instead of 0.01), a smaller integration step (0.01 instead of 0.1), a higher number of fuzzy sets on each input variable (7 instead of 2) and the presence of the six added fictitious input variables. As shown in the following, we have achieved good MSEs, though higher than that found in [24] by Jang using the well-known ANFIS neuro-fuzzy system with 4 inputs, 2 fuzzy sets per input, 16 rules and 64 conditions (*don't care* is not used therein). The reasons are the following: i) we used a bigger dataset (and thus more difficult to deal with), having both a higher number of samples and a higher number of inputs, ii) we limit ourselves to consider systems with lower complexity (only one condition per rule instead of four and a maximum of 30 conditions in total instead of

64), iii) we do not perform membership functions optimization. On the other hand, it has to be said that it would require too much time to run the ANFIS system with membership function optimization on the problem at hand even to perform only one iteration step.

Above we have explained how to generate training and test sets for the time series forecasting problem. Thus, we are now ready to carry out experiments. Here we have repeated the two experiments carried out on the regression dataset on the problem at hand. Again, in the first experiment, we show how, on equal execution time, the fast method with reuse obtains Pareto fronts which dominate the ones obtained by the classical method. In the second experiment, we point out how, on equal number of epochs, the fast method with reuse achieves Pareto fronts comparable with the ones obtained by the classical method, but saving approximately 96.5% of time.

The objective functions used here are the same as those used for the regression problem (number of conditions and MSE). The parameters used for the execution of the (2+2)M-PAES are the same as those shown in Table 16.2, with the exception of the maximum number of rules M_{max}, which has been set to 30 instead of 64, and the maximum number of conditions per rule Γ_{max} (1 instead of 3). As regards mutation probability, we used the same used for the regression problem on both the experiments.

16.7.2.1 Comparisons under Equal Execution Times on the Time Series Forecasting Problem

Here we repeated the first experiment carried out on the regression problem. Thus we have used the execution time as stopping criterion and we have set the maximum amount of time to 1800 sec. We have repeated the experiment for eight times using different randomly extracted training and test sets (by using a different starting point x[0]). Figures 16.8 and 16.9 show the trend of the Pareto fronts after approximately 450, 900, 1350, and 1800 seconds on the training set for the classical (figure on the left) and fast (figure on the right) methods and for two out of the eight trials, randomly selected. We can observe that, independently of the trial, the fast method executes a much higher number of epochs on equal time, thus achieving better Pareto fronts even for this problem.

Figure 16.10 shows the final Pareto fronts achieved on the test set by the classical (figure on the left) and fast (figure on the right) methods. We can observe that the Pareto fronts obtained by the fast method outperform the Pareto fronts obtained by the classical method, thus testifying the good generalization properties of the TS systems generated by the fast method.

Table 16.5 shows the average results obtained by the classical method and the fast method executed both without exploiting and by exploiting reuse. We can observe that the fast identification of the consequent parameters allows increasing the average number of rules generated and evaluated during the 30 minutes from 10,706.9 of the classical method to 38,337.3 of the fast method. Further, the adoption of the reuse increases this average number of rules until 76,670.9. The tangible result of this decrease in time needed to generate and evaluate the rules is the increase in

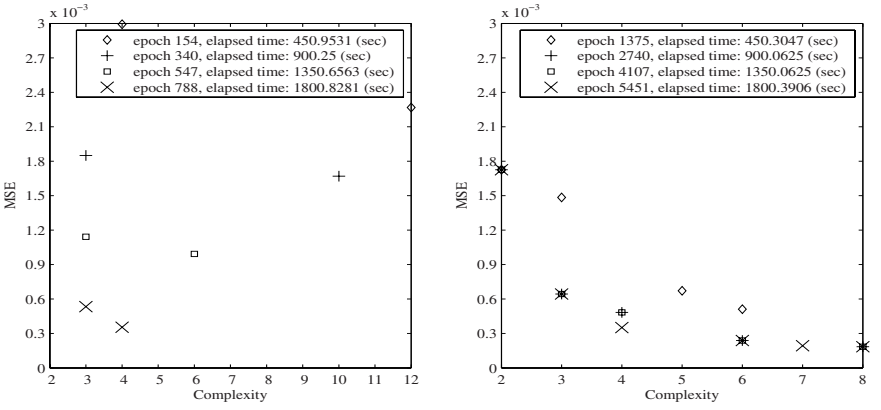


Fig. 16.8 Trends of the approximated Pareto fronts obtained within 30 minutes on the training set using classical (left) and fast (right) methods for a sample trial (time series forecasting problem)

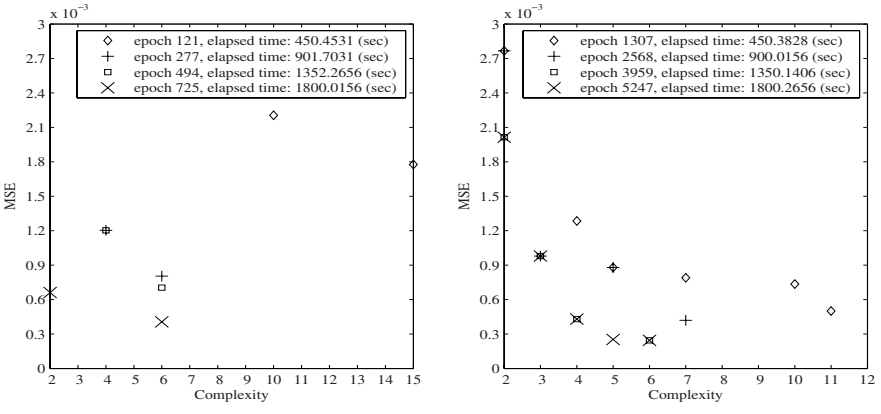


Fig. 16.9 Trends of the approximated Pareto fronts obtained within 30 minutes on the training set using classical (left) and fast (right) methods for another sample trial (time series forecasting problem)

accuracy achieved by the fast method thanks to the higher number of epochs. Concluding, this experiment has pointed out how speeding up the generation and evaluation of the TS systems allows executing a larger number of epochs, thus improving accuracy of the solutions.

16.7.2.2 Comparisons under Equal Number of Epochs on the Time Series Forecasting Problem

In this experiment, we have used the number of epochs as stopping criterion. We have set the maximum number G of epochs to 2500 and have repeated the

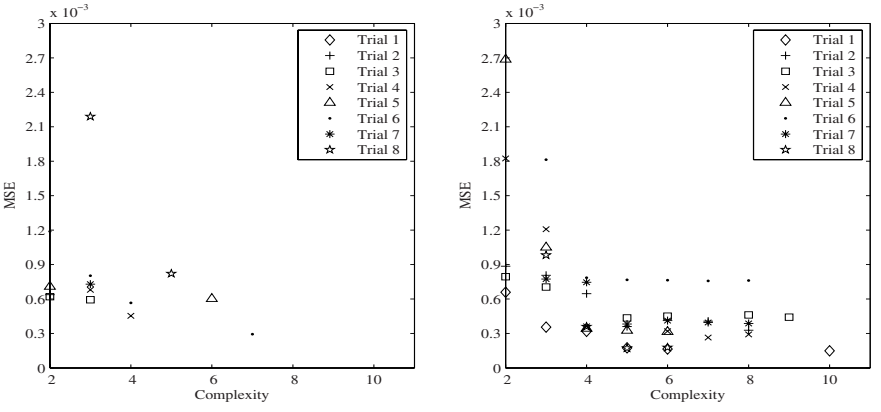


Fig. 16.10 Final Pareto fronts for each trial on the test set obtained using classical (left) and fast (right) methods after 30 minutes (time series forecasting problem)

Table 16.5 Results averaged on the eight trials after 30 minutes on the time series forecasting problem

	$\overline{G}$	$\overline{M}_{tot}$	$\overline{ET}_{rule}(sec)$	$\overline{MSE}_{TR}^{best}$	$\overline{MSE}_{TS}^{best}$
Classical	837.3	10,706.9	0.16503	$0.487 \cdot 10^{-3}$	$0.577 \cdot 10^{-3}$
Fast with no reuse	2,638.4	38,337.3	0.04387	$0.288 \cdot 10^{-3}$	$0.383 \cdot 10^{-3}$
Fast (with reuse)	5,265.3	76,670.9	0.02076	$0.239 \cdot 10^{-3}$	$0.350 \cdot 10^{-3}$

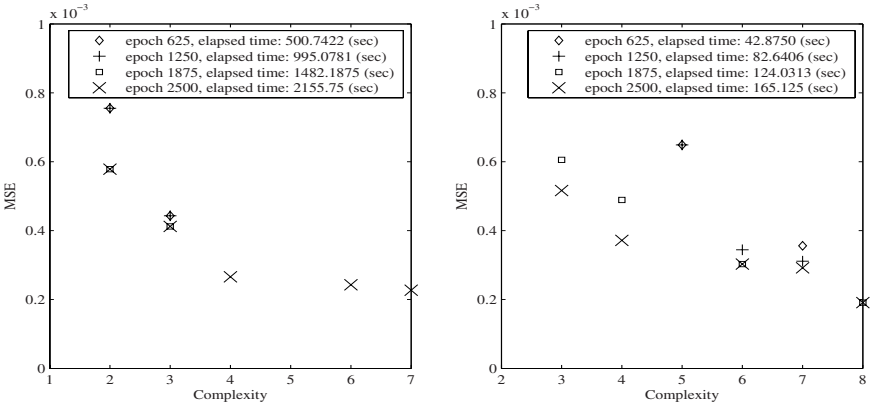


Fig. 16.11 Approximated Pareto fronts obtained on the training set using classical (left) and fast (right) methods after 625, 1250, 1875, and 2500 epochs for a sample trial (time series forecasting problem)

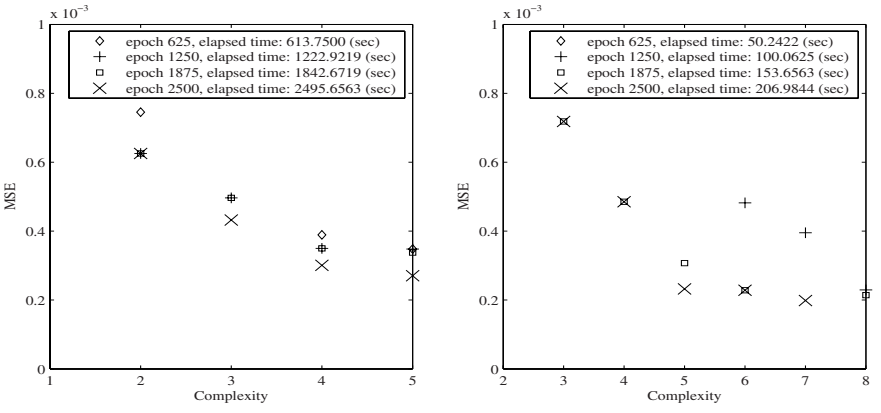


Fig. 16.12 Approximated Pareto fronts obtained on the training set using classical (left) and fast (right) methods after 625, 1250, 1875, and 2500 epochs for another sample trial (time series forecasting problem)

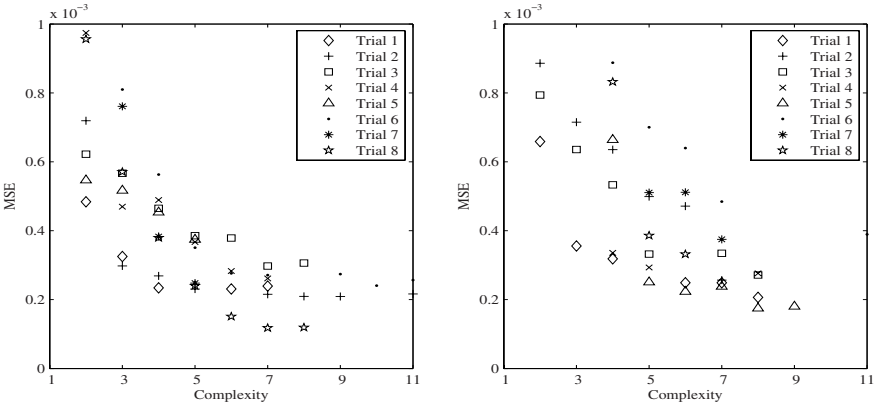


Fig. 16.13 Final Pareto fronts for each trial on the test set obtained using classical (left) and fast (right) methods after 2500 epochs (time series forecasting problem)

Table 16.6 Results averaged on the eight trials after 2500 epochs on the time series forecasting problem

Method	$\overline{ET}_{tot}(sec)$	$\overline{M}_{tot}$	$\overline{ET}_{rule}(sec)$	$\overline{MSE}_{TR}^{best}$	$\overline{MSE}_{TS}^{best}$
Classical	2,562.7	15,837.5	0.16181	$0.241 \cdot 10^{-3}$	$0.247 \cdot 10^{-3}$
Fast with no reuse	1,682.0	38,337.3	0.04387	$0.288 \cdot 10^{-3}$	$0.383 \cdot 10^{-3}$
Fast (with reuse)	89.1	17,286.6	0.00515	$0.189 \cdot 10^{-3}$	$0.285 \cdot 10^{-3}$

experiment for eight trials using different randomly extracted training and test sets and different MOEA evolutions. Figures 16.11 and 16.12 show the trend of the Pareto fronts after 625, 1250, 1875, and 2500 epochs on the training set for the classical (figure on the left) and fast (figure on the right) methods and for two out of the eight trials.

In Fig. 16.13 (which provides the final Pareto fronts achieved in the test set on each of the eight trials for classical and fast methods) we can observe that, even if the classical method seems slightly better, we can consider the average Pareto fronts approximatively equivalent. Of course, the Pareto fronts of the fast method have been obtained in a much shorter time even in this case.

Table 16.6 shows the average results obtained by the classical method and the fast method executed both without exploiting and by exploiting reuse. We can observe that the fast identification of the consequent parameters allows considerably reducing the average elapsed times from 2562.7 seconds of the classical method to 1682.0 seconds of the fast method with no reuse. Moreover, the adoption of the reuse further decreases the average elapsed time until 89.1. Thus, using the fast method we have been able to save $\cong 96.5\%$ of the time. It is interesting to observe that the average MSE of the best solutions achieved by the fast method is lower than the average MSE of the best solutions generated by the classical method on the training set, while they are comparable on the test set.

Even for the time series forecasting problem we can conclude that speeding up the generation and evaluation of the TS systems allows reducing the execution times without significantly deteriorating the accuracy of the solutions.

16.8 Conclusions

In this chapter, we have shown a possible roadmap towards the efficient design of multi-objective genetic Takagi-Sugeno fuzzy systems for high dimensional problems. We have proposed a method to speed up the identification of the consequent parameters of the TS rules. This method produces as a side-effect a decoupling of the rules. Thus, during the evolutionary process possible modifications in a rule do not affect the other rules and therefore we can avoid re-estimating parameters for all the rules which are not modified. Exploiting this observation, we have discussed how simply storing and reusing previously computed parameters we can further speed up the evolutionary process. In the experimental part we have shown the advantages of applying the efficient approach proposed in this chapter by using both regression and time series forecasting problems. Results have highlighted that, on average, the approach allowed saving approximately 90% and 96.5% of the execution times, respectively. When the execution times are the same, the proposed approach performs a significantly higher number of epochs and thus it better explores the search space, providing better Pareto fronts.

References

1. Angelov, P.P., Filev, D.P.: An approach to online identification of Takagi-Sugeno fuzzy models. IEEE Trans. on Systems, Man and Cybernetics: part B Cyb. 34(1), 484–498 (2004)
2. Babuska, R.: Fuzzy modeling for control. Kluwer Academic Publishers, Boston (1998)
3. Botta, A., Lazzerini, B., Marcelloni, F.: Context adaptation of Mamdani fuzzy rule-based systems. International Journal of Intelligent Systems 23(4), 397–418 (2008)
4. Botta, A., Lazzerini, B., Marcelloni, F., Stefanescu, D.C.: Context adaptation of fuzzy systems through a multi-objective evolutionary approach based on a novel interpretability index. Soft Computing 13(5), 437–449 (2009)
5. Branke, J., Schmeck, H., Deb, K.: Parallelizing multi-objective evolutionary algorithms: Cone separation. In: Proc. of the IEEE Congress on Evolutionary Computation 2004 - CEC 2004, Portland, Oregon, USA, June 19-23, pp. 1952–1957 (2004)
6. Bui, L.T., Abbass, H.A., Essam, D.: Fitness inheritance for noisy evolutionary multi-objective optimization. In: Proc. of the 2005 Conference on Genetic and Evolutionary Computation, Washington, D.C., USA, June 25-29, pp. 779–785 (2005)
7. Chen, C.-H., Hong, T.-P., Tseng, V.S., Chen, L.-C.: A multi-objective genetic-fuzzy data mining algorithm. In: Proc. of the IEEE International Conference on Granular Computing, Hangzhou, China, August 26-28, pp. 115–120 (2008)
8. Cococcioni, M., Corsini, G., Lazzerini, B., Marcelloni, F.: Approaching the ocean color problem using fuzzy rules. IEEE T. on Syst., Man & Cyb. - part B: Cyb. 34(3), 1360–1373 (2004)
9. Cococcioni, M., Corsini, G., Lazzerini, B., Marcelloni, F.: Solving the ocean color inverse problem by using evolutionary multi-objective optimization of neuro-fuzzy systems. International Journal of Knowledge-Based and Intelligent Engineering Systems 12(5-6), 339–355 (2008)
10. Cococcioni, M., Ducange, P., Lazzerini, B., Marcelloni, F.: A Pareto-based multi-objective evolutionary approach to the identification of Mamdani fuzzy systems. Soft Computing 11(11), 1013–1031 (2007)
11. Cococcioni, M., Lazzerini, B., Marcelloni, F.: Fast multiobjective genetic rule learning using an efficient method for Takagi-Sugeno fuzzy systems identification. In: Proc. of the 8th Int. Conference on Hybrid Intelligent Systems (HIS 2008), Barcelona, Spain, pp. 272–277 (2008)
12. Cococcioni, M.: The Evolutionary Multiobjective Optimization of Fuzzy Rule-Based Systems Bibliography Page (2009), http://www2.ing.unipi.it/~g000502/emofrbss.html
13. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. Journal of Symbolic Computation 9, 251–280 (1990)
14. Ducheyne, E.I., De Baets, B., De Wulf, R.R.: Fitness inheritance in multiple objective evolutionary algorithms: A test bench and real-world evaluation. Appl. Soft Comp. 8, 337–349 (2008)
15. Getreuer, P.: Writing fast Matlab code (2009), http://www.mathworks.com/matlabcentral/fileexchange/5685
16. Herrera, F.: Genetic fuzzy systems: Taxonomy, current research trends and prospects. Evolutionary Intelligence 1, 27–46 (2008)
17. Huang, X., Pan, V.Y.: Fast rectangular matrix multiplication and applications. Journal of Complexity 14, 257–299 (1998)
18. Ishibuchi, H.: Multiobjective genetic fuzzy systems: review and future research directions. In: Proc. of Fuzz-IEEE 2007, London, UK, July 23-26, pp. 1–6 (2007)

19. Ishibuchi, H., Murata, T., Turksen, I.B.: Selecting linguistic classification rules by two-objective genetic algorithms. In: Proc. of the 1995 IEEE International Conference on System, Man and Cybernetics, Vancouver, BC, Canada, vol. 2, pp. 1410–1415 (1995)
20. Ishibuchi, H., Murata, T., Turksen, I.B.: Single-objective and two-objective genetic algorithms for selecting linguistic rules for pattern classification problems. Fuzzy Sets and Systems 89(2), 135–150 (1997)
21. Ishibuchi, H., Nakashima, T., Murata, T.: Three-objective genetics-based machine learning for linguistic rule extraction. Information Sciences 136(1-4), 109–133 (2001)
22. Ishibuchi, H., Nojima, Y.: Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning. Int. J. of Appr. Reas. 4(1), 4–31 (2007)
23. Ishibuchi, H., Yamamoto, T.: Interpretability issues in fuzzy genetics-based machine learning for linguistic modelling. In: Lawry, J., Shanahan, J.G., Ralescu, A.L. (eds.) Modelling with Words. LNCS, vol. 2873, pp. 209–228. Springer, Heidelberg (2003)
24. Jang, J.-S.R.: ANFIS: adaptive-network-based fuzzy inference system. IEEE Trans. on Systems, Man and Cybernetics 23(3), 665–685 (1993)
25. Jin, Y.: Fuzzy modeling of high-dimensional systems: Complexity reduction and interpretability improvement. IEEE Trans. on Fuzzy Systems 8(2), 212–223 (2000)
26. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing 9, 3–12 (2005)
27. Knowles, J.D.: ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Trans. on Evol. Comp. 10(1), 50–66 (2006)
28. Knowles, J.D., Corne, D.W.: Approximating the non dominated front using the Pareto archived evolution strategy. Evolutionary Computation 8(2), 149–172 (2000)
29. Mackey, M.C., Glass, L.: Oscillation and chaos in physiological control systems. Science 197, 287–289 (1977)
30. Nauck, D.D.: GNU Fuzzy. In: Proc. of FUZZ-IEEE 2007, London, UK, pp. 1–6 (2007)
31. Schraudolph, N.N.: A fast, compact approximation of the exponential function. Neural Computation 11, 853–862 (1999)
32. Soukkou, A., Khellaf, A., Leulmi, S.: Multiobjective optimisation of robust Takagi-Sugeno fuzzy neural controller with hybrid learning algorithm. Int. Journal of Modelling, Identification and Control 2(4), 332–346 (2007)
33. Streichert, F., Ulmer, H., Zell, A.: arallelization of Multi-objective Evolutionary Algorithms Using Clustering Algorithms. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 92–107. Springer, Heidelberg (2005)
34. Tan, K.C., Yang, Y.J., Goh, C.K.: A distributed Cooperative coevolutionary algorithm for multiobjective optimization. IEEE Trans. on Evolutionary Computation 10(5), 527–549 (2006)
35. Van Veldhuizen, D.A., Zydallis, J.B., Lamont, G.B.: Considerations in engineering parallel multiobjective evolutionary algorithms. IEEE Trans. on Evol. Comp. 7(2), 144–173 (2003)
36. Yen, J., Gillespie, L.W., Gillespie, C.W.: Improving the interpretability of TSK fuzzy models by combining global learning and local learning. IEEE Trans. on Fuzzy Syst. 6(4), 530–537 (1998)

Chapter 17
Evolutionary Algorithms for the Multi Criterion Minimum Spanning Tree Problem

Madeleine Davis-Moradkhan and Will Browne

Abstract. In many real world network problems several objectives have to be optimized simultaneously. To solve such problems, it is often appropriate to use the multi-criterion minimum spanning tree (MCMST) model, a combinatorial optimization problem that has been shown to be NP-Hard. In Pareto Optimization of the model no polynomial time algorithm is known to find the Pareto front for all instances of the MCMST problem. Researchers have therefore developed deterministic and evolutionary algorithms. However, these exhibit a number of shortcomings such as lack of scalability and large CPU times. Therefore, the hybridised Knowledge-based Evolutionary Algorithm (KEA) has been proposed, which does not have the limitations of previous algorithms because of its speed, its scalability to more than 500 nodes in the bi-criterion case and scalability to the multi-criterion case, and its ability to find both the supported and non-supported optimal solutions. KEA is faster and more efficient than NSGA-II in terms of spread and number of solutions found. The only weakness of KEA is the dominated middle of its Pareto front. In order to overcome this deficiency, a number of modifications have been tested including KEA-M, KEA-G and KEA-W. Experimental results show that when time is expensive KEA is preferable to all other algorithms tested.

17.1 Introduction

The multi-criterion minimum spanning tree (MCMST) problem is a combinatorial optimization problem that has attracted attention in recent years due to its applications in many real world problems, in particular in designing networks (computer,

Madeleine Davis-Moradkhan
School of Systems Engineering, University of Reading, Pepper Lane, Reading RG6 6AY, UK
e-mail: mdmtig@yahoo.co.uk

Will Browne
Senior Lecturer, Victoria University of Wellington, New Zealand
e-mail: will.browne@ecs.vuw.ac.nz

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 423–452.
springerlink.com

communication, electrical, pipeline, etc). This problem is also interesting for theoretical reasons, as it has been shown to be NP-Hard [1]. Existing algorithms applied to this problem exhibit a number of shortcomings such as lack of scalability and large CPU times. A 500-node MCMST problem has a search space of 500^{498} feasible solutions, which account for the difficulty of finding good solutions in a practical time.

17.1.1 Problem Definition

Given a connected, undirected and labeled graph $G = (N,E)$, where N is the finite set of n labeled nodes and E is the finite set of m edges linking the nodes, suppose there are p positive real numbers associated with each edge representing costs denoted by $c_=(c_{1j},c_{2j},\cdots,c_{pj})$, for $j = 1,2,\cdots,m$. Thus c_{ij}, represents the $i^{th.}$ cost associated with edge j for $i = 1,2,\cdots,p$, and $C = [c_{ij}]$ is the associated costs matrix.

A *spanning tree* $T = (N,B)$ is a connected sub graph on the n nodes of G with no cycles, where $B \subset E$ and $|B| = n-1$. Adding one more edge of G to B creates one and only one cycle; and removing any one edge causes disconnection. G has n^{n-2} spanning trees if it is complete.

Let $x = (x_1,x_2,\cdots,x_m)$, where $x_j = 1$ if edge e_j is present in a given spanning tree (ST), and $x_j = 0$ otherwise. The vector x is the characteristic vector, and each spanning tree on G can be expressed by one such vector x. Let X be the set of characteristic vectors corresponding to all the spanning trees in G. The MCMST problem can be represented by the following model [50], where $z_i(x)$ is the $i^{th.}$ objective to be minimized and is expressed as the sum of the corresponding costs of the edges forming the spanning tree.

$$
\begin{aligned}
minz_1(x) &= \sum_{j=1}^{m} c_{1j}x_j, \\
minz_2(x) &= \sum_{j=1}^{m} c_{2j}x_j, \\
&\cdots, \\
minz_p(x) &= \sum_{j=1}^{m} c_{pj}x_j
\end{aligned}
$$

subject to $x \in X$.

Figure 17.1 shows the evaluated graph G on five nodes and two of its spanning trees, each minimizing one of the criteria. The numbers in brackets indicate the values of the two costs (criteria) corresponding to each edge of G.

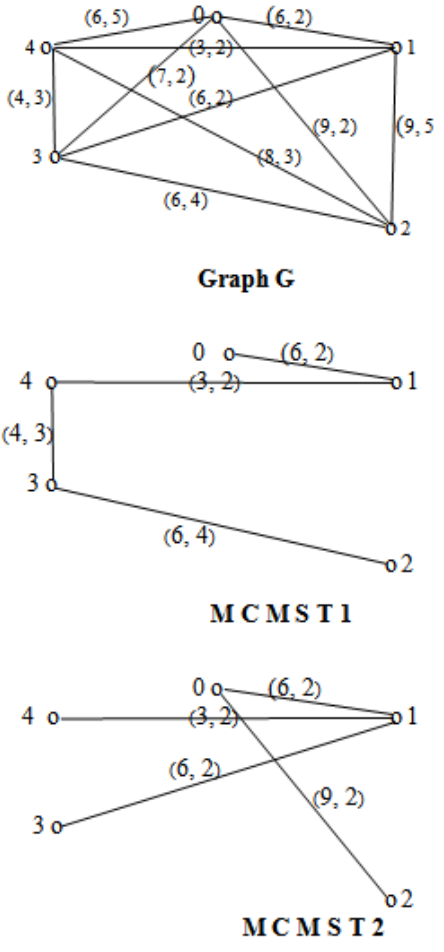


Fig. 17.1 Graph G and two of its spanning trees, which are the extreme points of the Pareto front, each minimizing one of the criteria

17.1.2 Solution Approaches

The MCMST problem admits very rarely a single solution particularly when objectives are conflicting. It is possible to determine and assign a weight w_i to every objective function $z_i(x) \forall i$, where the weights reflect the preferences of the decision maker on each objective. Although the problem can be reduced to minimizing a scalarized objective function, this approach has certain drawbacks and limitations [9], such as the need to fix the relative importance of the objective functions [43].

In Pareto Optimization, also called vector optimization, all criteria are considered equally important. A solution x^* is efficient, non-dominated or Pareto optimal if

there does not exist another solution x such that: $z_i(x) \leq z_i(x^*), \forall i = 1, \cdots, p$, and $z_k(x) < z_k(x^*)$, for at least one k.

The set of all Pareto optimal solutions is called the efficient set or the Pareto front (PF) [9]. There are two types of efficient solution [42], as shown in Figure 17.2. Formally 'the supported efficient solutions are minima for a convex combination of criteria. The non-supported efficient solutions cannot be computed by minimizing a convex combination of criteria, whatever the considered strictly positive weights' [14]. The supported efficient solutions are situated on the convex hull of the feasible region, and are relatively easy to find. The non-supported efficient solutions, situated in the segment formed by two consecutive supported solutions and the corresponding local nadir point , are difficult to discover and most algorithms do not find them. This is the case in the weighted-sum methods where the optimal solutions found by varying the weights associated with each criterion are all supported efficient solutions.

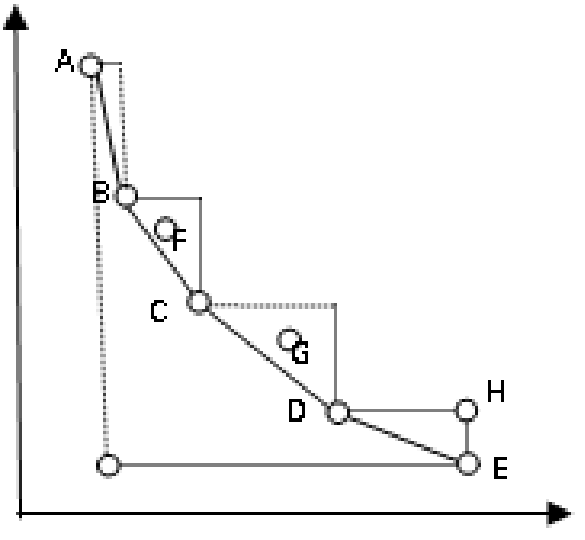


Fig. 17.2 Points A, B, C, D and E are the supported efficient solutions. H is the local nadir point corresponding to D and E. All the points situated inside a segment, for example segment DEH, are non-supported efficient solutions. Thus, points F and G are non-supported efficient solutions

No polynomial time algorithm is known to find the PF for all instances of the MCMST problem. For small complete graphs $(n \leq 10)$ exhaustive search has been used to find the true PF [25]. However, the exhaustive search is impractical with the present speed of computers for complete graphs having more than 10 or 11 nodes. Authors have therefore turned to deterministic heuristics and evolutionary algorithms that succeed only in calculating an *approximate Pareto set (APS)*. Most

of the existing algorithms are applicable to the bi-objective case and have been tested using simple or planar graphs with few edges.

A number of deterministic algorithms for the bi-criterion case have been proposed for use in decision aid or for validating evolutionary algorithms , all of which are restricted to two criteria, have been tested on small instances and have not been validated with an exhaustive search. Hamacher and Ruhe [21] are among the pioneers to propose an approximate algorithm for the bi-criterion MST that consists of two phases. Although their algorithm has been criticized as inefficient and incapable of producing the true PF [42], their 2-phase methodology has been adopted successfully by many researchers not only for the MCMST problem, but also in other problems such as the *Traveling Salesman Problem* [39], the *Quadratic Assignment Problem* [23] and the *max-ordering problem* [15]. Other 2-phase heuristics have been proposed for the MCMST problem by Anderson et al. [1], Ramos et al. [42], Steiner and Radzik [47].

In order to measure the efficacy of their evolutionary algorithm, some authors have proposed simple deterministic heuristics, based on Kruskals algorithm [31] or on Prims algorithm [40]. These algorithms, applicable to two criteria, include the *Enumeration method* of Zhou and Gen [50], which was shown to be incorrect [25], the *mc-Prim* proposed by Knowles [25], and the *mc-Kruskal* by Arroyo et al. [2]. In both mc-Prim and mc-Kruskal a parameter controls the number of solutions that will be calculated; thus they will not produce all the supported Pareto optimal solutions if their number exceeds this parameter.

Recently the *Extreme Point Deterministic Algorithm (EPDA)* has been proposed [10] that improves upon previous algorithms as it finds both supported and non-supported efficient solutions for more than two criteria. EPDA is validated against the exhaustive search algorithm *EXS*, based on the method proposed by Christofides [8], using benchmark instances generated by algorithms suggested by Knowles [25] which consist of complete graphs having three different cost functions.

When the number of nodes is large, and deterministic algorithms are slow to converge and become impractical; probabilistic algorithms can be used to find a near-optimal solution in less time. The few evolutionary algorithms proposed in literature have been tested on small instances on two criteria and the majority find only the supported efficient solutions.

Zhou and Gen [50] appear to have been the first to suggest applying, the first version of *Non-dominated Sorting Genetic Algorithm (NSGA)* [46], to the bi-criterion MST problem. This algorithm was criticized [25] for failing to calculate non-dominated solutions.

The first Evolution Strategy (ES) for the bi-criterion MST is proposed by Knowles and Corne [28] and Knowles [25] called *Archived Elitist Steady-State EA (AESSEA)*, which is a $(\mu + 1)$-ES and a population-based variant of *Pareto-Archived ES (PAES)* [27].

Bosman and Tierens [5] have proposed and tested the *Multi-objective Mixture-based Iterated Density Estimation Evolutionary Algorithm (MIDEA)* in two different multi-objective problem domains, real-valued continuous and binary combinatorial problems. The second domain includes the MCMST problem. Experimental

results, presented in [5] and [6], comparing MIDEA with *Non-dominated Sorting Genetic Algorithm (NSGA-II)* [13] and *Strength Pareto Evolutionary Algorithm (SPEA)* [48] show that MIDEA is at least comparable with the other two. NSGA-II is shown to be the most competitive in terms of *front occupation* and the *Average Front Distance*, whereas SPEA has a better *spread* than NSGA-II.

An algorithm applying the *Greedy Randomized Adaptive Search Procedure (mcGRASP)* has been suggested by Arroyo et al. [2], who argue that the randomization helps to obtain non-supported efficient solutions as well as supported ones.

More recently, other genetic algorithms have been proposed by Han and Wang [22] and by Chen et al. [7], which have been tested on small graphs in two criteria.

Gue et al. [20] have presented a particle swarm algorithm that is expected to work for more than two criteria, but no results have been reported for the multi-criterion case.

Evolutionary algorithms [29], [30] and particle swarm algorithms [19] have been suggested for the bi-objective degree-constrained MCMST problem. Moreover, special types of bi-objective MST problems have been addressed [33], [36], where a constraint in a single objective MST has been transformed into a second objective.

In order to overcome the limitations of previous algorithms the novel, fast and scalable *Knowledge-based Evolutionary Algorithm (KEA*) is proposed. KEA is designed to achieve all of the following unlike its predecessors.

- To be applicable to more than two criteria.
- To calculate both the supported and the non-supported Pareto optimal solutions.
- To be fast so that it can be used for large graphs with more than 500 nodes.

The main features of KEA include:

- The application of deterministic approaches to calculate the extreme points of the Pareto front. These are used to produce the initial population comprising of an elite set of parents.
- An elitist evolutionary search attempts to find the remaining Pareto optimal points by applying a knowledge-based mutation operator. The domain knowledge is based on the k-best approaches in deterministic methods.
- Marking schemes that reduce the re-evaluation of solutions; and cut-off points that eliminate the dominated regions of the search space are applied.

Experimental results are obtained from hard benchmark instances of the problem that are generated using the algorithms proposed by Knowles [25] for complete graphs. KEA is verified and validated against the exhaustive search algorithm *EXS*, based on the method suggested by Christofides [8]. Comparative results with an adapted version of *Non-dominated Sorting Genetic Algorithm (NSGA-II)* [13] and with the *Extreme Point Deterministic Algorithm (EPDA)* [10] are reported. It is shown that the strength and superiority of KEA is due to the domain knowledge that is autonomously incorporated, making it efficient, fast and scalable.

Because of its speed and efficiency, KEA has much potential for rendering the MCMST model applicable to real world problems arising in diverse systems:

1. Analysis and design of various networks [3], for example, large scale telecommunication networks [16], distributed computing networks [38], spatial networks for commodity distribution (gas pipelines or train tracks) [18] and processor networks [32].
2. Hardware design [17], for example, connectionist architectures for analog and VLSI circuits [37].
3. Data collection and information retrieval, for example, file mirroring/transfer, bit-compression for information retrieval and minimizing message passing [4].

17.2 Knowledge-Based Evolutionary Algorithm (KEA)

To represent the spanning trees (STs), the edge-set encoding suggested by Raidl and Julstrom [41] is used with the difference that each edge (*gene*) is represented by only one integer, which is the index of the corresponding Edge List. As a result each edge-set (*Genotype*) requires $n-1$ integers, i.e. half the space used in [41], where each edge was represented by two integers: the end nodes. Recently an efficient representation has been proposed by Soak et al. [45]; however, when used in conjunction with Kruskals algorithm 'it does not guarantee to obtain a valid tree' [45] and supporting heuristics have to be applied to ensure validity. The edge-set representation ensures that each ST is represented by a unique encoding as discussed and demonstrated by Raidl and Julstrom [41].

The *Fitness function* is a fitness vector defined to be the vector of *Total Costs*, calculated by summing the costs of the edges forming the ST with respect to each criterion.

Each solution on the *approximate Pareto set*(*APS*) is comprised of the *Genotype*, the *Total Costs*, and the *Origin* of the ST. Origin is an integer equal to i for $i = 1,\cdots,p$ and represents the *Edge List* from which the ST was calculated. Each edge has a flag used in phases 2 and 3 of KEA. The size of the APS is flexible.

The *Population size* is not fixed, because it is not possible to calculate a priori how many STs will be created. Although KEA is similar to a $(\mu+\lambda)$-ES with $\rho = 1$ (cloning), neither μ nor λ is fixed a priori, where μ is the number of parents, λ the number of children and ρ is the number of parents involved in the reproduction process. Michalewicz [35] discusses GAs with varying population size some of which introduce the concept of age. In KEA, an individual does not age and is eligible to be selected provided it has not been selected previously.

At each iteration, *Mutation* is the only genetic operator applied. Preliminary tests showed that the crossover operator increased CPU time without increasing performance. Given the inheritance of the characteristic genes (defined in section B below), an ordinary crossover operator would be impractical. It is possible to use a sophisticated crossover operator; however, it was found that that the resulting offspring were usually dominated even if the parents were non-dominated.

The *Parameters*: Two parameters that depend on the size of the graph are: n = number of the nodes and m = number of the edges in the graph being considered. The latter is derived from n by the following formula which applies only to complete

graphs: $m = (n \times (n-1))/2$ and p is the number of criteria. *Eps* is the domain dependent precision parameter. Two cost values are considered equal if their absolute difference is less than *Eps*. The size of the PF is controlled by varying the value of *Eps*. The crossover and mutation probabilities are equal to zero and one respectively. Finally, g is the number of generations.

17.2.1 Phases of KEA

KEA consists of three phases as summarized in Figure 17.3.

In **phase 1**, p lists of edges are created, where $EdgeList[i]$ is ordered with respect to the $i^{th.}$ criterion. Then the extreme minimum spanning trees $MSTs$ are found, each of which minimizes one of the criteria, applying Kruskals algorithm [31] to the corresponding $EdgeList$. Kruskals algorithm was selected because its ordered edge list based approach was suited to the data structures used in KEA.

To each edge a Boolean flag is assigned called *InTree*, which is initialized to zero. Once an MST is calculated, the $InTree$ flags of its edges are set to one in the corresponding $EdgeList$.

If the p extreme $MSTs$ are identical, then the APS is reduced to a single solution and the algorithm stops.

Phase 2 corresponds to the creation of the initial population of spanning trees STs by elementary tree transformations [8]. This consists of mutating only one edge of the extreme $MSTs$ at a time for each ST. These STs are the genotypic neighbours of the $MSTs$, because they differ in only one edge, called the *Characteristic* edge. The use of two flags related to the *Characteristic* and *InTree* edges is similar to methods used by Gabow [17] and Katoh et al. [24] in order to avoid re-evaluation of solutions that have already been discovered. The *Characteristic* flags are initialized to -1.

All the non-dominated genotypic neighbours of MST_i, for $i = 1, \cdots, p$, are calculated as follows. For each edge (u,v) in MST_i that is removed, the algorithm scans $EdgeList[i]$ for as many as possible feasible edges (r,s) to replace (u,v). Each edge that satisfies these three conditions is used as a replacement for edge (u,v):

a) $(r,s) \notin InTree$,
b) The inclusion of edge (r,s) will not create a cycle,
c) $C_j(r,s) < C_j(u,v)$ for at least one j, where $j = 1, \cdots, p, j \neq i$.

Condition (c) constitutes further domain knowledge that is incorporated in KEA, which improves performance as subsequently it does not evaluate a large number of STs that will be dominated. This condition is similar to that used in the k-best approaches. A similar method has been applied by Raidle and Julstrom [41] so that the offspring is at least as good as its parent.

If the edge (r,s) satisfies the above conditions, it is marked as *Characteristic* by setting its characteristic flag equal to the index of the edge (u,v). Next the *Total Costs (TC)* of the new ST (child) are calculated from the following relation: $\forall i, TC_i(Child) = TC_i(Parent) - C_i(u,v) + C_i(r,s)$.

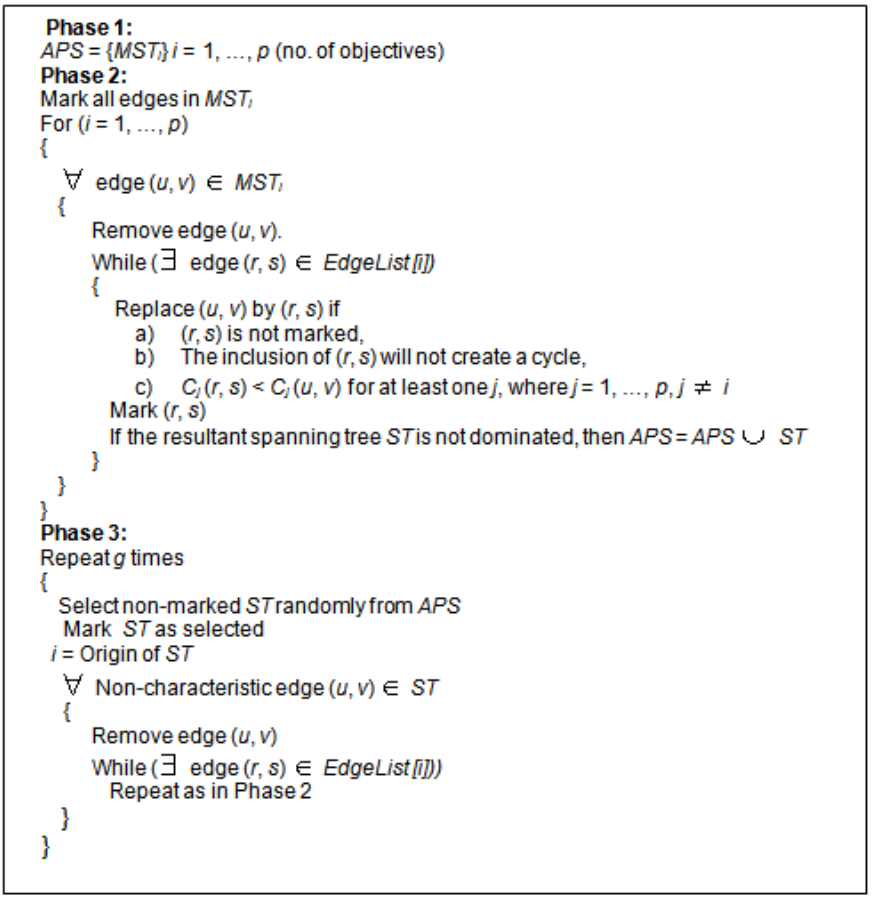


Fig. 17.3 The code of KEA

The *Total Costs* of the new *ST* are compared with the *Total Costs* of non-dominated *STs* on the *APS*. If the new *ST* is not dominated it is added to the *APS*. If the new *ST* dominates one or more *STs* on the *APS*, then they are discarded from it.

Once all the possible edges (r,s) have been found to replace the edge (u,v) and the corresponding *STs* are created, MST_i is restored by inserting the edge (u,v). Then another edge of MST_i is considered to be replaced. The procedure is repeated until all edges of MST_i are mutated one by one.

The advantage of KEA constructing the *APS* from all the extreme points is three-fold:

a) Genotypic neighbours of the extreme MSTs are more likely to be non-dominated than other solutions. Therefore, they form an elite initial population superior to a randomly generated population.

b) The algorithm converges more rapidly.
c) It guarantees diversification.

The inconvenience of this strategy is that despite the marking scheme some re-evaluation of solutions is unavoidable as a few of the STs created from one end may also be created from another end. Therefore, if the fitness function of a new *ST* is equal to that of an *ST* in the *APS*, the new *ST* is considered a duplicate and discarded without comparing the edges to determine whether they are the same. The policy of having one representative genotype for all those that map to the same fitness function has been adopted by previous authors [47], [34].

Phase 3 corresponds to the actual evolutionary algorithm in which the first generation *STs*, calculated in phase 2, are treated as parents. This phase, which is recursive and repeated g times, is comprised of selection and reproduction. The *APS* serves as the pool of potential parents. One individual from the *APS* is selected randomly in each generation and used to create as many *STs* as possible. Only the non-characteristic edges can be mutated. Thus the selected parent will have none, one or several children, all in the course of one generation. The characteristic edges of a parent are passed down to its offspring. Each selected parent, if non-dominated throughout the algorithm, will remain on the *APS*, but will now be excluded from the pool of potential parents. Non-dominated *STs* will be added to the *APS* to become parents if selected. The same procedure, as in Phase 2, is used for reproduction except that in condition (c) above, the clause $j \neq i$ is removed. The final stopping criterion is the user specified number of generations.

17.3 Experimental Results

17.3.1 Benchmark Data Sets

Three types of hard benchmark instances are generated using algorithms suggested by Knowles [25], which consist of real-valued costs, imposing the additional difficulty of comparing real numbers.

Random Costs when used in the bi-objective case are uniformly distributed on the closed interval $[10.0, 100.0]$, and $[10.0, 50.0]$. In order to test scalability of KEA, this benchmark also considers the tri-criterion case where random costs are uniformly distributed on closed intervals $[10.0, 50.0 \times p]$, for $p = 1$ to 3. *Correlated* and *Anti-Correlated Costs* set the coefficient of correlation equal to 0.7 and -0.7, respectively, for all the instances.

17.3.2 Benchmark Algorithms

In order to compare KEA with existing evolutionary algorithms, *Non-dominated Sorting Genetic Algorithm, NSGA-II* was investigated as it is commonly applied to multi-criterion optimization problems. NSGA-II is a fast elitist multi-objective genetic algorithm, proposed by Deb et al. [13], [12] based on the first version proposed by Srinivas and Deb [46], which was applied to the MCMST problem [50] and to

related problems [16]. As highlighted earlier, comparative experimental results on the MCMST problem presented by Bosman and Thierens [5], [6], have shown NSGA-II to be the most competitive in terms of front occupation and the average front distance, whereas SPEA [48] has a better spread than NSGA-II. Thus, the general-purpose NSGA-II was selected for comparison with KEA and was specifically tailored to the MCMST problem; hence renaming it *NSGA2*.

The adapted version, NSGA2 is faithful to the original NSGA-II . The initial parent population consists of all the n star trees that contain all the edges (genes) and for this reason is superior to a random initial population. If the size of the initial parent population is such that $n < P$ (where P is the set population size), then $2P - n$ children are created by the usual procedures of selection, crossover and mutation, in order to complete the initial combined parent and child population (the combined population contains $2P$ individuals).

The NSGA-II schemes for reproduction [13], [12] required tuning for the MCMST problem. In addition to the Total Costs of an ST, the Rank and the Crowding Distance are also used as dominance criteria as in NSGA-II.

Population size (P), number of APSs (F) and number of generations (g) are the user defined parameters for NSGA2. Preliminary trials were used to select the NSGA2 parameters including that there should be twice as many fronts as population size in case each individual in a combined parent and child population is placed in a distinct front. The crossover probability was thus set equal to 0.9 [13], whilst the mutation probability was calculated as $1.0/((n \times 1.0) - 1.0)$ [12].

The exhaustive search algorithm described by Christofides [8] was adapted to the MCMST problem and termed EXS to be used for validation and verification tests for instances with at most ten nodes. For larger graphs the Pareto fronts discovered by the Extreme Point Deterministic Algorithm (EPDA) [10] are used as reference.

17.3.3 Performance Indicators

An important issue in multi-criterion optimization is the comparison of algorithms in a quantitative manner. According to Schwefel [44], 'Numerical tests employ various model objective functions. The results are evaluated from two points of view':

- Efficiency or speed of approach to the objective,
- Effectivity or reliability under varying conditions.

To compare the efficiency of the algorithms, the three classical performance indicators, which are recommended in literature [48], [26] will be used, namely the hyper-volume (the S-measure), the additive binary epsilon (the I-indicator) and the attainment surface plots. Four further indicators will be measured in order to refine the comparison, defined as follows:

CPU Duration of program execution in seconds.
TTC Total number of trees created.
FO Front occupation, the number of non-dominated solutions an algorithm places on its APS.

$\%FO = (FO/TTC) \times 100$, shows the efficiency of an algorithm. An efficient algorithm will derive its APS with few calculations, whilst avoiding the infeasible region, resulting in a large $\%FO$.

Effectivity is demonstrated by testing with the three cost functions and increasing problem sizes.

17.3.4 Numerical Results

All the programs were coded in C and executed on Microsoft Window XP Professional with X86-based PC system and Intel 2.8 GHz processor. Two series of results are reported in this section.

The first series of experiments correspond to verification tests conducted against EXS for graphs having four to ten nodes with $Eps = 0.0001$. In all these validation tests KEA and NSGA2 performed very well. NSGA2, although slower, determined all the points on the PF. KEA was able to find the true PF in all the 28 different experiments except that, in one case it found all but one of the optimal solutions.

Table 17.1 shows the results for the anti-correlated cost instance with 10 nodes. KEA has the largest $\%FO$ showing its efficiency. It has succeeded in finding the entire PF with only 0.464 K calculations as opposed to 42.6 K calculations by NSGA2 and 100 000 K evaluations by EXS. NSGA2 found the true PF in 50 generations with a population size of 380, as opposed to 40 generations for KEA. The speed of KEA was also confirmed by its CPU time of 6.6 seconds as opposed to 13.8 seconds for NSGA2. Since the results for the other cost types are similar [11], they are not shown here.

Table 17.1 Results for a graph with 10 nodes, 45 edges, $10^8 (= 100,000,000)$ STs and anti-correlated costs with $coef.of cor. = -0.7$, $Eps = 0.0001$, (NSGA2: crossover-prob. = 0.9, mutation-prob. = 0.111), g = number of generations, P = population size for NSGA2 and F is the number of fronts. TTC measures total number of STs created of which a number equal to front occupation , FO, are placed in the APS. The values of $\%FO$ show the non-dominated portion of the TTC. CPU is measured in seconds

	EXS	KEA	NSGA2
g		40	50
P			380
F	1	1	760
TTC	100,000,000	**464**	42,575
FO	38	38	38
$\%FO$	$38^{-6}\%$	**8%**	0.09%
CPU Sec.	467.5	**6.6**	13.8

Parameter tuning tests were performed with benchmark complete graphs having 20, 30, 50 and 100 nodes in order to determine the best values for decision parameters. These tests were performed with two different values for the precision

parameter, $Eps = 0.001$ and $Eps = 0.0001$. It was found that when $Eps = 0.0001$, the corresponding APS for the same graph contained more solutions and was more refined. All solutions with equal Total Costs are mapped into a single point on the APS. Since two real cost values are considered equal if their difference is less than Eps, variations in Eps result in variations in the number of points on the APS. As noted by Kumar et al. [33], this is 'analogous in some way to a sort of sharing/niching mechanism (in objective space) which effectively controls the selection pressure and thus partly contributes to diversity'. Thus Eps may be used as a control parameter to vary the size of the niches and hence the number of reported points on the APS.

In the second series of experiments, the best parameter values that were determined in the preliminary tests were used to obtain statistical results. Since KEA and NSGA2 obtained similar fronts for graphs with $n \leq 50$, in this section the statistical results averaged over 10 runs for the 100-node graphs will be presented in Table 17.2, Table 17.3 and Figures 17.4 and 17.5. More runs whould be impractical due to the CPU times of NSGA2.

Table 17.2 shows the statistical values of the S-measure (mean, μ, and standard deviation, σ), which indicates the spread of the discovered APS. The bounding point for the calculation of S values is chosen to be the anti-ideal (nadir) point of the true PF whose coordinates can be found from the extreme points of the APS calculated by EPDA or KEA. KEA has the larger values for the S-measure since it has a larger spread than NSGA2. KEA also has a higher mean and a smaller standard deviation showing a more reliable performance. The Mann-Whitney rank-sum test confirms that the differences between the means of the two distributions are significant in all the cases at 99% level of confidence.

Comparison of TTC, FO, $\%FO$ and CPU shows that NSGA2 is slower than EPDA and calculates more spanning trees, yet it finds APSs that contain fewer solutions. KEA is the fastest algorithm arriving at its final solution set in a few minutes compared with hours in some cases. In the case of anti-correlated costs , even though KEA calculates a larger number of trees than NSGA2 (3,759 K against 2,938 K), it does so in a shorter period of time (698 sec. against 9,080 sec.).

In all the instances tested, the APS found by KEA is on average three times larger than the APS calculated by NSGA2. It was also found that the FO of KEA was quite close to the FO of EPDA in all cases. The $\%FO$ confirms that KEA is capable of producing superior APSs to NSGA2 but with fewer calculations.

The median and the inter-quartile range (IQR) values of the I-indicator are shown in Table 17.3. The positive I-indicators show that KEA and NSGA2 are incomparable in anti-correlated and random instances. However, since I(KEA) values are smaller than I(NSGA2), it can be concluded that, in a weaker sense, KEA is better in more than %50 of the runs. In the case of correlated costs, the median score of KEA is negative indicating that it is better than NSGA2 in a strict sense on more than %50 of runs. Therefore, the additive epsilon indicator confirms the superiority of KEA over NSGA2 in all instances. The Mann-Whitney rank-sum tests confirm that the distributions of I values are significantly different at %99 level of confidence.

Table 17.2 (100 Nodes) Mean and standard deviation of the *S*-measure, the CPU times in seconds and the values of other parameters averaged over 10 runs of KEA and NSGA2 on Graphs with 100 Nodes and three different Cost Types. g = number of generations, P = population size for NSGA2. TTC measures total number of STs created of which a number equal to front occupation, FO, are placed in the APS. The values of $\%FO$ show the non-dominated portion of the TTC

	Correlated Costs	**Anti-Correlated Costs**	**Random Costs**
Eps	0.0001	0.0001	0.0001
g KEA	3 K	20 K	20 K
g , P NSGA2	1000 , 1.5 K	500 , 5 K	500 , 5K
S-value KEA μ , (σ)	**25.1** , **(0.1)**	**2,051** , **(2.5)**	**7,705 K** ,**(17,184)**
S-value NSGA2 μ , (σ)	17.7 , (1.1)	1,878.7 , (7.9)	7,07 K , (26,700)
Mann-Whitney z-value	-3.74	-3.74	-4.35
Significant level	$> 99\%$	$> 99\%$	$> 99\%$
CPU Sec. EPDA	78.2	2,407	2,021
CPU Sec. KEA μ , (σ)	**37.1** , **(3)**	**698.3** , **(59.8)**	**515.5** , **(51.9)**
CPU Sec. NSGA2 μ , (σ)	1,197 , (21.9)	9,079.8 , (52.4)	8,852.9 , (33.6)
TTC EPDA	412 K	5,696 K	4,643
TTC KEA μ , (σ)	**383 K** , **(31 K)**	**3,759 K** , **(350 K)**	**3,063** , **(199 K)**
TTC NSGA2 μ , (σ)	1,942 K , (55 K)	2,938 K , (44 K)	21,837 K , (19 K)
FO EPDA	**1,339**	**4,453**	**4,805**
FO KEA μ , (σ)	1,315 , (61.2)	4,255 , (124.1)	4,430 , (139.8)
FO NSGA2 μ , (σ)	501 , (74.3)	1,550 , (308.3)	1,860 , (208.2)
$\%FO$ EPDA	0.32%	0 08%	0.10%
$\%FO$ KEA μ , (σ)	**0.34%** , **(0.20)**	**0.11%** , **(0.04)**	**0.14%** , **(0.07)**
$\%FO$ NSGA2 μ , (σ)	0.03% , (0.13)	0.05% , (0.69)	0.01% , (1.1)

Table 17.3 Median and Inter-quartile Range (IQR) values of the Additive Epsilon Indicator Obtained after 10 evaluations of KEA and NSGA2 on Graphs with 100 Nodes and different Cost Types. Lower values indicate better performance. $Eps = 0.0001$

	I**(KEA)**	I**(KEA)**	I**(NSGA2)**	I**(NSGA2)**	**z-value**
Cost Type	*Median*	*IQR*	*Median*	*IQR*	$> 99\%$
Anti-Correlated	**2.19**	0.17	3.40	0.18	-12.22
Correlated	**-0.36**	0.17	1.18	0.35	-12.22
Random	**154.40**	43.24	301.23	12.20	-12.22

Figure 17.4 compares the best attainment surface plots obtained by the three algorithms in the case of random costs. Since the median and the worse attainment surfaces are close to the best, they are not plotted for clarity. Figure 17.4 shows that the middle section of the surface plot of KEA is dominated by EPDA; and that NSGA2 mostly dominates the middle portion of KEA and is almost superimposed on the middle section of EPDA. Nonetheless, NSGA2 and KEA are incomparable

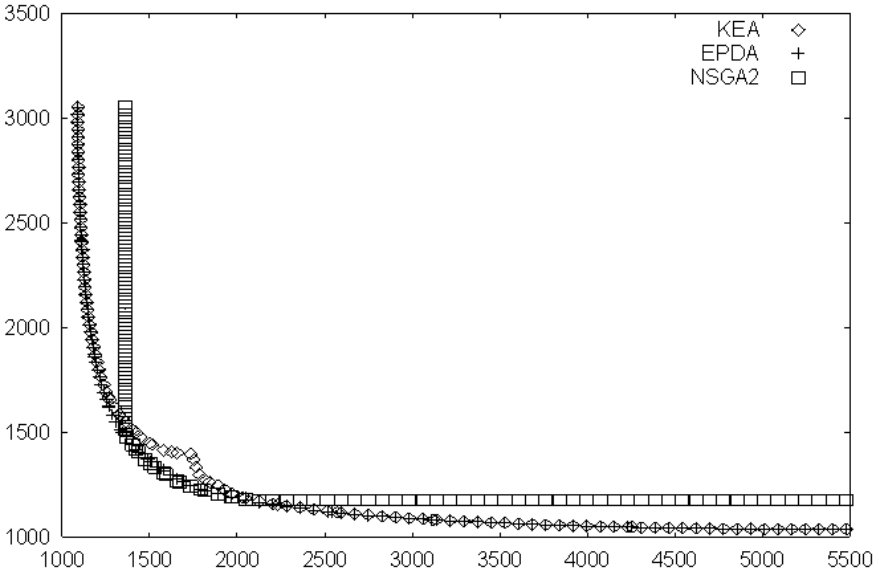


Fig. 17.4 Best attainment surface plots by KEA and NSGA2 compared with the attainment surface of EPDA for a graph with 100 nodes and random costs, (NSGA2: $P = 5,000$, $g = 500$, crossover-prob. = 0.9, mutation-prob. = 0.01) (KEA: $g = 20,000$)EPDA $FO = 4,805$, KEA $FO = 4,317$, NSGA2 $FO = 2,030$, $Eps = 0.0001$

in terms of the I-indicator. The attainment surface plots showed that in all instances NSGA2 converges to a small area in the middle of the objective space, due to the way NSGA2 explores the decision space (similar to Knowles [26]).

In practical situations with a short time limit, KEA dominates both EPDA and NSGA2. To illustrate this point, the attainment surface plots for the random cost instance are shown in Figure 17.5, where the running times of EPDA and NSGA2 have been limited to that of KEA. In this figure KEA dominates both algorithms in the middle section of the APS, which is not the case for an unlimited execution time. Where time is not a critical issue and the middle sections are important, neither KEA nor NSGA2 is recommended as EPDA dominates both.

17.3.5 Experimental Complexity of KEA and NSGA2

Since the true PF is not known and since the task of creating STs is probabilistic, only a probabilistic theoretical complexity for both KEA and NSGA2 can be estimated, which is beyond the scope of this chapter. The experimental complexities of the two algorithms, compared in this section, depend on the implementation and data structures used. Most of the functions and procedures that perform time consuming operations (to check for non-dominance and to check for the creation of cycles)

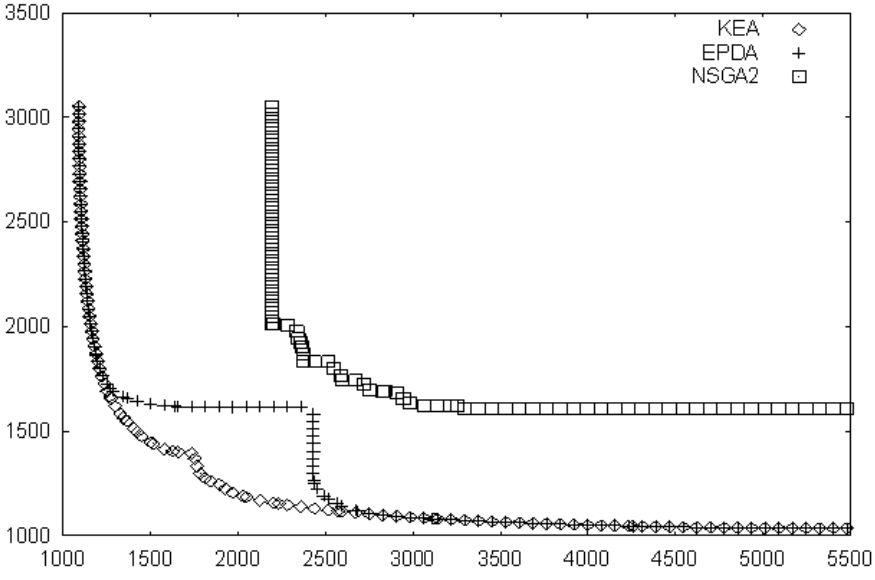


Fig. 17.5 Attainment surface plots for a graph with 100 nodes and random costs, where the execution time has been limited to that of KEA. Approximate $CPU = 515.5$ sec. (KEA: $g = 20,000$), resolution = 60, total no. of test points = 120 (NSGA2: $P = 5,000$, crossover-prob. = 0.9, mutation-prob. = 0.01) EPDA $FO = 2,782$, Approximate KEA $FO = 4,317$, NSGA2 $FO = 21$, $Eps = 0.0001$

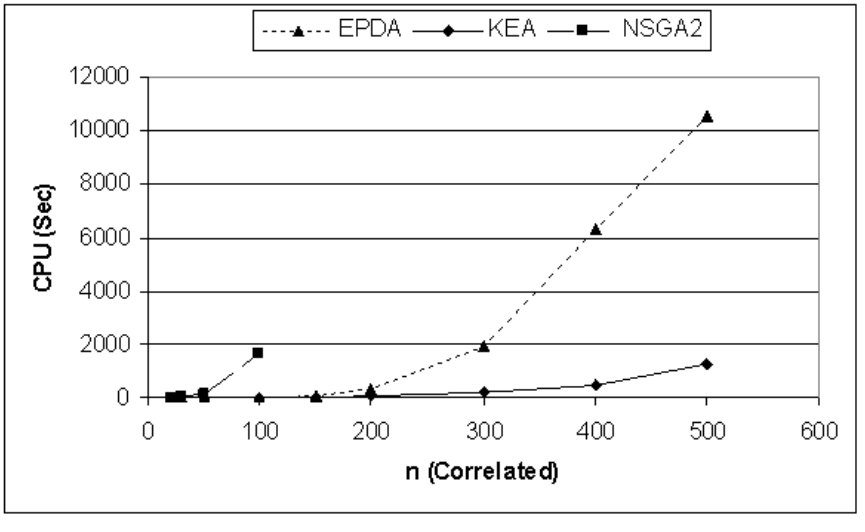


Fig. 17.6 Plots showing *CPU* times of EPDA, KEA and NSGA2 vs. no. of nodes of graphs with correlated costs, $Eps = 0.01$

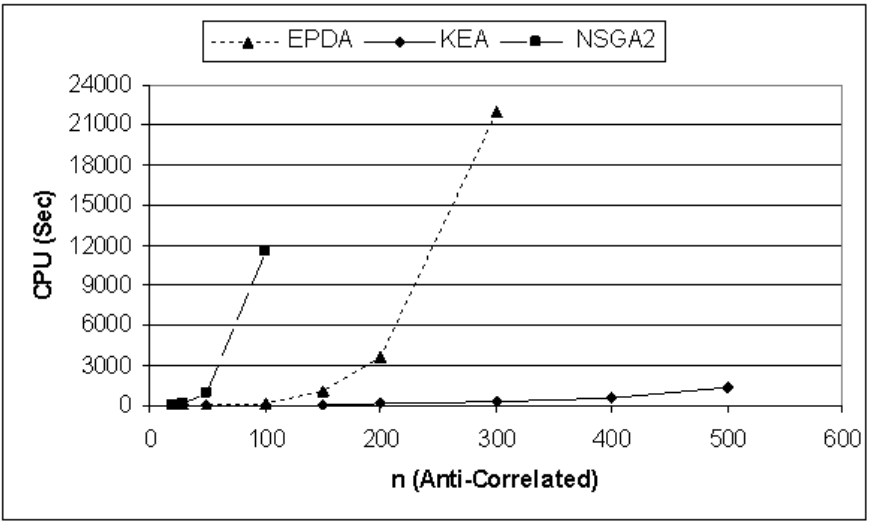


Fig. 17.7 Plots showing *CPU* times of EPDA, KEA and NSGA2 vs. no. of nodes of graphs with anti-correlated costs, $Eps = 0.01$

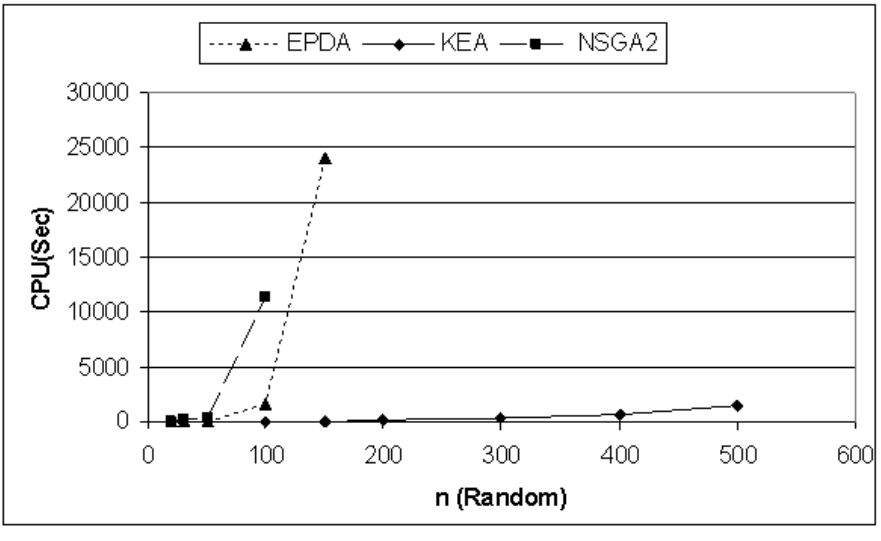


Fig. 17.8 Plots showing *CPU* times of EPDA, KEA and NSGA2 vs. no. of nodes of graphs with random costs, $Eps = 0.01$

are common to all the algorithms (EXS, EPDA, KEA and NSGA2). Therefore their experimental complexities can be compared on the basis of the CPU times.

The superiority of KEA over NSGA2 in terms of *CPU* time is demonstrated in Figures 17.6 to 17.8 by plots showing *CPU* times of both algorithms versus the number of nodes. *CPU* times of EPDA are also shown to demonstrate that NSGA2 does not scale well. In order to compare the speed of KEA with NSGA2 more effectively, both algorithms have been executed for 1,000 generations. The population sizes of NSGA2 have been set approximately equal to the Front Occupation of EPDA for each instance.

The plots show that although NSGA2 is relatively fast for graphs with ≤ 50 nodes, its time complexity becomes impractical for graphs with 100 or more nodes. For example, Figure 17.7 demonstrates that NSGA2 can take nearly five times longer than EPDA for a graph with twice the number of nodes. KEA is very fast compared with both EPDA and NSGA2, e.g. it uses only 1362 seconds on the anti-correlated instance with 500 nodes.

17.3.6 Scalability of KEA

Generally, MCMST algorithms are tested on two criteria and $n < 100$ problems. A proposed benefit of KEA is scalability both in the number of nodes and the number of criteria. To test scalability in number of nodes, KEA was compared with EPDA for instances with 150 and 200 nodes on the three benchmark cost types. Because NSGA2 became impractically slow with these problems (see Figures 17.6 to 17.8),

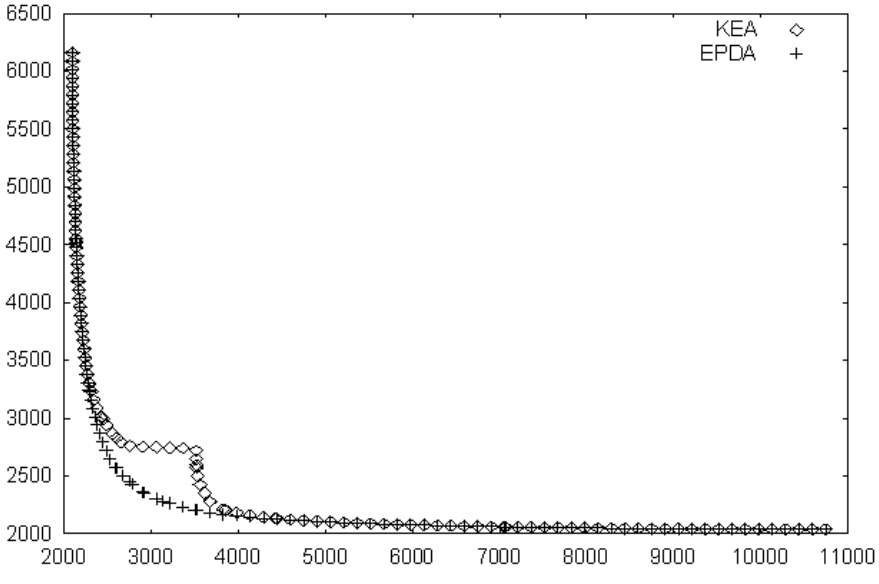


Fig. 17.9 Attainment surface plots for a Graph with 200 nodes and random costs, (KEA: $g = 60,000$), resolution = 60. The middle of the attainment surface of KEA is dominated by that of EPDA while the tails are almost identical, $Eps = 0.01$

it was not tested on these graphs. In the case of graphs with 150 nodes, the *S*-measures of KEA were very close to EPDAs. This suggests that the APSs of the two algorithms were very close, although the positive *I*-indicators mean that they were incomparable. However, since *I*(EPDA) indicators were smaller, it means that in a weak sense EPDA dominated KEA. The weak dominance of EPDA over KEA is due to the dominated middle part of KEAs front. The Mann-Whitney rank sum test indicated that the *I* values are significantly different in all instances at %99 level of confidence.

In some 200-node instances, KEA obtained better values for the S-measure than EPDA. Moreover, although both *I*-indicators were positive, KEA was better in a weak sense as its *I*-indicator was smaller. In the case of Random costs with 200 nodes, the *S*-measure of KEA was close to EPDAs, suggesting that the APS discovered by KEA was relatively close to the APS of EPDA. The strength of KEA lies in that it was able to find this APS in a third of the time of EPDA.

The best attainment surface for KEA is compared with EPDA in Figure 17.9, which confirms that the APSs calculated by the two algorithms are very close except for the middle section.

17.3.7 Tri-Criterion Instances

To test the scalability in the number of criterion, KEA was executed on benchmark complete graphs for the tri-criterion case with random costs. Results for 10, 50 and

Table 17.4 (Three Criteria) Values of the *S* and the *I* indicators and the *CPU* times in seconds averaged over 10 runs of KEA compared with EPDA on graphs with 10, 50 and 100 nodes and random costs in three criteria

	10 Nodes	**50 Nodes**	**100 Nodes**
Eps	0.0001	9	9
g KEA	1 K	10 K	30 K
S-value EPDA	9 M	**3.296 M**	22,575 M
S-value KEA μ , (σ)	**9.5 M** , **(198 K)**	2,988 M , (135 M)	**24,088 M** , **(1,185 M)**
I (EPDA, KEA) *Median* , (*IQR*)	**0** , **(0)**	**69.57** , **(26.13)**	170.75 , (68.81)
I (KEA, EPDA) *Median* , (*IQR*)	2.66 , (5.96)	82.03 , (28.81)	**139.67** , **(36.85)**
Mann-Whitney *z*-value	-3.36	-1.85	-3.74
Significant level	$>$ 99%	95%	$>$ 99%
CPU Sec. EPDA	2.9	933	86K
CPU Sec. KEA μ , (σ)	**0.65** , **(0.08)**	**887.2** , **(51.79)**	**23 K** , **(1,797.1)**
TTC EPDA	25 K	14,807 K	342,811 K
TTC KEA μ , (σ)	**23 K** , **(885.8)**	**13,466 K** , **(531 K)**	**205,800 K** , **(8,657 K)**
FO EPDA	**628**	**1,743**	**4,265**
FO KEA μ , (σ)	624.5 , (3.2)	1,700.1 , (55.6)	3,400.9 , (113.9)
%FO EPDA	2.54%	0.01%	0.001%
%FO KEA μ , (σ)	**2.73%** , **(0.36%)**	0.01% , (0.01)	**0.002%** , **(0.001%)**

100 nodes are compared with those obtained by EPDA in Table 17.4. It can be seen that KEA is much faster than EPDA, in particular for $n = 100$. The $\%FO$ shows that the computational effort of the two algorithms is almost identical. In the tri-criterion case, there is no clear recommendation for the more suitable algorithm as the I-indicator is positive in all cases. The S-measure suggests that KEA is better for the 10 and 100 node instances.

The attainment surface obtained by EPDA and the best attainment surface obtained by KEA for the graph with 100 nodes are shown in Figure 17.10. To aid visualization, the resolution is reduced so that only 84 points are shown on the attainment surfaces. Similar to the bi-criterion case, the APS determined by KEA and EPDA are close, but EPDA partly dominates KEA.

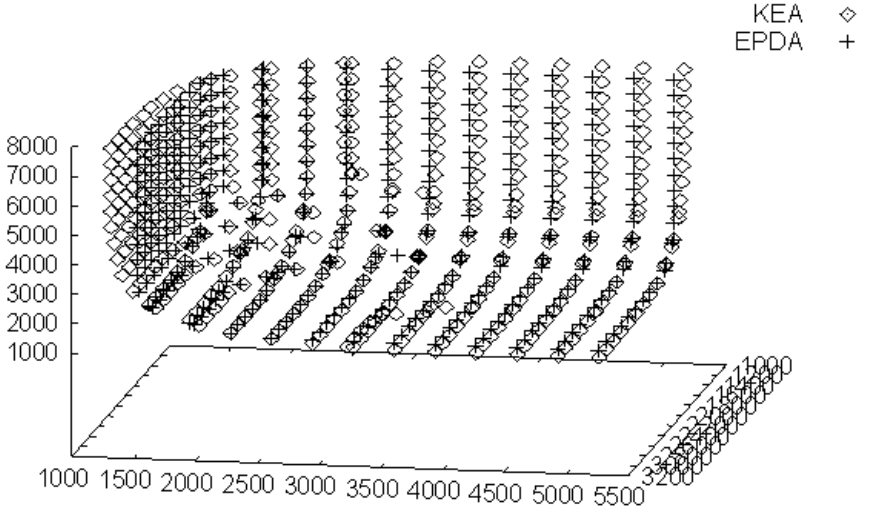


Fig. 17.10 Attainment surface plot obtained by EPDA and the best attainment surface plot obtained by KEA for a graph with 100 nodes, random costs and three criteria, (KEA: $g = 30000$), resolution 14 (only 84 points are projected for clarity), $Eps = 9$

17.4 Alternative Algorithms Based on KEA

The only deficiency of KEA is that the middle section of the APS it calculates for large graphs is dominated. As shown in figures 17.3, and 17.9, while the tails of the attainment surface of KEA are almost identical to those of EPDA, the middle is dominated. In order to overcome this deficiency, a number of modifications have been tested [10].

In KEA-M, a version of KEA, the middle point of the APS and all its first neighbours are calculated after the evolutionary Phase 3. These new solutions are then placed on the existing APS and used to eliminate the dominated points. However,

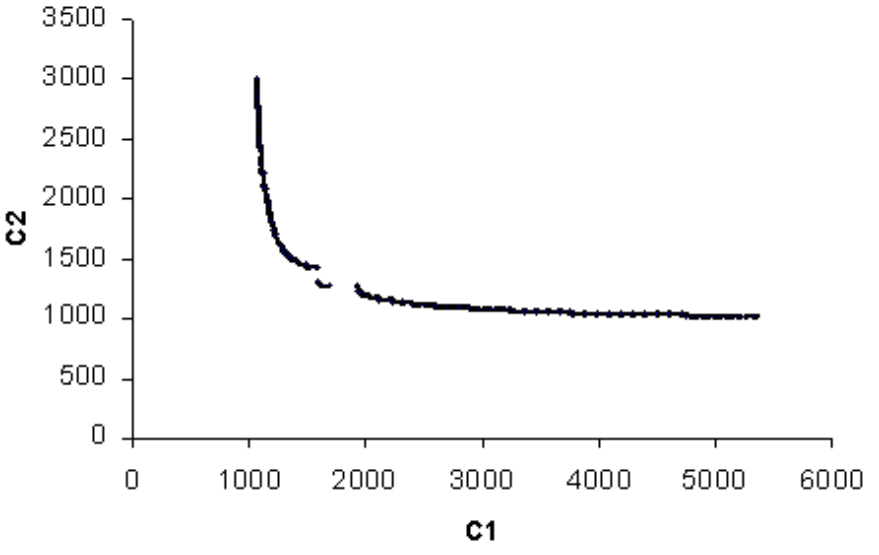


Fig. 17.11 The approximate Pareto set obtained by KEA-M after 20000 generations for a graph with random costs and 100 nodes which shows two gaps and still dominated solutions, $Eps = 0.0001$

as seen in Figure 17.11, these solutions are unable to eliminate all the dominated solutions and leave gaps in the final APS.

In KEA-G all the supported solutions are found and placed on the APS along with the extreme points in the first phase; and then the other phases are performed as in KEA. However, this algorithm does not perform well in verification tests as it is unable to calculate the complete APS. The reason for this failure is that the supported solutions dominate and consequently eliminate many of the intermediate solutions whose non-dominated offspring are never produced.

Finally, KEA-W [11] is proposed that includes the search for non-dominated solutions from the middle point of the APS as well as the extreme points. The inclusion of the middle point, found by the geometric method of Ramos et al. [42], ensures that the middle section of the APS is not dominated. Verification tests against EXS, demonstrated that KEA-W is capable of finding all the optimal solutions in graphs with four to ten nodes in all the instances tested. In particular KEA-W found the solution that was missed out by both EPDA and KEA in the case of the graph with random costs and ten nodes.

Further experimental results were carried out with graphs having 20, 30, 50 and 100 nodes and three different cost types, comparing KEA-W with EPDA, KEA and NSGA2. Table 17.5 shows the statistical results for KEA-W averaged over ten runs compared with those obtained by KEA, NSGA2 and EPDA. KEA-W has a larger S-value in all instances compared with KEA and NSGA2, whilst the S-values obtained

Table 17.5 (100 Nodes, bi-criterion) Mean and standard deviation of the *S*-measure and other parameters and the *C*PU times in seconds averaged over 10 runs of KEA-W compared to those obtained by KEA, EPDA and NSGA2 on graphs with 100 nodes and three different cost types

	Correlated Costs	**Anti-Correlated Costs**	**Random Costs**
Eps	0.0001	0.0001	0.0001
g KEA-W	3 K	20 K	20 K
g KEA	3 K	20 K	20 K
g , *P* NSGA2	1000 , 1.5 K	500 , 5K	500 , 5 K
S-value EPDA	25.3	**2073.5**	**7,810**
S-value KEA-W μ , (σ)	25.3 , (0.004)	**2,073.6** , **(0.16)**	7,807 K , (1,172)
S-value KEA μ , (σ)	25.3 , (0.1)	2,063.6 , (6.7)	7,727 K , (30,977)
S-value NSGA μ , (σ)	17.7 , (1.1)	1,878.7 , (7.9)	1,077 K , (26,700)
CPU Sec. EPDA	78.2	2,407	2,021
CPU Sec. KEA-W μ , (σ)	874 , (209)	27 K , (8 K)	11 K , (2 K)
CPU Sec. KEA μ , (σ)	**37.1** , **(3)**	**698.3** , **(59.8)**	**515.5** , **(51.9)**
CPU Sec. NSGA2 μ , (σ)	1,197 , (21.9)	9,079.8 , (52.4)	8,852.9 , (33.6)
TTC EPDA	412 K	5,696 K	4,643 K
TTC KEA-W μ , (σ)	56 M , (5 M)	235 M , (25 M)	294 M , (22 M)
TTC KEA μ , (σ)	**383 K** , **(31 K)**	**3,759 K** , **(350 K)**	**3,063 K** , **(199 K)**
TTC NSGA2 μ , (σ)	1,942 K , (55 K)	2,938 K , (44 K)	2,837 K , (19 K)
FO EPDA	1,339	4,453	4,805
FO KEA-W μ , (σ)	**1,457** , **(40)**	**5,236** , **(86)**	**5,824** , **(207)**
FO KEA μ , (σ)	1,315 , (61.2)	4,255 , (124.1)	4,430 , (139.8)
FO NSGA2 μ , (σ)	501 , (74.3)	1,550 , (308.3)	1,860 , (208.2)
%FO EPDA	0,32%	0.08%	0.10%
%FO KEA-W μ , (σ)	0.003% , (0.001)	0.002% , (0.0003)	0.002% , (0.0001)
%FO KEA μ , (σ)	**0.34%** , **(0.20)**	**0.11%** , **(0.04)**	**0.14%** , **(0.07)**
%FO NSGA2 μ , (σ)	0.03% , (0.13)	0.05% , (0.69)	0.01% , (1.1)

by EPDA and KEA-W are very close. The inclusion of the mid-point of the PF has increased the accuracy of KEA-W compared with that of KEA as well as its front occupation. However, this precision has been achieved at the cost of large *CPU* time, which is due to the large total number of STs calculated (*TTC*). The presence of an additional search point results in the unnecessary re-evaluation of many solutions.

The median and the inter-quartile range (*IQR*) values of the *I*-indicator are shown in Table 17.6, where lower values indicate better performance. The positive *I*-indicators show that KEA-W and KEA are incomparable in all instances. However, since *I*(KEA-W, KEA) values are smaller than *I*(KEA, KEA-W), it can be concluded that, in a weaker sense, KEA-W is better in more than 50% of the runs. The same conclusion can be made when comparing KEA-W with EPDA. In the case of NSGA2, the median scores of KEA-W are negative in all instances tested indicating

Table 17.6 Median and Inter-quartile Range (*IQR*) values of the Additive Epsilon Indicator obtained after 10 evaluations of KEA-W, KEA and NSGA2 on graphs with 100 nodes, $Eps = 0.0001$

Algorithms	*I*-values	Correlated Costs	Anti-Correlated Costs	Random Costs
I(KEA-W, KEA)	*Median*	**0.001**	**0.013**	**1.219**
I(KEA-W, KEA)	*IQR*	**0.010**	**0.014**	**2.201**
I(KEA, KEA-W)	*Median*	0.046	2.361	162.398
I(KEA, KEA-W)	*IQR*	0.019	0.791	36.196
I(KEA-W, NSGA2)	*Median*	**-0.381**	**-0.088**	**-2.434**
I(KEA-W, NSGA2)	*IQR*	**0.161**	**0.087**	**6.879**
I(NSGA2, KEA-W)	*Median*	1.179	3.398	301.227
I(NSGA2, KEA-W)	*IQR*	0.352	0.184	12.203
I(KEA-W, EPDA)	*Median*	**0.008**	**0.078**	**7.019**
I(KEA-W, EPDA)	*IQR*	**0.010**	**0.026**	**0.700**
I(EPDA, KEA-W)	*Median*	0.010	0.168	8.912
I(EPDA, KEA-W)	*IQR*	0.000	0.000	0.000

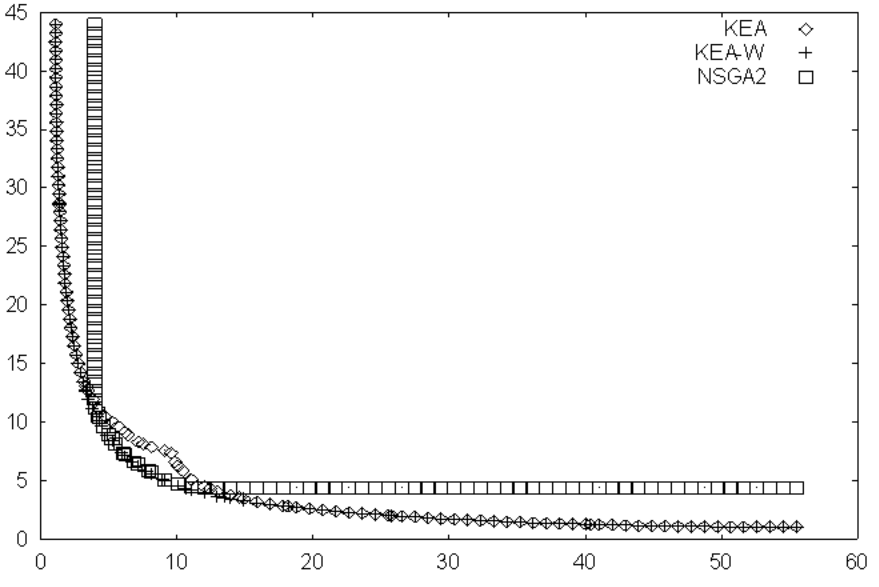


Fig. 17.12 Attainment surface plots obtained by KEA-W, KEA and NSGA2 with a graph with 100 nodes and anti-correlated costs, resolution 60, total number of test points 120, $Eps = 0.0001$

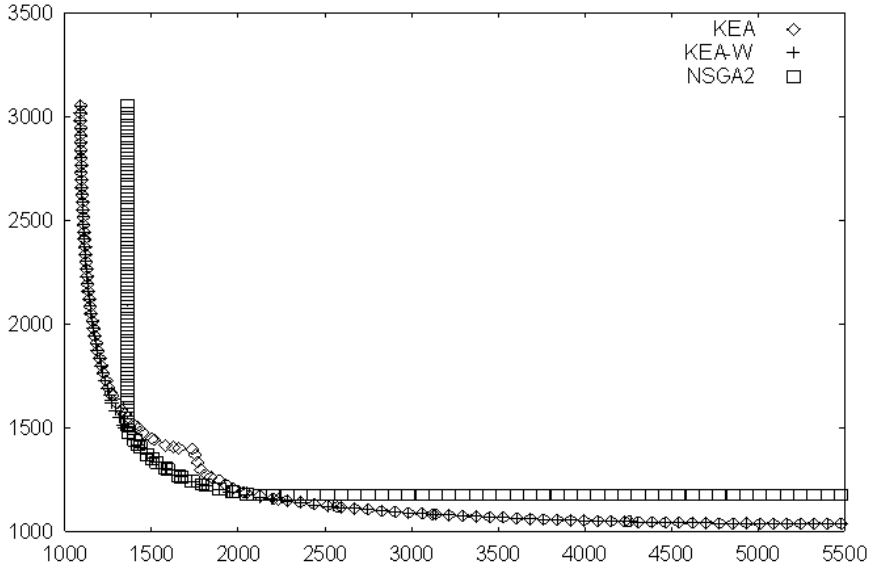


Fig. 17.13 Attainment surface plots obtained by KEA-W, KEA and NSGA2, with a graph with 100 nodes and random costs, resolution 60, total number of test points 120, $Eps = 0.0001$

that it is better than NSGA2 in a strict sense on more than 50% of runs. Therefore, the additive epsilon indicator confirms the superiority of KEA-W over KEA, EPDA and NSGA2 in all three instances.

Figures 17.12 and 17.13 compare the best attainment surface plots obtained by the three algorithms, KEA-W , KEA, and NSGA2 for the random and anti-correlated costs with $Eps = 0.0001$. These figures show that the middle section of each surface plot of KEA is dominated by KEA-W. NSGA2 mostly dominates the middle portion of KEA and is almost superimposed on the middle section of KEA-W. However, the tails of KEA-W dominate those of NSGA2. Therefore, the attainment surface plots also attest the superiority of KEA-W over KEA and NSGA2.

In order to discover how expensive these algorithms are if only an approximate PF is sought, all the four algorithms were executed for a limited time and their Attainment surface plots are compared in Figure 17.14. The solutions of NSGA2 are dominated by the other three techniques when the *CPU* time is limited. EPDA diverges away from the middle of the APS, so it is dominated by KEA which diverges less. At this middle point of the APS, KEA-W dominates all algorithms, but diverges on either side of the APS, where it is dominated by KEA and EPDA. Thus, when time is limited, a member of the KEA family can be selected depending on the part of the APS of interest.

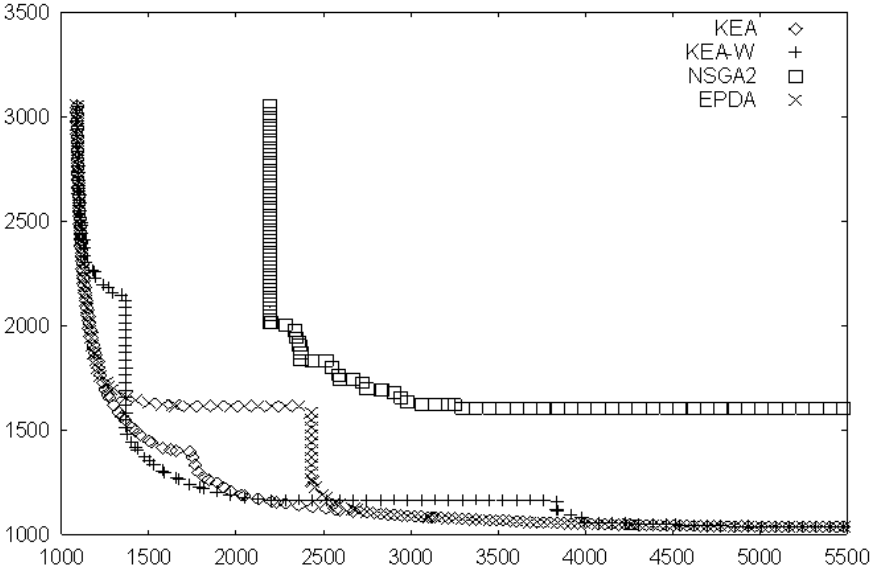


Fig. 17.14 Attainment surface plots for a graph with 100 nodes and random costs, where the execution time has been limited to approximately 515 sec. EPDA $FO = 2,782$, approximate KEA $FO = 4,317$, NSGA2 $FO = 21$, KEA-W $FO = 1,629$, resolution = 60, total no. of test points = 120, $Eps = 0.0001$

17.5 Discussions and Analysis

The main strength of the proposed approach of KEA is that it incorporates domain knowledge both prior and during the evolutionary process in order to get as close to the true PF as possible. The large spread and front occupation of KEA is due to its initial population that contains the extreme points of the PF and their non-dominated neighbors, most of which are optimal solutions. This elite initial population helps to concentrate the search in a specific area of the combinatorial search space. NSGA2 was also given a deterministic initial population consisting of all the n star trees that contain all the edges (genes).

Another reason for the speed of KEA is that it allows only mutations. A crossover operation, such as used in NSGA2, does not usually produce non-dominated solutions in the MCMST problem, so any associated algorithm will be slow to find non-dominated solutions.

Moreover, the mutation operator in KEA is knowledge-based, and adopts a greedy-type heuristic, which also helps to eliminate the calculation of dominated solutions. The advantage of heuristics is high performance in solving specific problems, but their disadvantage is in the low applicability. However, the underlying philosophy of KEA can be extended to other domains, in particular its method of exploring the multi-objective landscape.

One of the advantages of KEA is having an APS with flexible size. In most algorithms the APS is a vector with a fixed size. Therefore if the APS is full, when new solutions are found other solutions have to be discarded. This may be a disadvantage since valuable information is lost. Given the present computational memory capacity, a large APS should not cause computational problems. If in special applications there is a risk of memory getting saturated, the size of APS can be tailored accordingly. KEA controls the size of the APS in two ways. By varying the precision parameter, Eps, different sizes of APS can be obtained. Furthermore, the solutions with equal fitness functions are deleted from the APS.

Most algorithms proposed in literature have been tested with integer costs. In the past, to the best of our knowledge, only Kumar et al. [33], who have used real-valued data, have made observations about the effects of the precision parameter, Eps. Other researchers have fixed the precision parameter and therefore have not noted that with different values of the precision parameter they could have obtained different sized APSs in the objective space. It is emphasized that the precision parameter affects the Pareto front only in the objective space. The number of the optimal solutions in the decision space will not be affected; regardless of the value of the precision parameter. Nevertheless, since algorithms are compared with their APS in the objective space, it is important to note the value of the precision parameter used when results are reported to enable meaningful comparisons with future algorithms.

The only weakness of KEA lies in its dominated solutions in the middle of the APS in certain instances. This is because these points are the farthest away from the initial ancestors, the end points of the PF. Moreover, unlike EPDA that mutates systematically all the STs in the APS, KEA selects the STs randomly. This has a number of important effects. 1) KEA is substantially faster since not all STs are mutated. 2) At each iteration, dominated STs are eliminated, which again speeds up future processing, but at the cost that the non-dominated offspring of these STs cannot be produced. 3) Some existing non-dominated STs may not be selected in order to produce new non-dominated offspring. The last two effects result in the dominated middle section of the APS.

17.6 Conclusions

A fast knowledge-based Evolutionary Algorithm, called KEA, was presented for the multi-criterion minimum spanning tree problem. KEA was validated and tested using hard benchmark instances of the problem generated with algorithms from literature. The verification tests in the bi-criterion case against an exhaustive search algorithm, for complete graphs having four to ten nodes and three different cost functions, showed that KEA is capable of finding the true Pareto fronts.

KEA was compared with an adapted NSGA-II (NSGA2), on complete graphs with 20, 30, 50 and 100 nodes and three different cost functions. The approximate Pareto sets calculated by a deterministic algorithm (EPDA) were used as reference. It was shown that KEA outperforms NSGA2 in terms of speed, spread and front occupation. Further experiments with larger graphs of up to 200 nodes in two criteria

and up to 100 nodes in three criteria showed that it can obtain approximate Pareto sets that are almost as good as those obtained by EPDA in less time. The speed of KEA was demonstrated on problems with up to 500 nodes. The main advantages of KEA over its predecessors include its scalability to more than two criteria; its ability to calculate both the supported and the non-supported Pareto optimal solutions ; exploring regions of the search space that other algorithms do not thus finding evenly distributed Pareto fronts; and its speed making it applicable to problems with more than 500 nodes.

The only deficiency of KEA is that the middle section of its approximate Pareto sets for large graphs is dominated. In order to overcome this deficiency, a number of modifications were tested. It was found that KEA-W obtains the best results at the cost of large CPU times. Therefore when time is expensive and constitutes a limiting factor, KEA is still preferable over all the algorithms tested.

Although KEA has been tailored for the MCMST problem, its underling philosophy is anticipated to be applicable to other domains, where the calculation of extreme points is possible.

Acknowledgements

We are sincerely indebted to the reviewers for their comments and suggestions that have improved the quality of this chapter. Our thanks are also due to Dr. J. Knowles for making the software to calculate attainment surfaces to be plotted by the gnu plot available on his website.

References

1. Anderson, K., Jörnsten, A.K., Lind, M.: On bicriterion minimal spanning trees: An approximation. Comput. Oper. Res. 23, 1171–1182 (1996)
2. Arroyo, J.E.C., Vieira, P.S., Vianna, D.S.: A GRASP algorithm for the multi-criteria minimum spanning tree problem. In: Second Brazilian Symp. on Graphs (GRACO 2005), Rio de Janeiro, Brazil (2005); Also the Sec. Multidiscip. Conf. on Sched.: Theory and Apps., New York (2005)
3. Balakrishnan, A., Magnanti, T.L., Mirchandani, P.: Heuristics, LPs and trees on trees: Network design analyses. Ops. Res. 44, 478–496 (1996)
4. Bookstein, A., Klein, S.T.: Compression of correlated bit-vectors. Inf. Syst. 16, 110–118 (1996)
5. Bosman, P.A.N., Thierens, D.: Multi-objective optimization with diversity preserving mixture-based iterated density estimation evolutionary algorithms. Int. J. App. Reas. 31, 259–289 (2002)
6. Bosman, P.A.N., Thierens, D.: A thorough documentation of obtained results on real-valued continuous and combinatorial multi-objective optimization problems using diversity preserving mixture-based iterated density estimation evolutionary algorithms, Tech. Rep., Institute of Information and Computing Sciences, Utrecht University, The Netherlands (2002)
7. Chen, G., Chen, S., Guo, W., Chen, H.: The multi-criteria minimum spanning tree problem based genetic algorithm. Inf. Sci. 177, 5050–5063 (2007)

8. Christofides, N.: Graph Theory: An Algorithmic Approach. Academic Press Inc., London (1975)
9. Coello Coello, C.A.: A short tutorial on evolutionary multi-objective optimization. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 21–40. Springer, Heidelberg (2001)
10. Davis-Moradkhan, M., Browne, W.: A hybridized evolutionary algorithm for multi-criterion minimum spanning tree problem. In: Proc. 8th. Int. Conf. Hybrid Intell. Sys. (HIS 2008), pp. 290–295 (2008)
11. Davis-Moradkhan, M., Browne, W., Grindrod, P.: Extending evolutionary algorithms to discover tri-criterion and non-supported solutions for the minimum spanning tree problem. In: Proc. Genet. Evol. Comput. (GECCO 2009), Montréal, Canada, pp. 1829–1830 (2009)
12. Deb, K., Pratap, A., Agrawal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
13. Deb, K., Pratap, A., Agrawal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6, 181–197 (2002)
14. Ehrgott, M., Gandibleux, X.: Multiple Criteria Optimization: State of the Art. Annotated Bibliographic Surveys. Kluwer's International Series (2002)
15. Ehrgott, M., Skriver, A.J.: Solving biobjective combinatorial max-ordering problems by ranking methods and a two-phase approach. Eur. J. Oper. Res. 147, 657–664 (2003)
16. Flores, S.D., Cegla, B.B., Cáceres, D.B.: Telecommunication network design with parallel multi-objective evolutionary algorithms. In: Proc. IFIP/ACM Lat. Am. Conf.: Towards Lat Am Agenda Network Res, La Paz, Bolivia (2003)
17. Gabow, H.N.: Two algorithms for generating weighted spanning trees in order. SIAM J. Comput. 6, 139–150 (1977)
18. Gastner, M.T., Newman, M.E.J.: Shape and efficiency in spatial distribution networks. J. Stat. Mech. Theory & Exp. 1, 1015–1023 (2006)
19. Goldbarg, E.F.G., De Souza, G.R., Goldbarg, M.C.: Particle swarm optimization for the bi-objective degree-constrained minimum spanning tree. In: Proc. IEEE Congr. Evol. Comput., Vancouver, BC, Canada, pp. 1527–1534 (2006)
20. Guo, W., Chen, G., Feng, X., Yu, L.: Solving multi-criteria minimum spanning tree problem with discrete particle swarm optimization. In: Proc. 3rd Int. Conf. Nat. Comput. (ICNC 2007), pp. 471–478 (2007)
21. Hamacher, H.W., Ruhe, G.: On spanning tree problems with multiple objectives. An. Oper. Res. 52, 209–230 (1994)
22. Han, L., Wang, Y.: A novel genetic algorithm for multi-criteria minimum spanning tree problem. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005. LNCS (LNAI), vol. 3801, pp. 297–302. Springer, Heidelberg (2005)
23. López-Ibáñez, M., Paquete, L., Stützle, T.: On the design of ACO for the biobjective quadratic assignment problem. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 214–225. Springer, Heidelberg (2004)
24. Katoh, N., Ibaraki, T., Mine, H.: An algorithm for finding K minimum spanning trees. SIAM J. Comput. 10, 247–255 (1981)

25. Knowles, J.D.: Local search and hybrid evolutionary algorithms for Pareto optimization, Ph.D. Dissertation R8840, Department of Comput. Sci., University of Reading, Reading, UK (2002)
26. Knowles, J.D.: ParEGO: A Hybrid Algorithm with On-Line Landscape Approximation for Expensive Multi-objective Optimization Problems. IEEE Trans. Evol. Comput. 10, 50–66 (2006)
27. Knowles, J.D., Corne, D.W.: Approximating the non-dominated front using the Pareto archived evolution strategy. Evol. Comput. 8, 149–172 (2000)
28. Knowles, J.D., Corne, D.W.: A Comparison of Encodings and Algorithms for Multi-objective Minimum Spanning Tree Problems. In: Proc. Congr. Evol. Comput. (CEC 2001), pp. 544–551. IEEE Press, Los Alamitos (2001)
29. Knowles, J.D., Corne, D.W.: A Comparative Assessment of Memetic, Evolutionary, and Constructive Algorithms for the Multi-objective d-MST Problem. In: Proc. Genet. Evol. Comput. Conf. (Workshop WOMA II), Available on author's website (2001), http://dbk.ch.umist.ac.uk/klowles/
30. Knowles, J.D., Corne, D.W.: Benchmark Problem Generators and Results for the Multi-objective Degree-Constrained Minimum Spanning Tree Problem. In: Proc. Genet. Evol. Comput. Conf. (GECCO 2001), pp. 424–431. Morgan Kuafman Publishers, San Francisco (2001)
31. Kruskal Jr., K.B.: On the shortest spanning subtree of a graph and the travelling salesman problem. Proc. Amer. Math. Soc. 7, 48–50 (1956)
32. Kumar, R., Banerjee, N.: Multicriteria network design using evolutionary algorithms. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2724, pp. 2179–2190. Springer, Heidelberg (2003)
33. Kumar, R., Singh, P.K., Chakrabarti, P.P.: Multiobjective EA approach for improved quality of solutions for spanning tree problem. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 811–825. Springer, Heidelberg (2005)
34. Laumanns, M., Thiele, L., Zitzler, E.: An efficient, adaptive parameter variation scheme for metaheuristics based on the epsilon-constraint method. Eur. J. Oper. Res. 169, 932–942 (2006)
35. Michalewics, Z.: Genetic Algorithms + Dta Structures = Evolution Programs. Springer, U.S.A (1992)
36. Neumann, F., Wegener, I.: Minimum spanning trees made easier via multi-objective optimization. In: Beyer, H.-G. (ed.) Proc. Genet. Evol. Comput. (GECCO 2005), pp. 763–769 (2005)
37. Ng, H.S., Lam, K.P., Tai, W.K.: Analog and VLSI implementation of connectionist network for minimum spanning tree problems. In: Proc. IEEE Reg. 10 Int. Conf. Microelectron & VLSI (TENCON 1995), Hong Kong, pp. 137–140 (1995)
38. Obradović, N., Peters, J., Ružić, G.: Multiple communication trees of optimal circulant graphs with two chord lengths, Tech. Rep. SFU-CMPT-TR-2004-04, School of Comput. Sci., Simon Fraser University, Barnaby, British Colombia, V5A 1S6, Canada (2004)
39. Paquete, L., Stützle, T.: A two-phase local search for the biobjective traveling salesman problem. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 479–493. Springer, Heidelberg (2003)
40. Prim, R.C.: Shortest connection networks and some generalizations. Bell Syst. Tech. J. 36, 1389–1401 (1957)

41. Raidl, G.R., Julstrom, B.A.: Edge sets: An effective evolutionary coding of spanning trees. IEEE Trans. Evol. Comput. 7, 225–239 (2003)
42. Ramos, R.M., Alonso, S., Sicila, J., González, C.: The Problem of the optimal bi-objective spanning tree. Eur. J. Oper. Res. 111, 617–628 (1998)
43. Roy, B.: Méthodologie Multicritère d'Aide à la Décision, Ed. Économica, Paris (1985)
44. Schwefel, H.-P.: Evolution and Optimum Seeking. John Wiley & Sons, Inc., USA (1995)
45. Soak, S.-M., Corne, D., Ahu, B.-H.: A powerful new encoding for tree-based combinatorial optimization problem. In: Yao, X., Burke, E.K., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J.E., Tiňo, P., Kabán, A., Schwefel, H.-P. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 430–439. Springer, Heidelberg (2004)
46. Srinivas, N., Deb, K.: Multi-Objective Function Optimization Using Non-Dominated Sorting Genetic Algorithm. Evol. Comput. 2, 221–248 (1995)
47. Steiner, S., Radzik, T.: Solving the bi-objective minimum spanning tree problem using a k-best algorithm. Tech. Rep. TR-03-06, Department of Comput. Sci., King's College London (2003)
48. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans. Evol. Comput. 3, 257–271 (1999)
49. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multi objective optimizers: An analysis and review. IEEE Trans. Evol. Comput. 7, 117–132 (2003)
50. Zhou, G., Gen, M.: Genetic algorithm approach on multi-criterion minimum spanning tree problem. Eur. J. Oper. Res. 114, 141–152 (1999)

Chapter 18
Loss-Based Estimation with Evolutionary Algorithms and Cross-Validation

David Shilane, Richard H. Liang, and Sandrine Dudoit

Abstract. Statistical estimation in multivariate data sets presents myriad challenges when the form of the regression function linking the outcome and explanatory variables is unknown. Our study seeks to understand the computational challenges of regression estimation's underlying optimization problem and design intelligent procedures for this setting. We begin by analyzing the size of the parameter space in polynomial regression in terms of the number of variables and the constraints on the polynomial degree and the number of interacting explanatory variables. We subsequently propose a new procedure for statistical estimation that relies upon cross-validation to select the optimal parameter subspace and an evolutionary algorithm to minimize risk within this subspace based upon the available data. This general purpose procedure can be shown to perform well in a variety of challenging multivariate estimation settings. Furthermore, the procedure is sufficiently flexible to allow the user to incorporate known causal structures into the estimate and to adjust computational parameters such as the population mutation rate according to the problem's specific challenges. Furthermore, the procedure can be shown to asymptotically converge to the globally optimal estimate. We compare this evolutionary algorithm to a variety of competitors over the course of simulation studies and in the context of a study of disease progression in diabetes patients.

18.1 Introduction

Many statistical inference methods rely on selection procedures to estimate a parameter of the joint distribution of the data structure $X = (W, Y)$ that consists of explanatory variables $W = (W_1, \ldots, W_J)$, $J \in \mathbb{Z}^+$, and a scalar outcome Y. The *parameter of interest* often takes the form of a functional relationship between the outcome and explanatory variables, as in the regression setting's estimation of $E[Y|W]$, the conditional expectation of the outcome given a set of covariates. In loss-based estimation,

David Shilane
e-mail: dshilane@stanford.edu

Richard H. Liang
e-mail: rhliang@berkeley.edu

Sandrine Dudoit
e-mail: sandrine@stat.berkeley.edu

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 453–484.
springerlink.com

the parameter of interest is defined as the risk minimizer for a user-supplied loss function. Risk optimization is particularly challenging for high-dimensional estimation problems because it requires searching over large parameter spaces to accommodate general regression functions with possibly higher-order interactions among explanatory variables. We will show that the size of the parameter space in polynomial regression grows at a doubly exponential rate. In light of this expensive optimization problem, we seek new procedures that rely upon computational intelligence to provide accurate statistical estimates. In this study, we propose an evolutionary algorithm (EA) for risk optimization that may be applied in the context of loss-based estimation with cross-validation.

Statistical estimation in regression settings may consider a variety of approaches. For a fixed regression function relating the outcome and explanatory variables, a least squares approach [12] may be employed when the sample size is large relative to the number of explanatory variables. When the sample size is insufficient for the dimension of the problem, sparse regression procedures such as the Lasso [19], Least Angle Regression [10], or the Dantzig Selector [4] may be applied as shrinkage procedures. By including only a small number of explanatory variables as main effects, these estimators aim to produce interpretable models of the outcome. By contrast, other estimators focus solely on their predictive value; these include Classification and Regression Trees [3], Random Forests [2], Multivarite Adaptive Regression Splines [13, 14], and neural network approaches [17]. Loss–based estimation with cross–validation seeks to balance the goals of producing interpretable models and reliable estimates with predictive value by selecting among a variety of candidate estimators in terms of their fit on an independent validation set. Like shrinkage procedures, loss–based estimation with cross-validation results in some degree of variable selection but also allows for the exploration of higher–order variable interactions. Furthermore, cross–validation can be shown to be an asymptotically optimal selection procedure in terms of the sample size [7, 15]. Because of its strong theoretical properties and its utility in producing reliable and interpretable estimates, we will rely upon cross–validation as a general method for estimation and propose a new procedure for risk optimization to be used within this context.

The proposed methodology is motivated by the general road map for statistical loss–based estimation using cross–validation of van der Laan and Dudoit [15] and Dudoit and van der Laan [7]. Risk optimization may be considered a sub–problem of this road map. Sinisi and van der Laan [16] introduced a general Deletion/Substitution/Addition (DSA) algorithm for generating candidate estimators that seek to minimize empirical risk over subspaces demarcated by basis size (Section 18.3.1). However, Wolpert and MacReady [20] have shown that no single optimization algorithm can competitively solve all problems; therefore, we are interested in generating complementary risk optimization algorithms for use in estimator selection procedures. Within the estimation road map [7, 15], this project seeks to analyze the size of the parameter space for a polynomial regression function in terms of the number of explanatory variables, the maximum number of interacting variables, and either the polynomial degree or the variable degree. It also introduces an EA to generate candidate estimators and minimize empirical risk within parameter subspaces. Relying upon V-fold cross-validation to select an optimal parameter subspace, the

procedure effectively estimates the parameter of interest in a manner that seeks to minimize true risk.

The proposed EA for estimator selection includes a stochastic mutation mechanism that can be shown to assign all candidate estimators within a parameter subspace to a single communicating class. By doing so, the EA prevents the risk optimization from becoming trapped at local optima. An elitist selection procedure is employed to retain the best candidate estimator among those considered at every generation of the evolution process. When all candidate estimators form a single communicating class and the selection mechanism is elitist, the optimization algorithm converges asymptotically in generation to the global optimum [11]. The EA also includes computational parameters such as the population size and mutation probability that may be varied to produce an arbitrary number of different optimization routines; this allows the user to tailor the procedure to the problem at hand. The proposed algorithm may also be applied to general parameterizations and loss functions. Finally, the EA is modular in its design, so the user may easily substitute alternative components according to situational needs. As a result, the proposed estimator selection procedure is widely applicable in scientific settings such as regression studies in biology and public health. In addition to studying the proposed algorithm's efficacy through simulation experiments, we investigate a diabetes data set to explore the combination of factors contributing to the progression of the disease in human patients.

Section 18.2 summarizes loss-based estimation with cross-validation and outlines our procedure for estimator selection. Section 18.3 reviews the parametrization of polynomial basis functions, considers constraints that may be imposed on the degree and interaction order of these basis functions, and analyzes the size of the parameter space under specific combinations of these constraints. Section 18.4 proposes a new EA for risk optimization that may be applied within the procedure of Section 18.2.2. Section 18.5 compares the performance of the proposed EA to that of the DSA in simulation experiments. We then apply these techniques in Section 18.6 to estimate a regression function in the context of a diabetes study. Section 18.7 concludes the study with a discussion, and Section 18.8 is an appendix of further details for the analysis of the parameter space.

18.2 Loss-Based Estimation with Cross-Validation

18.2.1 The Estimation Road Map

As summarized by Dudoit and van der Laan [7], we assume that the data X are generated according to a distribution P belonging to a statistical model $\mathscr{M}$, which is a set of possibly non-parametric distributions. Consider a parameter mapping $\Psi : \mathscr{M} \to \mathscr{F}(\mathscr{D},\mathscr{R})$ from the model $\mathscr{M}$ into a space $\mathscr{F}(\mathscr{D},\mathscr{R})$ (or $\mathscr{F}$ in short) of functions with domain $\mathscr{D}$ and range $\mathscr{R}$. A *parameter* is a realization $\psi \equiv \Psi(P) : \mathscr{D} \to \mathscr{R}$ of Ψ for data generating distribution P. The *parameter space* is defined as $\Psi \equiv \{\psi = \Psi(P) : P \in \mathscr{M}\} \subseteq \mathscr{F}$. Given a random variable X and a parameter value ψ, a *loss function* $L(X,\psi) : (X,\psi) \to \mathbb{R}$ is a measure of distance between the parameter and the data. Given a data generating distribution P, we can

summarize loss in terms of a corresponding *risk function*, which is defined as the expected value of the loss function:

$$\Theta(\psi, P) \equiv \int L(x, \psi) dP(x) = E[L(X, \psi)]; \qquad P \in \mathscr{M}. \tag{18.1}$$

It is assumed that the loss function is specified such that the parameter of interest ψ minimizes the risk function. For instance, in regression, the parameter of interest is the regression function $\psi(W) = E[Y|W]$, which minimizes risk for the L_2 loss function $L(X, \psi) = (Y - \psi(W))^2$. We can then define the optimal risk over the parameter space as:

$$\theta \equiv \Theta(\psi, P) = \min_{\psi' \in \Psi} \Theta(\psi', P) = \min_{\psi' \in \Psi} \int L(x, \psi') dP(x). \tag{18.2}$$

Given $\mathscr{X}_n$, a set of n independent, identically distributed (i.i.d.) observations of data $X_i = (W_i, Y_i)$, $i \in \{1, \ldots, n\}$, from (typically unknown) distribution P, our goal is to select an estimator $\psi_n = \hat{\Psi}(P_n)$ based upon the empirical distribution P_n of the data $\mathscr{X}_n$ in a manner that seeks to minimize true risk. The empirical risk of a parameter value ψ is defined as:

$$\Theta(\psi, P_n) \equiv \int L(x, \psi) dP_n(x) = \frac{1}{n} \sum_{i=1}^{n} L(X_i, \psi). \tag{18.3}$$

The general road map for loss-based estimation [7, 8, 15] contains three steps:

1. *Define the parameter of interest.* This parameter is the value that minimizes risk for a user-supplied loss function.
2. *Generate candidate estimators.* The parameter space is divided based upon a sieve of increasing dimensionality into subspaces whose union approximates the complete parameter space. Within each subspace, a candidate estimator is chosen to minimize empirical risk.
3. *Apply cross-validation.* Select the optimal estimator among the candidates produced in Step 2 using cross-validation.

18.2.2 Estimator Selection Procedure

An estimator ψ_n seeking to minimize *empirical risk* $\Theta(\psi_n, P_n)$ on the learning set $\mathscr{X}_n$ is prone to over-fitting at the expense of predictive value. The general road map for loss-based estimation [7] resolves the issue of over-fitting by first selecting a subspace of the parameter space using a cross-validation procedure and then selecting the optimal estimator to minimize empirical risk on the learning set over this subspace. We may apply the estimation road map in the following estimator selection procedure:

1. The user specifies a set $\mathscr{K}$ of candidate subsets of the parameter space to be searched. By default, the subspaces are indexed by $\mathscr{K} = \{1, \ldots, K\}$, with $K \in \mathbb{Z}^+$.
2. The user specifies $V \in \mathbb{Z}^+$, the number of folds to use in cross-validation. Although we employ V-fold cross-validation in this procedure, it should be noted that alternative cross-validation procedures such as Monte-Carlo cross-validation may be substituted [7].

3. Data points $X_i = (W_i, Y_i)$, $i \in \{1, \ldots, n\}$, from the learning set $\mathscr{X}_n$ are randomly assigned to a class in $\{1, \ldots, V\}$ such that each class contains an approximately equal number of observations. Let $Q = (q_1, \ldots, q_n)$ refer to the data's class assignments.
4. For each fold $v \in \{1, \ldots, V\}$:
   a. Assign data points to the training set:
   $$\mathscr{T}_n(v) = \{X_i : q_i \neq v\}. \tag{18.4}$$
   b. Assign data points to the validation set:
   $$\mathscr{V}_n(v) = \{X_i : q_i = v\}. \tag{18.5}$$
   c. For each candidate subspace $k \in \mathscr{K}$:
      i. Search within the subspace for the candidate estimator $\psi_{k,n}$ that minimizes empirical risk on the training set $\mathscr{T}_n(v)$.
      ii. Compute the validation set risk $\Theta\left(\psi_{k,n}, P_n^{\mathscr{V}_n(v)}\right)$, where $P_n^{\mathscr{V}_n(v)}$ represents the empirical distribution on the validation set $\mathscr{V}_n(v)$.
5. Calculate the *mean cross-validated risk* for each subspace and store it in the vector
$$\Theta^{CV} \equiv \left(\Theta_1^{CV}, \ldots, \Theta_K^{CV}\right) = \left(\frac{1}{V}\sum_{v=1}^{V}\Theta\left(\psi_{1,n}, P_n^{\mathscr{V}(v)}\right), \ldots, \frac{1}{V}\sum_{v=1}^{V}\Theta\left(\psi_{K,n}, P_n^{\mathscr{V}(v)}\right)\right). \tag{18.6}$$
6. Select the subspace that minimizes mean cross-validated risk:
$$k_n = \underset{k\in\{1,\ldots,K\}}{\operatorname{argmin}}\ \Theta_k^{CV}. \tag{18.7}$$
7. Finally, search within the parameter subspace k_n for the estimate ψ_n minimizing empirical risk $\Theta(\psi_n, P_n)$ on the learning set data $\mathscr{X}_n$.

Steps 4(c)i and 7 of the above procedure rely upon searching a parameter subspace for the estimator that minimizes empirical risk when applied to the specified (training or learning) data set. An exhaustive search of the parameter subspace may be employed when doing so is computationally tractable. However, in estimation problems over general regression functions with possibly higher-order interactions, the parameter space can grow complex and large (Section 18.3.3) for even a moderate number of explanatory variables. We therefore require a *search algorithm* to minimize risk within a parameter subspace in the allotted computational time. The DSA [16] is one candidate search algorithm; Section 18.4 will introduce a class of evolutionary algorithms as an alternative procedure for risk minimization.

18.3 The Parameter Space for Polynomial Regression

18.3.1 Parametrization

Given a parameter of interest ψ and a suitable loss function $L(X, \psi)$, we seek to characterize the set of candidate estimators to be searched by an estimator selection procedure. In regression, the *parameter space* can be defined by the class of *basis functions* for the explanatory variables (e.g. polynomial functions or set indicators),

the choice of the *link function* (e.g. logit or probit) mapping the selected basis functions to the outcome variable, and the *constraints* that limit the way in which explanatory variables may interact. Much as in Sinisi and van der Laan [16], the proposed estimator selection procedure may be applied to any estimation setting, including but not limited to robust and weighted regression, censored data structures, and generalized linear models for any choice of link function $h : \mathbb{R} \to \mathbb{R}$. Because of its approximation capabilities [18], we will focus on the parameter space consisting of the set of polynomial combinations of the explanatory variables with real-valued coefficients. In this parametrization, the set of basis functions $\boldsymbol{\Phi}$ consists of all monomial functions ϕ that can be expressed in terms of an exponent vector $\mathbf{d} = (d_1, \ldots, d_J)$ as

$$\phi = W^{\mathbf{d}} \equiv W_1^{d_1} \cdots W_J^{d_J}. \tag{18.8}$$

A parameter value ψ may be specified in terms of a subset of basis functions:

$$\varphi = \left\{ \phi_{i_1}, \ldots, \phi_{i_k} \right\} \subseteq \boldsymbol{\Phi}, \tag{18.9}$$

with cardinality $|\varphi| = k$ referred to as the *basis size*. Given the link function h, a size k set of basis functions φ, and a $(k+1)$-dimensional vector $\beta = (\beta_0, \beta_1, \ldots, \beta_k)$ of real-valued coefficients, a regression function for ψ has the form

$$\psi = h\left(\beta_0 + \sum_{i:\phi_i \in \varphi} \beta_i \phi_i \right). \tag{18.10}$$

In seeking an estimate ψ_n that minimizes true risk, we will first search for the optimal set of basis functions φ_n and then subsequently seek an optimal estimate β_n of β. Estimating β given φ_n is a standard regression problem that is solved in a closed form for linear regression and with numeric optimization methods for non-linear regression. Given φ_n and β_n, the estimate ψ_n is defined as:

$$\psi_n = h\left(\beta_{0_n} + \sum_{i:\phi_i \in \varphi_n} \beta_{i_n} \phi_i \right). \tag{18.11}$$

18.3.2 Constraints

At the user's discretion, constraints may be imposed on the set of basis functions $\boldsymbol{\Phi}$. These constraints may take the form of limits on the interaction order and the polynomial or variable degree. The interaction order constraint, which limits the number of variables that interact in a basis function, may be stated as:

$$1 \leq \sum_{j=1}^{J} \mathbf{1}\{d_j > 0\} \leq S; \qquad S \in \mathbb{Z}^+. \tag{18.12}$$

A polynomial degree constraint may be phrased in the form:

$$1 \leq \sum_{j=1}^{J} d_j \leq D; \qquad d_j \in \mathbb{Z}^+,\ j \in \{1, \ldots, J\}. \tag{18.13}$$

A variable degree constraint, which may be used as an alternative to the polynomial degree constraint, allows each component variable in a basis function to independently attain a maximum degree D_0. The variable degree constraint is:

$$0 \leq d_j \leq D_0; \qquad j \in \{1,\ldots,J\} \text{ with } \sum_{j=1}^{J} d_j \geq 1. \tag{18.14}$$

Although the constraints (18.12), (18.13), and (18.14) are not required, they allow the researcher to restrict attention to a particular subset of the class of chosen basis functions. By default, the interaction order S can be no greater than $\min(J,D)$ under constraint (18.13) and is limited to J under (18.14). The extreme cases $S = 1$ and either $S = \min(J,D)$ or $S = J$ correspond, respectively, to constraints allowing no interactions and interactions of any order.

18.3.3 Size of the Parameter Space

Under the above formulation, the set of basis functions Φ consists of all unique monomials ϕ corresponding to an exponent vector $\mathbf{d} = (d_1,\ldots,d_J)$ satisfying the interaction order constraint (18.12) and either the polynomial degree constraint (18.13) or the variable degree constraint (18.14). We can then determine the number I of basis functions in Ψ using combinatorial arguments. When subject to the polynomial degree constraint (18.13), the value of I is given by:

$$I = \sum_{s=1}^{S} \binom{J}{s} \left[1 + \sum_{d=s+1}^{D} \sum_{k=1}^{\min(s,d-s)} \binom{s}{k} \binom{d-s-1}{k-1} \right]. \tag{18.15}$$

The sum from $s = 1$ to S represents all possible values for the number of variables to appear in a monomial. Once this is selected, the $\binom{J}{s}$ term provides the number of ways to choose s variables $W_{j_1},\ldots,W_{j_s}$ from the J total variables. Given $W_{j_1},\ldots,W_{j_s}$, we turn our attention to the number of valid monomials using exactly all of these variables. Because the multilinear term $W_{j_1}W_{j_2}\cdots W_{j_s}$ is always included, the number of valid monomials is 1 plus the number of higher-order monomials. The sum from $d = s+1$ to D represents the choice of higher order polynomial degree for the monomial in the set $\{s+1,\ldots,D\}$. Once we have chosen a degree, we must distribute it over all of the selected variables. Knowing that each of these s variables must have a degree of at least one, this leaves $d-s$ powers to distribute to s variables. The sum over k chooses the number of monomial variables allocated a higher degree. A total of $\binom{s}{k}$ combinations of variables may receive higher power. Finally, by a well-known result in combinatorics, the number of ways to distribute at least one degree to each of the k variables selected to receive more power is $\binom{d-s-1}{k-1}$.

As an example, suppose we assign $S = 1$ and impose constraint (18.12) to preclude all variable interactions. In this case, the summations over s and k involve only a single iteration, leaving us with $\binom{J}{1}\left[1+\sum_{d=2}^{D}\binom{1}{1}\binom{d-2}{0}\right] = \binom{J}{1}[1+(D-1)] = JD$.

That is, when no variables may interact, the set of possible basis functions consists of all choices of a single variable W_j, $j \in \{1,\ldots,J\}$, raised to a power $d_j \in \{1,\ldots,D\}$.

When the parameter space is instead restricted by the variable degree constraint (18.14), the number of basis functions is I_0:

$$I_0 = \sum_{s=1}^{S} \binom{J}{s} D_0^s. \tag{18.16}$$

In (18.16), all allowed basis functions may be categorized by the number of interacting variables $s \in \{1,\ldots,S\}$. Given s, a total of $\binom{J}{s}$ combinations of variables may be selected to interact. Each of these interacting variables must have a positive degree in the set $\{1,\ldots,D_0\}$ that may be assigned independently. For a given regression setting subject to the above constraints, the number of basis functions I (18.15) (or I_0 (18.16)), provides an indication of the problem's size and may be used to guide the selection of computational parameters in the risk optimization algorithm. Because each basis function may be included in or excluded from a regression function, there are 2^I (or 2^{I_0}) possible estimators among which to choose. Within a size k parameter subspace, exactly k basis functions are included in any estimator, so the subspace contains $\binom{I}{k}$ or $\binom{I_0}{k}$ estimators, respectively. We can analyze the size of the parameter space by providing upper and lower bounds on the number of basis functions. Using the notation of Cormen et al [6], the functions Ω and O may be used to specify asymptotic lower and upper bounds on the order of the number of basis functions I or I_0 in terms of the interaction order bound S and the polynomial degree bound D or variable degree bound D_0, respectively. (Please refer to the Appendix for details.) When subject to the interaction order constraint (18.12) and the polynomial degree constraint (18.13), the value of I is bounded below by a function of order $\Omega\left(2^S\right)$. When the variable degree constraint (18.14) is employed subject to the interaction order constraint (18.12), the value of I_0 is bounded below by a function that is $\Omega\left(D_0^S\right)$. When no interaction order constraint is imposed, then $S = \min(J,D)$ under the constraint (18.13), and so the lower bound on I is $\Omega\left(2^{\min(J,D)}\right)$. Likewise, when no interaction order constraint is imposed under the constraint (18.14), then $S = J$, and I_0 is bounded below by $\Omega\left(D_0^J\right)$. Because all interactions are allowed when $S = J$, the summation for I_0 in (18.16) may be expressed as a polynomial in D_0 of degree J. Therefore, when $S = J$, I_0 is both $\Omega\left(D_0^J\right)$ and $O\left(D_0^J\right)$, which jointly imply a tight bound on I_0. The latter bound may also be used as a loose upper bound when $S < J$. Furthermore, because any basis function allowed under the constraints (18.12) and (18.13) is also permitted under (18.12) and (18.14) when $D = D_0$, this upper bound on I_0 is also a trivial upper bound on I. Therefore, I is $O\left(D^J\right)$. Finally, because the size of the parameter space is 2^I or 2^{I_0}, these quantities are respectively bounded below by functions of order $\Omega\left(2^{2^S}\right)$ and $\Omega\left(2^{D_0^S}\right)$. Likewise, upper bounds of $O\left(2^{D^J}\right)$ and $O\left(2^{D_0^J}\right)$ may also be established, where the former is a trivial bound, and the latter is a loose bound that is only tight in the extreme case of $S = J$. These results are proved in the Appendix and summarized in Table 18.1.

Table 18.1 Size of the parameter space under the interaction order constraint (18.12) and either the polynomial degree constraint (18.13) or the variable degree constraint (18.14)

	Polynomial Degree Constraint (18.13)	Variable Degree Constraint (18.14)
Upper Bound	$O\left(2^{D^J}\right)$	$O\left(2^{D_0^J}\right)$
Lower Bound	$\Omega\left(2^{2^S}\right)$	$\Omega\left(2^{D_0^S}\right)$

Because the size of the parameter space is at least of a doubly exponential order of the number of variables J and the polynomial degree bound D or variable degree bound D_0 when the interaction order is not constrained, even moderate degree constraints imposed on a small number of variables may result in an intractable parameter space to search. In this setting, significant computation may be required to obtain a reliable estimate of the parameter of interest. Figure 18.1 depicts the growth of $\log(I)$ and $\log(I_0)$ as the polynomial degree bound D and the variable degree bound D_0 increase in an estimation setting with $J = 11$ variables and no interaction order constraint; i.e. $S = \min(J, D)$ for the constraint (18.13), and $S = J$ for the constraint (18.14). The approximately linear growth on the logarithmic scale confirms that the values I and I_0 are exponential functions of their respective degree bounds. The value I is consistently smaller than I_0 because the polynomial degree constraint (18.13) restricts the parameter space to a subset of that specified by the variable degree constraint (18.14). Furthermore, the maximum value of S under the constraint (18.14) is J, whereas S is constrained to $\min(J, D) \leq J$ under the constraint (18.13).

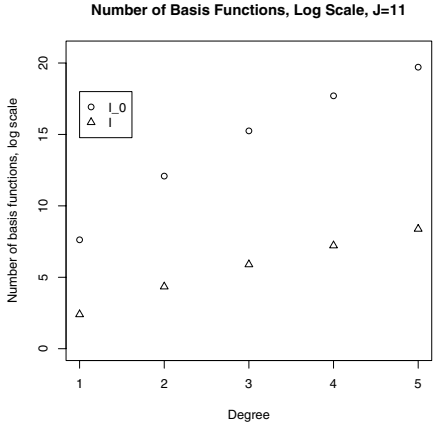


Fig. 18.1 The natural logarithm of the numbers of basis functions I and I_0 as a function of the polynomial degree bound D and the variable degree bound D_0, respectively, for $J = 11$ variables and no interaction order constraint. For the constraint (18.13), we have $S = \min(J, D)$, and for the constraint (18.14), this value is $S = J$

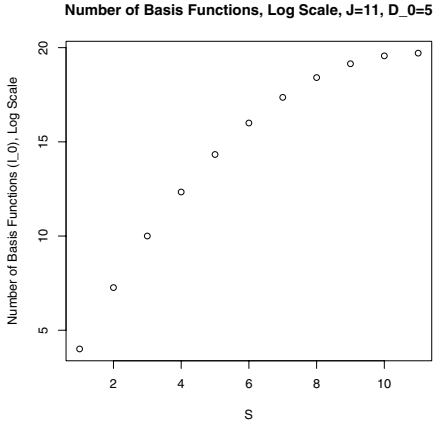


Fig. 18.2 The natural logarithm of the number of basis functions I_0 as a function of the interaction order bound S for $J = 11$ variables with the variable degree constraint (18.14) specified by $D_0 = 5$

Therefore, for a fixed level of the interaction order bound S, the polynomial degree constraint (18.13) always results in a smaller parameter space than that specified by the variable degree constraint (18.14) when $D = D_0$.

Figure 18.2 plots the growth of $\log(I_0)$ as a function of the interaction order bound S for an estimation setting including $J = 11$ variables and degree bound $D_0 = 5$, which corresponds to Model 5 presented in Section 18.5. In practice, S is often chosen according to scientific insight for the problem at hand. However, the choice of S can also be used to effectively prune the parameter space to a manageable size.

18.4 Evolutionary Algorithms as Risk Optimization Procedures

Evolutionary Algorithms (EA) comprise a class of stochastic optimization algorithms that generate candidate solutions via a process similar to biological evolution [1, 11]. Although Wolpert and MacReady [20] have shown that no single algorithm can best solve all optimization problems, EAs are sufficiently flexible to be applied to many types of problems, perform reasonably well in a variety of settings, and provide few difficulties in software development [11]. Furthermore, we can immediately generate many candidate algorithms for comparison by varying the proposed EA's computational parameters. In this section, we will familiarize the reader with EA methodology, elucidate the underlying stochastic process by which an EA seeks to optimize a function, and incorporate this class of algorithms as a search procedure in estimator selection (as in Section 18.2.2).

An EA seeks to optimize a real-valued *objective* (or *fitness*) function. When generating candidate estimators ψ_n, our objective is to minimize the risk $\Theta(\psi_n, P)$ over parameter subspaces, where, depending upon the context, P denotes the empirical

distribution for either the full learning set or a cross-validation training set. A candidate optimum of this fitness function is given by an *individual* consisting of a *genotype* vector $\mathbf{e} = (e_1, \ldots, e_{kJ}) \in (\mathbb{R}^+)^{kJ}$ and a corresponding *phenotype* vector:

$$\mathbf{d}=\mathbf{d}(\mathbf{e})\equiv\left([d(e_1,\ldots,e_J)],[d(e_{J+1},\ldots,de_{2J})],\ldots,\left[d(e_{(k-1)J+1},\ldots,e_{kJ})\right]\right)\in\left(\mathbb{R}^+\right)^{kJ}. \tag{18.17}$$

Each block of J phenotypic components $\left[d(e_{(j-1)J+1}),\ldots,d(e_{jJ})\right]$ serves as the exponent vector of a particular basis function, and the k basis functions collectively specify a subset φ of the form (18.9) that map to a candidate optimum ψ_n. An individual's fitness is given by the risk $\Theta(\psi_n, P)$ of its associated estimate ψ_n. Although the user may proceed by directly specifying a phenotype vector, a data structure including both a genotype and a phenotype allows for a greater variety of evolutionary information to be stored in an individual. For instance, a continuous genotype may be used to break ties in the phenotype when an interaction order constraint is imposed. In this setting, the elements of the genotype vector $\mathbf{e}$ may belong to the positive real numbers $\mathbb{R}^+$, the elements of the phenotype vector $\mathbf{d}$ may be limited to the set of positive integers $\mathbb{Z}^+$, and any function with domain $\mathbb{R}^+$ and range $\mathbb{Z}^+$ may be used to map from an individual's genotype to its phenotype. In the procedure of Section 18.4.1, we choose this function according to the selected degree and interaction order constraints. When an interaction order constraint is imposed, a continuous genotype structure allows some genes to maintain a genotype while remaining dormant in terms of phenotype. In this scenario, if a gene whose phenotype previously interacted mutates, a gene of dormant phenotype may immediately take its place as one of the at most S interacting phenotypic components of a given basis function. Furthermore, a data structure incorporating both a genotype and a phenotype generalizes the EA so that it may be tailored to a particular constraint profile (e.g. those of Section 18.3.2) solely through the choice of the phenotype function.

Starting from a random *initial population*, EAs typically generate subsequent *populations* of individuals in *generations* of *offspring* created from existing *parents* via iterations of *evolutionary mechanisms*. Although other mechanisms may be used, each generation of the proposed EA consists of a reproduction, mutation, and selection phase, and these mechanisms collectively create and evaluate new individuals for quality in terms of fitness. After allowing the population to evolve for $G \in \mathbb{Z}^+$ generations, the individual with optimum observed fitness is retained as the algorithm's *result*, which specifies an estimate ψ_n with an associated risk given by $\Theta(\psi_n, P)$.

18.4.1 Proposed EA

The following EA is used to optimize risk within a parameter subspace of size $k \in \mathbb{Z}^+$ based on monomial basis functions of the J explanatory variables under the interaction order constraint (18.12) and either the polynomial degree constraint (18.13) or the variable degree constraint (18.14). A schematic diagram of this

algorithm is depicted in Figure 18.3. Each step of the algorithm is first summarized here and then further elucidated below.

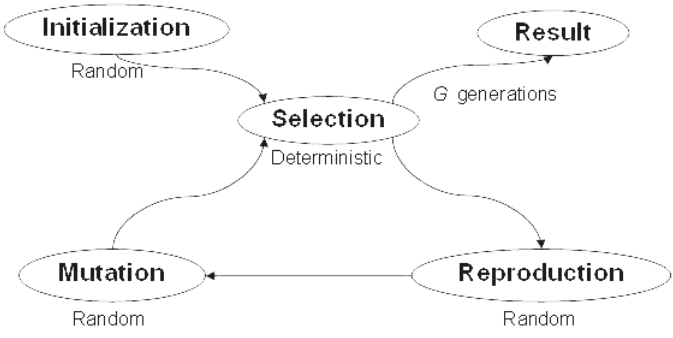


Fig. 18.3 Schematic diagram for the proposed EA risk optimization procedure

1. **Initialization:** Create a population of candidate solutions.
2. **Evolution:** Create a new population from the existing population. The three evolutionary mechanisms are performed in order at each of G generations:
   a. **Selection:** Rank individuals according to fitness and select a proportion of the population to survive and mate.
   b. **Reproduction:** Pair selected individuals and create offspring via a random combination of parental genotypes.
   c. **Mutation:** Alter offspring genotypes according to a stochastic mutation process.
3. **Result:** After G generations of evolution, perform the selection mechanism on the resulting population and return the individual with optimal fitness as the EA's result. This individual's phenotype vector $\mathbf{d}$ has an associated subset of basis functions φ_n that correspond to an estimate ψ_n according to (18.11).

Mapping from Genes to Estimates: The following steps may be used to map from a genotype vector $\mathbf{e}$ to a candidate estimate ψ_n:

1. Each individual contains k blocks of J genes. Rank gene values within each block in decreasing order in a vector $\mathbf{r}$:

$$\mathbf{r} = \left((r_1,\ldots,r_J),(r_{J+1},\ldots,r_{2J}),\ldots,\left(r_{(k-1)J+1},\ldots,r_{kJ}\right)\right) \tag{18.18}$$

   so that each block $(r_{(j-1)J+1},\ldots,r_{jJ})$, $j \in \{1,\ldots,k\}$, contains a permutation of $\{1,\ldots,J\}$. When the interaction order constraint (18.12) is employed, this ranking is used to select the basis function corresponding to the (at most) S genes of largest value in each block.

2. Calculate the individual's phenotype vector $\mathbf{d}$ from its genotype $\mathbf{e}$. This computation differs depending on which degree constraint is used. However, this step is the only location within the EA for which the procedure differs depending upon the constraint. This is an additional advantage of a data structure that includes both a genotype and a phenotype.
   - For the polynomial degree constraint (18.13): Within each block of J genes of the genotype vector $\mathbf{e}$, begin with the variable of highest rank (corresponding to the largest gene value). Assign the minimum of the floor of this gene value and the remaining polynomial degree as the variable's phenotype within the block. Repeat this procedure on each variable in order of its gene rank until the monomial is of degree D or the interaction order constraint (18.12) is binding.
   - For the variable degree constraint (18.14): Within each block of J genes, compute the phenotype by assigning the floor of the genotype for each of the at most S interacting variables. (Because all gene values are within the interval $(0, D_0 + 1)$, the floor function ensures that no phenotype exceeds D_0.) All non-interacting variables receive phenotype 0. This computation may be performed via the following equation:

$$\mathbf{d} = \lfloor \mathbf{e} \rfloor \mathbf{1}\{\mathbf{r} \leq S\} = (\lfloor e_1 \rfloor \mathbf{1}\{r_1 \leq S\}, \ldots, \lfloor e_{kJ} \rfloor \mathbf{1}\{r_{kJ} \leq S\}). \tag{18.19}$$

   In order to ensure that the resulting monomials are all of degree at least 1 under each of the above degree constraints, the variable of highest rank within each block of J genes may receive a phenotype of 1 when all gene values within the block are less than 1.
3. Given a phenotype vector $\mathbf{d}$, an individual has an associated subset of basis functions φ_n. Calculate from φ_n the corresponding estimate ψ_n according to (18.11).

Initialization: The user may specify the number $Z \geq 4$, $Z \in \mathbb{Z}^+$, of individuals in the initial population. Recall that the genotype vector $\mathbf{e}$ for an individual has length kJ. Initialization consists of generating genotype vectors for each individual from a (kJ)-variate uniform distribution on $(0, D+1)^{kJ}$ or $(0, D_0+1)^{kJ}$.

Selection: Given a population of individuals, each with an associated estimate ψ_n, we will rank individuals according to fitness via the following procedure:

1. Compute empirical risk $\Theta(\psi_n, P)$, where P is the empirical distribution with respect to either a cross-validation training data set $\mathscr{T}_n$ or the learning set $\mathscr{X}_n$.
2. Rank existing individuals in order of increasing empirical risk. Select the $2\lfloor Z/4 \rfloor$ individuals with smallest empirical risk for reproduction. We will refer to the ranked population as $(\mathbf{e}[1], \ldots, \mathbf{e}[Z])$, with each $\mathbf{e}[z]$ mapping to the gene vector $\mathbf{e}$ for the individual with the zth smallest risk.

An individual may be considered *cumulatively optimal* at generation g if its associated estimate ψ_n has a smaller risk than that of any other individual produced in the first g generations. The proposed selection mechanism is *elitist* in the sense that the cumulatively optimal individual is always selected at each generation. (Indeed, if the cumulatively optimal individual at generation g is not selected at generation

$g+1$, then it is supplanted by some other individual with an associated estimate of smaller risk. This new individual is then cumulatively optimal at generation $g+1$.) As a result, the associated estimate of the EA's cumulatively optimal individual is monotonically non-increasing in risk as a function of the generation of evolution.

Reproduction: Create offspring from selected individuals via the following procedure:

1. Assign selected individuals to mating pairs in order of increasing empirical risk. The individuals $\mathbf{e}[1]$ and $\mathbf{e}[2]$ are assigned to a mating pair, and the process is repeated on the remaining population until $\lfloor Z/4 \rfloor$ pairs of individuals have been assigned.
2. Breed each mating pair to produce $C = 2$ offspring. For each child $c \in \{1, \ldots, C\}$, generate a Bernoulli(p) random variable γ_c. The default value of p is 0.5.
3. Given the population sorted in order of fitness, then, for $z \in \{1, \ldots, \lfloor Z/4 \rfloor\}$, construct each child's genotype vector as:

$$\begin{aligned} \mathbf{e}\left[\lfloor Z/2 \rfloor + 2(z-1) + c\right] &= (\gamma_c)\, pmax\left(\mathbf{e}[2z-1], \mathbf{e}[2z]\right) \\ &\quad + (1-\gamma_c)\, pmin\left(\mathbf{e}[2z-1], \mathbf{e}[2z]\right). \end{aligned} \tag{18.20}$$

That is, a child's genotype is produced via either the pairwise maximum or pairwise minimum of the parental genes according to the flip of a weighted coin with probability p of selecting the pairwise maximum. The pairwise maximum *pmax* and pairwise minimum *pmin* are defined as the component wise maximum and minimum, respectively, of the two vector-valued arguments. Although it may seem redundant to produce two identical children if both coin flips match, dual propagation of the parental line may lead to a stronger evolutionary outcome over time. Furthermore, each child is subject to the mutation mechanism, so many identical twins produced at the reproduction stage may still result in genotypic differences. When the population size Z is a multiple of 4, the cth child of individuals $e[z-1]$ and $e[z]$ will replace individual $e\left[\lfloor Z/2 \rfloor + 2(z-1) + c\right]$. In this case, the reproduction mechanism ensures that each offspring replaces an individual not selected for reproduction in the previous generation to maintain the population size throughout the evolution process. For other values of Z, the population size decreases to the largest multiple of 4 less than Z after the first reproduction phase.

Mutation: Each offspring is subject to mutation immediately following birth. An offspring mutates with a user-specified mutation probability η. When an offspring mutates, select the number of mutating genes by a discrete uniform random variable on $\{1, \ldots, \lfloor \lambda kJ \rfloor\}$, with the mutation proportion parameter $\lambda \in [0,1]$ supplied by the user. (A current software implementation of this procedure suggests $\eta = 0.1$ and $\lambda = 0.25$ as the default values. However, when too many genes mutate, the algorithm devolves into random search. In seeking the proper overall rate of mutation for the problem at hand, the user must weigh the number of explanatory variables J, the interaction order constraint (18.12), the mutation probability η, and the mutation proportion parameter λ.) Each mutating gene is selected uniformly at random

and is independently assigned a new value on $(0, D+1)$ or $(0, D_0+1)$ according to a uniform random variable.

Result: After G generations of the evolutionary mechanisms, a final iteration of the selection phase is performed on the resulting population. Select the best-fit individual $(\mathbf{e}[1])$ from the final ranked population as the algorithm's result. This individual's phenotype vector $\mathbf{d}$ maps to a set of k basis functions φ_n with an associated estimate ψ_n of the parameter of interest. Because the selection mechanism is elitist, individual $\mathbf{e}[1]$ is cumulatively optimal at generation G. Within the search of size k estimators, the minimum observed risk for the specified data set is given by $\Theta(\psi_n, P)$.

In applying the EA to a particular regression setting, the user may select tuning parameters such as the population size Z, the number of generations G to run the algorithm, the mutation probability η, and the mutation proportion parameter λ specifying the maximum number of mutations that may occur in an individual. Any choice of these tuning parameters results in a new algorithm that may be compared to other estimation procedures directly in terms of risk on a test data set. Because the No Free Lunch theorem shows that no optimization algorithm solves all problems competitively [20], the EA's adaptability allows the user to generate an arbitrary number of algorithms from which to choose. Moreover, the modular structure of the proposed search algorithm allows for alternative procedures that may remove, replace, or add to the existing evolutionary mechanisms without requiring significant changes to the software's design. For instance, the user may choose to insert an additional component of mutation that modifies the mutation probability η according to the population's phenotypic homogeneity. Crossover, which splices each parent's genes into a random number of subvectors and recombines the segments to produce two complementary children, may be used as a substitute for the proposed reproduction mechanism. Although the specific combination of evolutionary mechanisms may be chosen by the user, any evolutionary algorithm requires selection pressure to generate estimators of increasing quality and some procedure (such as mutation) to identify new candidate estimators. Additionally, the user may incorporate prior information into the EA by inserting existing estimates into the initial population or limiting evolution to a subset of genes so that particular basis functions may be forced into the estimate.

As the number of evolutionary generations G increases toward infinity, the proposed EA will converge to the size k estimate with globally minimal risk. Indeed, Fogel [11] shows that an EA converges asymptotically in generation provided that all candidate optima form a single communicating class and an elitist selection mechanism is employed. In the proposed EA, any candidate optimum (as specified by an individual's phenotype) may be transformed into any other candidate optimum over the course of evolution through the mutation mechanism. Because all candidate optima form a single communicating class, the global optimum will eventually be reached over the course of evolution, and once this individual enters the population, elitist selection ensures that no other individual will ever supplant it. Therefore, the EA asymptotically converges as a function of generation to the

estimate of globally optimal risk within the size k parameter subspace. Because of this convergence, and because cross-validation is an asymptotically optimal procedure for selecting the basis size k_n as a function of the sample size n, the proposed estimator selection procedure asymptotically converges in risk to the parameter of interest ψ as n and G tend toward infinity.

When information is known about the risk surface for an estimator selection application, it may be incorporated into the design of an appropriate optimization algorithm. However, the risk surface topology is typically unknown, so we are unable to provide any general bounds on the rate at which convergence to the global optimum is achieved. Indeed, it is possible that an EA will not improve upon full enumeration in terms of the rate of asymptotic convergence; however, because an EA evolves the population according to the risk surface, it generally outperforms random search in practical settings with limited computational resources available. Similarly, it is difficult to provide *a priori* guidelines on how the EA's computational parameters should be tuned to facilitate optimization over the specific problem's unknown risk surface. However, the resulting estimator is typically most sensitive to the mutation parameters η and λ. Although allowing for a larger number of mutating genes through the choice of λ risks devolving the EA into a random search, one must take into account that not all mutations will affect the choice of basis function because of the interaction order constraint.

18.5 Simulation Studies

We conducted the following simulation experiment to test the efficacy of the proposed EA-based estimator selection procedure.

18.5.1 Simulation Study Design

Each trial consisted of generating $n = 1000$ explanatory variables and outcomes as functions (both random and non-random) of the explanatories. The trials were designed to reproduce a subset of the results obtained in Sinisi and van der Laan [16]. With $J = 11$, the explanatory variables $W = (W_1, \ldots, W_J)$ were independently generated from the uniform distribution on (0,1). Using these data, the following outcomes were created:

$$Y_1 = W_2 + W_3^2; \qquad Y_{1e} = Y_1 + \varepsilon; \qquad \varepsilon \sim N(0,1); \qquad \varepsilon \perp W; \tag{18.21}$$

$$Y_2 = W_2 W_4; \qquad Y_{2e} = Y_2 + \varepsilon; \qquad \varepsilon \sim N(0,1); \qquad \varepsilon \perp W; \tag{18.22}$$

$$Y_3 = W_2W_4W_6^2 + W_8W_{11}; \qquad Y_{3e} = Y_3 + \varepsilon; \qquad \varepsilon \sim N(0,1); \qquad \varepsilon \perp W; \tag{18.23}$$

$$Y_5 = W_1W_2^2W_3^2 + W_1W_2W_3^2W_4 + W_3^3 + W_5^4; \quad Y_{5e} = Y_5 + \varepsilon; \quad \varepsilon \sim N(0,1); \quad \varepsilon \perp W. \tag{18.24}$$

Each model a was subject to the variable degree constraint (18.14) with bounds of $D_1 = D_2 = 2$, $D_3 = 4$, and $D_5 = 5$. No interaction order constraint was imposed, so $S = J = 11$ by default. The total number of basis functions for each setting, which is given by the formula for I_0 in (18.16), is shown in Table 18.2. Figure 18.1 also shows the growth in I_0 as a function of D_0 with $J = 11$ variables and no interaction order constraint. Model 5 comprises the largest parameter space. Figure 18.2 shows how this parameter space can be pruned by introducing an interaction order constraint.

The above experiment was repeated for a total of $B = 193$ trials. Given a subset of basis functions, the parameter vector β in (18.10) was estimated using Ordinary Least Squares (OLS) linear regression. Candidate basis sizes were restricted to a maximum value of $K = 5$. V-fold cross-validation was conducted with $V = 5$ and the default values of all non-specified computational parameters. Both the EA presented in Section 18.4 and the DSA algorithm of [16] were used to estimate the parameter of interest $\psi_a = E[Y_a|W]$ for each of the above random and non-random models $a \in \{1,2,3,5\}$. However, the DSA used the polynomial degree constraint (18.13), and the EA relied upon the variable degree constraint (18.14). On each trial, the EA and DSA algorithms each performed separate random assignments of data to their respective training sets $\mathscr{T}$ and validation sets $\mathscr{V}$. Furthermore, because the DSA algorithm includes a stopping criterion based upon relative improvement in risk, the trials do not involve the same number of model fits. In general, the DSA was allowed to run until its stopping criterion was triggered, and the EA was run for the generation limits specified in Table 18.2. However, for the deterministic models, the EA was allowed to halt its search if an estimate with zero risk (with a round-off error tolerance of 10^{-15}) was located. If all V searches of a parameter subspace with basis size k located estimators that attained a validation set risk of zero, then no parameter subspaces of larger basis size were searched. Likewise, the EA also halted if the learning set search located an estimate with zero empirical risk. The current software implementation of the EA also allows for an exhaustive search of a parameter subspace if doing so is more computationally efficient than running the EA for the specified number of generations. For a population of size Z, the proposed EA fits a total of T regression estimates over G generations of evolution, where T is given by:

$$T = Z + 2G\lfloor Z/4 \rfloor. \tag{18.25}$$

The EA first fits regression estimates for each of the Z individuals in the initial population. At each generation, a total of $2\lfloor Z/4 \rfloor$ offspring are created. Because regression estimates are computationally costly, the value of T may be reduced if individuals within the population specify the same candidate solution. Similarly, when $\binom{I}{k} \leq T$ or $\binom{I_0}{k} \leq T$ for constraints (18.13) or (18.14), respectively, an exhaustive search of the size k parameter subspace is more computationally efficient

than evolution. As an additional computational parameter, the user may specify a *leeway* value l such that an exhaustive search is used provided that $\binom{I}{k} \leq T + l$ or $\binom{I_0}{k} \leq T + l$, which may be preferred when the computational limits are close to those required for an exhaustive search. In practice, an exhaustive search is often tractable when the basis size is $k = 1$ and may also be practical for size $k = 2$ when the number of variables J and the degree bound D or D_0 are of moderate size.

Table 18.2 Generation values G and variable degree bounds D_0 supplied to constraint (18.14) for the EA estimator selection procedure in the simulation studies. With $J = 11$ variables, constraint (18.14), and no interaction order constraint, the number of basis functions I_0 is calculated according to (18.16) for each estimation setting. Generation values were increased for estimation settings with a larger parameter space

Model	Cross-Validation Search Generations	Learning Set Search Generations	D_0	I_0
1	1,000	5,000	2	177,146
1*e*	2,000	5,000	2	177,146
2	1,000	5,000	2	177,146
2*e*	2,000	5,000	2	177,146
3	3,000	10,000	4	48,828,124
3*e*	5,000	12,000	4	48,828,124
5	12,000	10,000	5	362,797,055
5*e*	12,000	15,000	5	362,797,055

18.5.2 Simulation Study Results

For each simulation model studied, the EA and DSA produce estimates of the parameter of interest. We can assess these results in terms of the selected basis size, *sensitivity*, *specificity*, *cross-validated risk*, *empirical learning set risk*, and *empirical test set risk*. With respect to the true parameter ψ, an estimate ψ_n's sensitivity reflects the proportion of true basis functions included in the estimate, and its specificity denotes the proportion of the selected basis functions that are contained in the set generating the true parameter. If both quantities are one, then the estimator includes the same set of basis functions as that specifying the parameter of interest. In terms of the set of basis functions φ_n that generate an estimate ψ_n of ψ, we can define the sensitivity and specificity as follows:

$$sensitivity(\psi, \psi_n) = \frac{|\varphi \cap \varphi_n|}{|\varphi|}; \tag{18.26}$$

$$specificity(\psi, \psi_n) = \frac{|\varphi \cap \varphi_n|}{|\varphi_n|}. \tag{18.27}$$

The cross-validated risk is the minimum component of the vector (18.6) of mean validation set risks. The empirical learning set risk is the risk of the selected estimator on the learning set $\mathscr{X}_n$. The empirical test set risk is the risk of the selected estimator on a test set of 1 million observations generated independently of the learning

set from the same distribution. Because we seek to minimize risk, smaller quantities are preferred for the cross-validated, empirical learning set, and empirical test set risks. When independent and identical trials are conducted for a given algorithm on separate data sets, we can combine the results in terms of a performance metric. We are primarily concerned with the distribution of each type of risk in general and the median value in particular. The results of the simulation study are contained in Tables 18.3–18.5 and Figures 18.4–18.9. These figures contain *notched* boxplots, and evidence of a significant performance difference between the DSA and EA is noted when the notches of the respective boxplots fail to overlap [5].

The simulation's sensitivity results for the random and non-random models are summarized in Table 18.3. For non-random models, the EA consistently produces a sensitivity of 1 for estimating $E[Y_a|W]$ on Models 1, 2, and 3. Although results are variable for Model 5, the median sensitivity is 1. Meanwhile, the DSA produces strong results for some models but not for others. In the random models, the results are more varied. The EA successfully locates the proper basis functions for Model $1e$ and at least one true basis function for Models $3e$ and $5e$ but does not locate the proper term for Model $2e$. The DSA performs similarly to the EA on Models $1e, 2e$, and $5e$, but it does not locate any true basis functions for Model $3e$.

Table 18.3 Six number summaries for sensitivity measurements in the simulation study

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Model 1 EA	1.00	1.00	1.00	1.00	1.00	1.00
Model 1 DSA	1.00	1.00	1.00	1.00	1.00	1.00
Model $1e$ EA	0.00	0.50	1.00	0.81	1.00	1.00
Model $1e$ DSA	0.00	1.00	1.00	0.88	1.00	1.00
Model 2 EA	1.00	1.00	1.00	1.00	1.00	1.00
Model 2 DSA	0.00	0.00	0.00	0.00	0.00	0.00
Model $2e$ EA	0.00	0.00	1.00	0.58	1.00	1.00
Model $2e$ DSA	0.00	0.00	0.00	0.00	0.00	0.00
Model Y_3 EA	1.00	1.00	1.00	1.00	1.00	1.00
Model Y_3 DSA	0.00	0.00	0.00	0.00	0.00	0.00
Model $3e$ EA	0.00	0.00	0.00	0.23	0.50	1.00
Model $3e$ DSA	0.00	0.00	0.00	0.00	0.00	0.00
Model Y_5 EA	0.50	1.00	1.00	0.89	1.00	1.00
Model Y_5 DSA	0.50	0.50	0.50	0.50	0.50	0.50
Model $5e$ EA	0.00	0.00	0.25	0.18	0.25	0.75
Model $5e$ DSA	0.00	0.00	0.25	0.23	0.25	0.50

The specificity results are displayed in Table 18.4. The EA consistently selects only proper basis functions for Models $1, 2$, and 3, with 1 improper term and 4 correct terms typically selected for Model 5. The DSA includes both proper and improper terms for Models 1 and 5 but trails the EA in specificity on all non-random models. However, for random models, the DSA appears to perform better than the EA on Models $1e$ and $5e$, equally on Model $2e$, and worse on Model $3e$.

Table 18.4 Six number summaries for specificity measurements in the simulation study

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Model 1 EA	1.00	1.00	1.00	1.00	1.00	1.00
Model 1 DSA	0.40	0.40	0.40	0.44	0.40	1.00
Model 1*e* EA	0.00	0.25	0.40	0.36	0.40	1.00
Model 1*e* DSA	0.00	0.50	1.00	0.79	1.00	1.00
Model 2 EA	1.00	1.00	1.00	1.00	1.00	1.00
Model 2 DSA	0.00	0.00	0.00	0.00	0.00	0.00
Model 2*e* EA	0.00	0.00	0.20	0.15	0.20	1.00
Model 2*e* DSA	0.00	0.00	0.00	0.00	0.00	0.00
Model 3 EA	0.67	1.00	1.00	1.00	1.00	1.00
Model 3 DSA	0.00	0.00	0.00	0.00	0.00	0.00
Model 3*e* EA	0.00	0.00	0.00	0.11	0.20	1.00
Model 3*e* DSA	0.00	0.00	0.00	0.00	0.00	0.00
Model 5 EA	0.40	0.80	0.80	0.78	1.00	1.00
Model 5 DSA	0.40	0.40	0.40	0.40	0.40	0.50
Model 5*e* EA	0.00	0.00	0.20	0.16	0.20	0.75
Model 5*e* DSA	0.00	0.00	0.33	0.34	0.50	1.00

Table 18.5 Six number summaries for basis size error measurements in the simulation experiments

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Model 1 EA	0.00	0.00	0.00	0.00	0.00	0.00
Model 1 DSA	0.00	3.00	3.00	2.77	3.00	3.00
Model 1*e* EA	0.00	2.00	3.00	2.60	3.00	3.00
Model 1*e* DSA	0.00	0.00	0.00	0.37	1.00	3.00
Model 2 EA	0.00	0.00	0.00	0.00	0.00	0.00
Model 2 DSA	1.00	1.00	1.00	1.27	1.00	4.00
Model 2*e* EA	0.00	3.00	4.00	3.47	4.00	4.00
Model 2*e* DSA	0.00	1.00	1.00	1.26	1.00	4.00
Model 3 EA	0.00	0.00	0.00	0.01	0.00	1.00
Model 3 DSA	3.00	3.00	3.00	3.00	3.00	3.00
Model 3*e* EA	0.00	2.00	3.00	2.47	3.00	3.00
Model 3*e* DSA	−1.00	0.00	1.00	1.33	2.00	3.00
Model 5 EA	0.00	0.00	1.00	0.61	1.00	1.00
Model 5 DSA	0.00	1.00	1.00	0.99	1.00	1.00
Model 5*e* EA	−1.00	0.00	1.00	0.69	1.00	1.00
Model 5*e* DSA	−2.00	−2.00	−1.00	−0.97	0.00	1.00

Table 18.5 shows the basis size error. The error is standardized across models by subtracting the true basis size from the selected size on each trial, so an error of zero is desirable. In terms of median performance, the EA consistently selects the appropriate basis size on all non-random models but occasionally overestimates on Model 5. The DSA consistently overestimates the basis size for all non-random

models. However, for random models, the EA appears to overestimate the true basis size while the DSA either produces a smaller overestimate (Models 2*e* and 3*e*), selects the appropriate size (Model 1*e*), or underestimates the basis size (Model 5*e*).

The performance difference between the EA and DSA becomes clear when we compare the two procedures in terms of risk. Figure 18.4 (non-random models) and Figure 18.5 (random models) summarize the cross-validated risk for the estimates produced in the simulation study. Both procedures consistently locate the appropriate set of basis functions in the cross-validation stage of estimator selection on Model 1, but the EA produces a smaller median cross-validated risk for the other seven models studied. Furthermore, the EA consistently locates an estimate

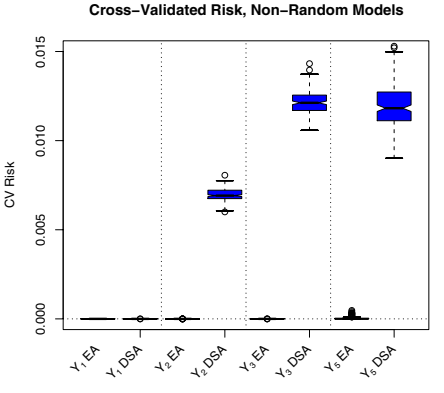


Fig. 18.4 Cross-validated risk of estimates produced by the EA and DSA algorithms for the non-random simulation models

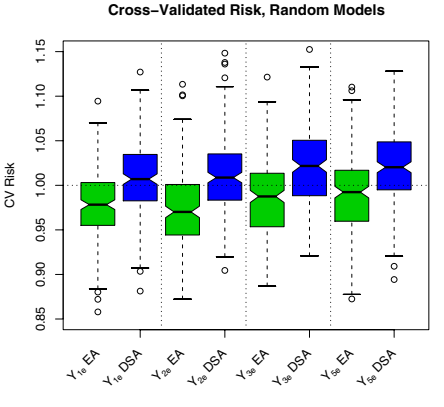


Fig. 18.5 Cross-validated risk of estimates produced by the EA and DSA algorithms for the random simulation models

resulting in essentially zero cross-validated risk (within a tolerance of 10^{-15} for rounding error) for the deterministic models.

Empirical learning set risk in the simulation study is displayed in the boxplots of Figures 18.6 and 18.7. Just as in the cross-validated risk measures, the EA consistently performs as well or better than the DSA in terms of median empirical learning set risk. Finally, Figures 18.8 and 18.9 convey an estimate of the EA and DSA's true risk obtained from a separate test data set consisting of one million observations. Again, the EA performs as well or better than the DSA on the non-random models of Figure 18.8. The DSA appears to outperform the EA on Model 1*e*, but the EA results in a superior median empirical test set risk on each of the other random models

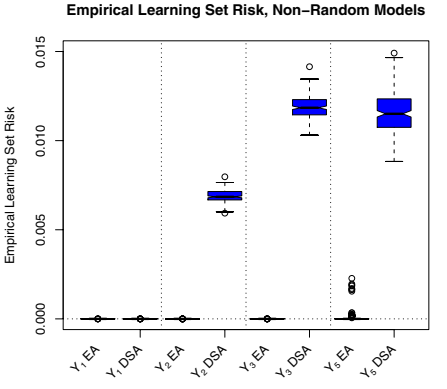


Fig. 18.6 Empirical learning set risk of estimates produced by the EA and DSA algorithms for the non-random simulation models

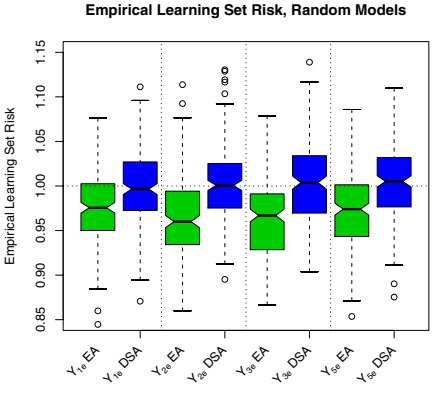


Fig. 18.7 Empirical learning set risk of estimates produced by the EA and DSA algorithms for the random simulation models

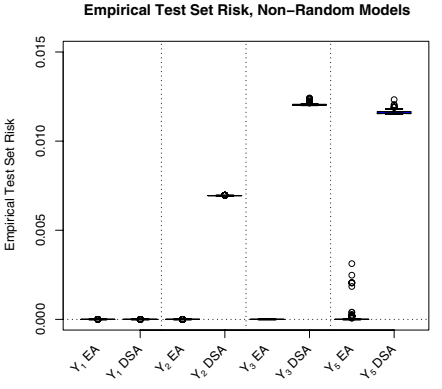


Fig. 18.8 Empirical test set risk of estimates produced by the EA and DSA algorithms for the non-random simulation models

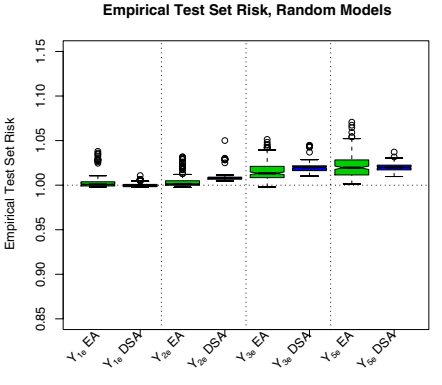


Fig. 18.9 Empirical test set risk of estimates produced by the EA and DSA algorithms for the random simulation models

of Figure 18.9. For the random models, the true risk (the risk of the true regression function) is given by the variance of the residual vector ε, which is 1 in this case because the residuals were generated from standard Normal random variables. For the simulation study, any increase above 1 in the test set risk can be attributed to a bias introduced by the selection of improper basis functions by the EA or DSA. Across all simulations, it appears that the EA's estimates produce a test set risk that exhibits greater variance than that of the DSA.

The cross-validated risk, empirical learning set risk, and empirical test set risk are all estimates of true risk for a given estimate of the regression function ψ. However, it is well known that the empirical learning set risk tends to underestimate true risk [7]. In the figures mentioned above, the median empirical learning set risk for the simulation results is smaller than the corresponding median cross-validated risk or empirical test set risk in each of the random models studied for both the EA and DSA. (In many of the non-random models, each median is zero.) In general, we prefer the empirical test set risk to the cross-validated risk in assessing an estimate's quality; because the test set data are not used in the estimator selection process, the resulting estimate cannot over-fit to the test set data. Although the median cross-validated and empirical test set risks were both close to the true risk of 1, the cross-validated risk exhibits significantly greater variability across trials than the corresponding empirical test set risk on each model. Therefore, the empirical test set risk appears to estimate true risk more reliably than the cross-validated risk in the random simulation models.

18.6 Data Analysis for a Diabetes Study

The proposed EA for estimator selection may be applied in a wide variety of estimation settings to investigate which explanatory variables contribute to an outcome of interest and examine the ways in which these variables interact. As an example of our procedure, we will study a diabetes data set used in Efron et al [10] to predict a quantitative measure of disease progression taken one year after the onset of illness. Explanatory variables include age, sex, body mass index (BMI), blood pressure (BP), and quantitative measures of six blood serum levels $S1,\ldots,S6$. Data are available for a total of $n = 442$ patients. We seek to estimate the expected value of disease progression given a particular 10-dimensional covariate profile.

In order to estimate true risk, we divided the diabetes data at random into a learning set of 392 observations and a test set of 50 data points. Using an R language implementation of the EA estimator selection procedure, we supplied the parameter values in Table 18.6 (with all other computational parameters set to the default values) to obtain an estimate of the expected disease progression given the covariate values on the learning set. Additionally, we allowed the DSA to run with the same candidate basis sizes, cross-validation folds V, and interaction order bound for constraint (18.12) as those supplied to the EA. For the DSA, the polynomial degree constraint (18.13) with $D = 3$ was used in place of the variable degree constraint (18.14) employed by the EA with $D_0 = 3$. Because the two procedures were not subject to the same constraints, the estimated regression functions are not directly comparable. It should be noted that an updated software release of the DSA (version 2.2.1) was used for this analysis compared to version 2.0.2 employed in the simulation study of Section 18.5. Computations were performed on a Unix workstation with approximately 11 gigabytes of RAM and a 2 megaHertz processor in the University of California, Berkeley's Statistical Computing Facility. The generation limits of Table 18.6 were chosen so that the computation could be performed

overnight, which was considered the maximum acceptable search time for the study. In total, this required 5.7 hours of computation.

Table 18.6 Tuning parameter values for the EA estimator selection algorithm applied to the diabetes data set of Efron et al [10]

Basis Sizes	V	D_0/D	S	Population Size, Z	CV Generations, G	Learning Set Generations, G
$\{0,1,\ldots,8\}$	5	3	3	20	5,000	10,000

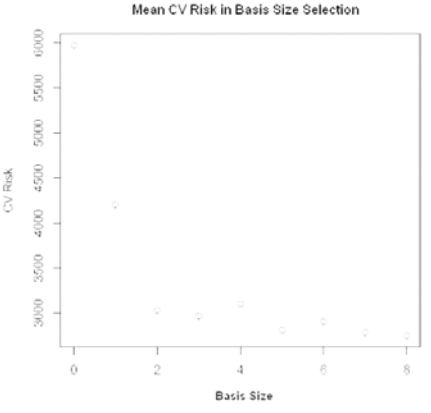


Fig. 18.10 Mean cross-validated risk of EA by candidate basis size

Figure 18.10 displays the cross-validated risk for each candidate basis size considered by the EA. During the cross-validation phase, the estimator selection algorithm selected a basis size of 8, the maximum considered. Figure 18.11 plots empirical learning set risk as a function of generation in the learning set risk optimization within the size 8 parameter subspace. Because the cumulatively optimal individual is retained at each generation, risk decreases monotonically as a function of generation. Somewhat after the 8,000th generation, the EA located an estimate that was not improved upon in the subsequent generations. The estimator selection procedure results in the OLS coefficient estimates contained in Table 18.7. Ordinarily, these coefficients are accompanied by estimated standard errors, t-statistics, and p-values for testing the null hypothesis of a zero coefficient. However, such inferences can only be drawn through a model of the underlying distribution of the estimator, which is currently an open problem for estimator selection procedures such as those considered in this paper. Similarly, Table 18.8 shows the regression coefficient estimates obtained by the DSA. The basis function including the $S5$ serum measurement was selected by both the EA and DSA, but otherwise the selected basis functions differed in terms of degree, order of interaction, and coefficient estimates. Most of the basis functions selected by the EA contain higher powers, a maximal

order of interaction, and generally large coefficient estimates. In contrast, the DSA produced an estimate with no higher powers assigned to any variable, relatively few variable interactions, and smaller coefficient estimates that produce a simpler interpretation for the effect of each variable. It is possible that the EA would also produce a more meaningful estimate if the polynomial degree constraint (18.13) were used in place of the variable degree constraint (18.14), which would limit the parameter space to a subspace of that considered here. However, at the time of this analysis, the software implementation of the EA for the polynomial degree constraint (18.13) was not yet available.

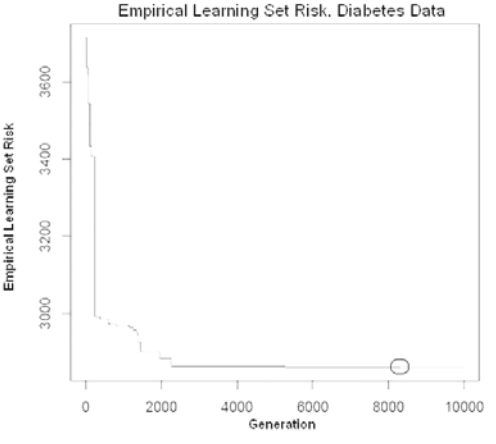


Fig. 18.11 Empirical learning set risk as a function of generation in risk optimization on estimates of size 8. The circled region contains the generation at which the final estimate was located by the EA

Table 18.7 EA regression coefficient estimates for disease progression in the diabetes study

Int.	$S5$	$Sex^3{:}BMI^3{:}S1^3$	$Sex^2{:}BMI^3{:}S4^2$	$Age^3{:}S3^3{:}S6^3$	$Age{:}S6$	$Sex^2{:}BMI$	$BP^2{:}S4^3{:}S6^2$	$BMI^2{:}S4^3{:}S6^2$
-3.67	5.95e02	-1.48e11	3.04e08	2.03e11	3.72e03	3.04e05	7.28e08	-3.76e08

Table 18.8 DSA regression coefficient estimates of disease progression in the diabetes study

Int.	*BMI*	*S5*	*S3*	*BP*	*SEX*	*BMI* : *BP*	*AGE* : *SEX*
−5.84	525.98	549.76	315.14	295.34	−255.71	3910.98	3913.89

We then compared the EA and DSA estimates to those obtained by a variety of other estimator selection procedures considered by Durbin et al [9]. These estimates'

test set risks were calculated on a total of $B = 100$ bootstrap samples produced from the test set data. Although the learning and test sets were identical to those used by Durbin et al [9], the specific bootstrap test set samples previously used were not available. However, the bootstrap test sets generated in this analysis are i.i.d. observations produced from the sampling technique of Durbin et al [9]. The results are displayed in Table 18.9. In terms of mean test set risk, both the EA and the DSA improved upon the performance of all estimators considered by Durbin et al [9]. In particular, the EA's estimate resulted in a mean test set risk that improved upon all previous results by approximately 7.9%. Moreover, the DSA's estimate improved upon that of the EA by approximately 10.2%. Figure 18.12 displays a notched boxplot of bootstrap test set risk for each estimator selection algorithm. These results may be directly compared to those contained in [9], which are reproduced in Figure 18.13 with the permission of the authors. Because its notches do not overlap with those of any other estimator, it appears that the DSA significantly outperforms all other estimators considered for this particular problem. The EA and DSA's 95% confidence intervals for test set risk appear to be wider than those of the other estimators studied. It is possible that the proposed EA produces a greater variability in its estimates on account of its stochastic mechanisms in the reproduction and mutation stages.

Table 18.9 Empirical test set risk of several estimator selection procedures on the diabetes data of Efron et al [10] based upon $B = 100$ bootstrap samples of the diabetes data test set of 50 observations. The EA and DSA results are compared in terms of risk to a number of estimators tested in Durbin et al [9] on the diabetes data . The table shows the mean bootstrap test set risk and 95% risk confidence interval for each estimator based on 100 bootstrap samples from the test set. Confidence intervals were produced from normal theory according to estimates of the mean and standard deviation for each estimator's risk. The third column compares each procedure's mean risk ratio to that of the EA, and the final column shows which covariates were included in each algorithm's selected estimator. It appears that all results obtained by Durbin et al [9] were at least 7.9% larger in test set risk than that obtained from the EA, and the DSA subsequently improved on the EA by approximately 10.2%

Estimator	Bootstrap Test Set Risk	Ratio	Covariates
lm	3182.0 (1966.2, 4397.7)	1.079	(all)
LARS (CV)	3270.9 (2118.6, 4423.2)	1.109	*Sex*, *BMI*, *BP*, $S1 - S3$, *S5*, *S6*
polymars	3301.8 (2123.8, 4479.9)	1.119	*Sex*, *BMI*, *BP*, *S3*, *S5*, *S6*
LARS (Cp)	3336.7 (2206.2, 4467.3)	1.131	*Sex*, *BMI*, *BP*, *S2*, *S3*, *S5*, *S6*
full nnet	3552.4 (2297.2, 4807.6)	1.204	(all)
nnet-DSA	3565.2 (2368.4, 4175.8)	1.208	(all)
rpart	3692.0 (2498.9, 4885.0)	1.251	*BMI*, *BP*, *S2*, *S3*, *S5*, *S6*
DSA	**2649.3 (1337.1, 3961.6)**	0.898	*Sex*, *BMI*, *BP*, *S3*, *S5*
EA	2950.3 (1670.6, 4230.0)	1	*Age*, *Sex*, *BMI*, *BP*, $S1, S3 - S6$

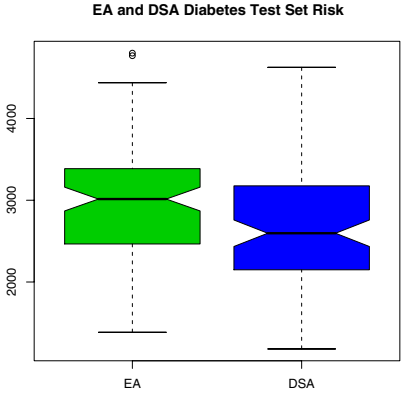


Fig. 18.12 Boxplots of bootstrap test set risk of the EA and DSA estimates obtained from the diabetes data based upon $B = 100$ bootstrap samples of the test set. These results may be directly compared to those obtained by Durbin et al [9] in Figure 18.13

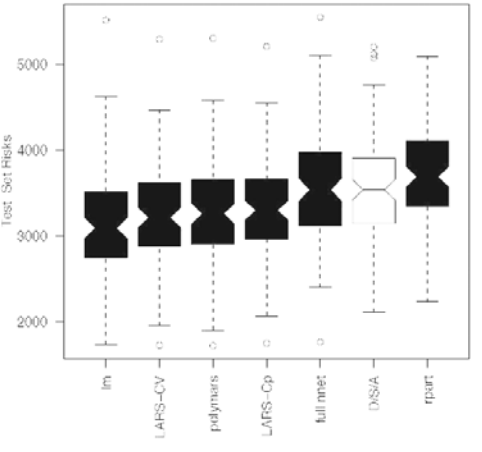


Fig. 18.13 Empirical test set risk of several estimator selection procedures on the diabetes data based upon $B = 100$ bootstrap samples of the diabetes data test set of 50 observations. This figure was originally produced by Durbin et al [9] and is reproduced here with the permission of the authors. These results may be directly compared to those of the EA and DSA, which are displayed in Figure 18.12

18.7 Conclusion

In light of the size of parameter spaces for the constraint profiles characterized in Section 18.3, estimator selection procedures operating according to the general road map for loss based estimation must be able to search quickly and effectively for candidate estimators minimizing empirical risk within parameter subspaces. EAs and similar stochastic optimization algorithms provide an aggressive approach to risk optimization and are sufficiently flexible to offer high-quality estimates in a wide variety of settings. The results of the simulation study and diabetes analysis establish the proposed EA as a competitive alternative to other procedures. Because the No Free Lunch Theorem [20] shows that no single algorithm can always outperform all others, the proposed EA may be used as a complement to the DSA as a general tool for estimator selection in regression settings. The EA is an attractive alternative because its computational parameters can be adapted to the problem at hand, and its modular design allows for variations of its evolutionary mechanisms without requiring significant changes in the overall software implementation. Furthermore, the EA converges asymptotically in generation to the global optimum within the size k parameter subspace to be searched. It should be noted that asymptotic convergence does not ensure that a global optimum will be reached in the allotted time. However, the EA performed competitively in the simulations and data analysis, and its asymptotic convergence property and elitist selection mechanism indicates that further computations would only improve the quality of its results.

While the DSA search algorithm shifts between parameter subspaces of different basis size, the EA independently searches each subspace. This separation allows for parallel computing techniques to simultaneously search different parameter subspaces on additional processors and also allows the user to tune computational parameters like the population size, mutation probability, and number of generations according to the size of the subspace. Although the EA described is designed to search a parameter space consisting of polynomial regression functions, the proposed methodology applies to general parameterizations (e.g. histogram regression and neural networks), which is an appealing feature of both the EA and the DSA.

The results obtained in this study come with a few caveats: first, in general the EA required significantly more time to produce its estimates than the DSA. This time difference may be attributed to the DSA's implementation in the C programming language, which is significantly faster than R. Although this project illustrates the EA's utility, it also demonstrates the need to improve the algorithm's speed in subsequent software packages. Future implementations of the EA may also apply parallel computing techniques to simultaneously search distinct subspaces or training sets in the cross-validation phase. However, because statistical estimation typically occurs at the end of a lengthy study, these computations are not especially time–sensitive, and in many cases it is reasonable to allow several hours or days for this task.

Because the DSA was treated as a black box in the simulations, a comparison to the EA in terms of the number of model fits required to obtain an estimate of a given quality is currently unavailable. However, the simulation results suggest that the DSA is vulnerable to local optima. Unlike EAs, the DSA risk-optimizing search

procedure is deterministic for a given split of the data into training and validation sets. Future versions of the DSA may consider introducing a stochastic component akin to the EA's mutation mechanism to work in concert with its existing elitist selection procedure. If the proposed augmentation ensures that all estimators within a parameter subspace form a single communicating class, then this modified DSA would asymptotically converge in time to the global optimum.

Additionally, estimator selection software packages may provide the researcher with the opportunity to include particular basis functions in all candidate estimates so that known causal relationships remain fixed while searching for additional factors that contribute to a quantity of interest. When the researcher wishes to compare results from a large number of distinct algorithms, an arbitrary number of alternative search procedures may be generated by varying the EA's tuning parameters such as the mutation probability. For a particular problem, an additional cross-validation procedure may be used to select among candidate mutation probabilities or other tuning parameters. Finally, the variability of the EA's results may be investigated as a function of generation to guide the choice of these computational parameters.

18.8 Appendix

Section 18.3.3 analyzed the size of the parameter space for polynomial regression under the interaction order constraint (18.12) and the polynomial degree constraint (18.13) or the variable degree constraint (18.14). We wish to substantiate the conclusions summarized in Table 18.1.

Under constraint (18.13), the number of basis functions is given by the value of I (18.15), which can be bounded below as follows:

$$I = \sum_{s=1}^{S} \binom{J}{s} \left[1 + \sum_{d=s+1}^{D} \sum_{k=1}^{\min(s,d-s)} \binom{s}{k} \binom{d-s-1}{k-1} \right] \geq \sum_{s=1}^{S} \binom{J}{s} \geq \sum_{s=1}^{S} \binom{S}{s}$$
$$= 2^S - 1 \Rightarrow \Omega\left(2^S\right). \tag{18.28}$$

The first equality restates (18.15), and the first inequality follows because all terms in the nested summations are positive. Under constraint (18.13), then $S \leq \min(J,D) \leq J$, and $\binom{S}{s} \leq \binom{J}{s}$ for all $s \in \{1,\ldots,S\}$, so the second inequality holds. The final equality is a direct consequence of the Binomial Theorem. Therefore, I is bounded below by a function of order $\Omega\left(2^S\right)$. For the extreme case of $S = \min(J,D)$, then $I = \Omega\left(2^S\right) = \Omega\left(2^{\min(J,D)}\right)$, which is an exponential function of the number of variables J and the polynomial degree bound D. We then turn our attention to the case of I_0 under the variable degree constraint (18.14). We can bound I_0 from below as follows:

$$I_0 = \sum_{s=1}^{S} \binom{J}{s} D_0^s \geq \sum_{s=1}^{S} \binom{S}{s} D_0^s = (D_0+1)^S - 1 > D_0^S \Rightarrow \Omega\left(D_0^S\right). \tag{18.29}$$

The first equality restates (18.16), and the first inequality follows because $J \geq S$ when constraint (18.14) is imposed. The next equality follows from the Binomial Theorem, and the remaining polynomial is of degree S. In the extreme case of $S = J$, the number of basis functions is then $\Omega\left(D_0^J\right)$. Because the summation in (18.29) is solved in a closed form and results in a polynomial when $S = J$, this asymptotic lower bound is also an asymptotic upper bound, and both are tight [6]. Therefore the number of basis functions is both $\Omega\left(D_0^J\right)$ and $O\left(D_0^J\right)$ when $S = J$. Because S is maximized, this upper bound is an overall upper bound on the number of basis functions under the variable degree constraint (18.14). Furthermore, because the set of basis functions under the polynomial degree constraint (18.13) is a subset of those under the variable degree constraint (18.14) when $D = D_0$, then the number of basis functions I is trivially bounded above by a function of order $O\left(D^J\right)$. Likewise, the value I_0 for constraint (18.14) is loosely bounded above by a function of order $O\left(D_0^J\right)$ that becomes tight if $S = J$.

The size of the parameter space is 2^I or 2^{I_0} in the constraint profiles of Section 18.3. By applying the previous bounds for I and I_0 to the parameter space analysis, we arrive at the conclusions summarized in Table 18.1. It should be noted that the order functions Ω and O imply that the bounds can be stated as a constant times the given function. In expressing the size of the parameter space in terms of the number of basis functions under different constraint profiles, the constant for the order of the size of the parameter space differs from that for the order of the number of basis functions.

Acknowledgments

The authors wish to thank Ron Peled, Cathy Tuglus, Burke Bundy, Mark van der Laan, and the anonymous reviewers for their helpful suggestions. Blythe Durbin provided information about the design of her previous diabetes analysis to facilitate a fair comparison of the proposed method to other predictors. Richard Liang gratefully acknowledges the support of the Natural Sciences and Engineering Research Council (NSERC) of Canada.

References

1. Bäck, T.: Evolutionary Algorithms in Theory and Practice. Oxford University Press, Oxford (1996)
2. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
3. Breiman, L., Friedman, J.H., Stone, C.J., Olshen, R.A.: Classification and Regression Trees. Chapman and Hall, Boca Raton (1984)
4. Candes, E., Tao, T.: The dantzig selector: Statistical estimation when p is much larger than n. Annals of Statistics 35(6), 2313–2351 (2007)
5. Chambers, J.M., Cleveland, W.S., Tukey, P.A.: Graphical Methods for Data Analysis. Duxbury Press (1983)
6. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. MIT Press, Cambridge (1990)

7. Dudoit, S., van der Laan, M.J.: Asymptotics of cross-validated risk estimation in estimator selection and performance assessment. Statistical Methodology 2(2), 131–154 (2005)
8. Dudoit, S., van der Laan, M.J., Keleş, S., Molinaro, A.M., Sinisi, S.E., Teng, S.L.: Loss-based estimation with cross-validation: Applications to microarray data analysis. In: Piatetsky-Shapiro, G., Tamayo, P. (eds.) Microarray Data Mining. SIGKDD Explorations, vol. 5, pp. 56–68. ACM, New York (2003), http://www.acm.org/sigs/sigkdd/explorations/issue5-2.htm
9. Durbin, B., Dudoit, S., van der Laan, M.J.: Optimization of the architecture of neural networks using a Deletion/Substitution/Addition algorithm. Tech. Rep. 170, Division of Biostatistics, University of California, Berkeley (2005), www.bepress.com/ucbbiostat/paper170
10. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression. Annals of Statistics 32(4), 407–499 (2004)
11. Fogel, D.B.: Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, Los Alamitos (2005)
12. Freedman, D.A.: Statistical Models: Theory and Practice, 2nd edn. Cambridge University Press, Cambridge (2009)
13. Friedman, J.H.: Multivariate adaptive regression splines. The Annals of Statistics 19(1), 1–141 (1991)
14. Friedman, J.H.: Fast sparse regression and classification. Tech. rep., Department of Statistics, Stanford University (2008), http://www-stat.stanford.edu/~jfh/
15. van der Laan, M.J., Dudoit, S.: Unified cross-validation methodology for selection among estimators and a general cross-validated adaptive ε-net estimator: Finite sample oracle inequalities and examples. Tech. Rep. 130, Division of Biostatistics, University of California, Berkeley (2003), www.bepress.com/ucbbiostat/paper130
16. Sinisi, S.E., van der Laan, M.J.: Deletion/substitution/addition algorithm in learning with applications in genomics. Statistical Applications in Genetics and Molecular Biology 3(1), Article 18 (2004), www.bepress.com/sagmb/vol3/iss1/art18
17. Specht, D.F.: A general regression neural network. IEEE Transactions on Neural Networks 2(6), 568–576 (1991)
18. Stoll, M.: Introduction to Real Analysis. Addison-Wesley, Reading (2000)
19. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society 58(1), 267–288 (1996)
20. Wolpert, D.H., MacReady, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)

Part III
Real-World Applications

Chapter 19
Particle Swarm Optimisation Aided MIMO Transceiver Designs

S. Chen, W. Yao, H.R. Palally, and L. Hanzo

Abstract. Multiple-input multiple-output (MIMO) technologies are capable of substantially improving the achievable system's capacity, coverage and/or quality of service. The system's ability to approach the MIMO capacity depends heavily on the designs of MIMO receiver and/or transmitter, which are generally expensive optimisation tasks. Hence, researchers and engineers have endeavoured to develop efficient optimisation techniques that can solve practical MIMO designs with affordable costs. In this contribution, we demonstrate that particle swarm optimisation (PSO) offers an efficient means for aiding MIMO transceiver designs. Specifically, we consider PSO-aided semi-blind joint maximum likelihood channel estimation and data detection for MIMO receiver, and we investigate PSO-based minimum bit-error-rate multiuser transmission for MIMO systems. In both these two MIMO applications, the PSO-aided approach attains an optimal design solution with a significantly lower complexity than the existing state-of-the-art scheme.

19.1 Introduction

Multiple-input multiple-output (MIMO) technologies are widely adopted in practice to improve the system's achievable capacity, coverage and/or quality of service [14, 15, 30, 32, 33, 41, 42, 43, 45]. The designs of MIMO receiver and/or transmitter critically influence the system's ability to approach the MIMO capacity. MIMO transceiver designs, which are typically expensive optimisation tasks, have motivated researchers and engineers to develop efficient optimisation techniques that can attain optimal MIMO designs with affordable costs. Hence, the particle swarm optimisation (PSO) as an advanced optimisation tool can offer an efficient means for aiding MIMO transceiver designs. PSO [25] is a population based stochastic optimisation technique inspired by social behaviour of bird flocking or fish schooling. The algorithm commences with random initialisation of a swarm of individuals, referred to as particles, within the problem's search space. It then endeavours to find

S. Chen · W. Yao · H.R. Palally · L. Hanzo
School of Electronics and Computer Science,
University of Southampton, Southampton SO17 1BJ, UK
e-mail: {sqc,wy07r,hrp1v07,lh}@ecs.soton.ac.uk

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 487–511.
springerlink.com

a global optimal solution by gradually adjusting the trajectory of each particle toward its own best location and toward the best position of the entire swarm at each evolutionary optimisation step. The PSO method is popular owing to its simplicity in implementation, ability to rapidly converge to a "reasonably good" solution and its robustness against local minima. The PSO method has been successfully applied to wide-ranging optimisation problems [10, 12, 13, 16, 18, 26, 27, 35, 37, 38]. In particular, many research works have applied the PSO techniques to multiuser detection (MUD) [11, 17, 28, 29, 36]. In this contribution we consider the PSO aid MIMO transceiver designs. Specifically, we develop the PSO aided semi-blind joint maximum likelihood (ML) channel estimation and data detection for MIMO receivers and we investigate the PSO-based minimum bit error rate (MBER) multiuser transmission (MUT) for MIMO systems.

In a MIMO receiver, if the channel state information (CSI) is available, optimal ML data detection can be performed using for example the optimised hierarchy reduced search algorithm (OHRSA) aided detector [2], which is an advanced extension of the complex sphere decoder [34]. Accurately estimating a MIMO channel however is a challenging task, and a high proportion of training symbols is required to obtain a reliable least square channel estimate (LSCE) which considerably reduces the achievable system throughput. Although blind joint ML channel estimation and data detection does not reduce the achievable system throughput, it suffers from drawbacks of excessively high computational complexity and an inherent estimation and decision ambiguities [40]. An interesting scheme for semi-blind joint ML channel estimation and data detection has been proposed in [1], in which the joint ML channel estimation and data detection optimisation is decomposed into two levels. At the upper level a population-based optimisation algorithm known as the repeated weighted boosting search (RWBS) algorithm [7] searches for an optimal channel estimate, while at the lower level the OHRSA detector [2] recovers the transmitted data. Joint ML channel estimation and data detection is achieved by iteratively exchanging information between the RWBS-aided channel estimator and the OHRSA data detector. The scheme is semi-blind as it employs a few training symbols, approximately equal to the rank of the MIMO system, to provide an initial LSCE for aiding the RWBS channel estimator to improve its convergence. The employment of a minimum training overhead has an additional benefit in terms of avoiding the ambiguities inherent in pure blind joint channel estimation and data detection. This study advocates the PSO aided alternative for semi-blind joint ML channel estimation and data detection. We will demonstrate that this PSO aided scheme compares favourably with the existing state-of-the-art RWBS based method, in terms of performance and complexity.

In the downlink of a space-division multiple-access (SDMA) induced MIMO system, mobile terminal (MT) receivers are incapable of cooperatively performing sophisticated MUD. In order to facilitate the employment of a low-complexity high-power efficiency single-user-receiver, the transmitted signals have to be preprocessed at the base station (BS), leading to the appealing concept of multiuser transmission (MUT) [50], provided that accurate downlink CSI is available at the transmitter. The assumption that the downlink channel impulse response (CIR) is

known at the BS may be deemed valid in time division duplex (TDD) systems, where the uplink and downlink signals are transmitted at the same frequency, provided that the co-channel interference is also similar at the BS and the MTs. MUT-aided transmit preprocessing may hence be deemed attractive, when the channel's coherence time is longer than the transmission burst interval. However, for frequency division duplex (FDD) systems, where the uplink and downlink channels are expected to be different, CIR feedback from the MT's receivers to the BS transmitter is necessary [51]. Most of the MUT techniques are designed based on the minimum mean-square-error (MMSE) criterion [44, 51]. Since the achievable bit error rate (BER) is the ultimate system performance indicator, interests on minimum BER (MBER) based MUT techniques have increased recently [21, 39]. The optimal MBER-MUT design is a constrained nonlinear optimisation [21, 39], and the sequential quadratic programming (SQP) algorithm [31] is typically used to obtain the precoder's coefficients for the MBER-MUT [21, 23, 39]. In practice, the computational complexity of the SQP based MBER-MUT solution can be excessive for high-rate systems [23] and, therefore, it is difficult for practical implementation. In this contribution, the PSO algorithm is invoked to find the precoder's coefficients for the MBER-MUT in order to reduce the computational complexity to a practically acceptable level. Our results obtained in [52] have demonstrated that the PSO aided MBER-MUT design imposes a much lower computational complexity than the existing SQP-based MBER-MUT design.

The rest of this contribution is structured as follows. In Section 19.2, the PSO algorithm is presented. Section 19.3 is devoted to the development of the PSO-aided semi-blind joint ML scheme, while Section 19.4 derives the PSO assisted optimal MBER-MUT scheme. Our conclusions are then offered in Section 19.5.

Throughout our discussions we adopt the following notational conventions. Boldface capitals and lower-case letters stand for complex-valued matrices and vectors of appropriate dimensions, respectively, while $\mathbf{I}_K$ and $\mathbf{1}_{K\times L}$ denote the $K\times K$ identity matrix and the $K\times L$ matrix of unity elements, respectively. The (p,q)th element $h_{p,q}$ of $\mathbf{H}$ is also denoted by $\mathbf{H}|_{p,q}$. Furthermore, $(\,)^T$ and $(\,)^H$ represent the transpose and Hermitian operators, respectively, while $\|\,\|^2$ and $|\,|$ denote the norm and the magnitude operators, respectively. $E[\,]$ denotes the expectation operator, while $\Re[\,]$ and $\Im[\,]$ represent the real and imaginary parts, respectively. Finally, $j=\sqrt{-1}$.

19.2 Particle Swarm Optimisation

Consider the generic optimisation task defined as follows

$$\mathbf{U}_{\text{opt}} = \arg\min_{\mathbf{U}} F(\mathbf{U}) \tag{19.1}$$

$$\text{s.t. } \mathbf{U}\in\mathsf{U}^{N\times M} \tag{19.2}$$

where $F(\,)$ is the cost function of the optimisation problem, $\mathbf{U}$ is a $N\times M$ complex-valued parameter matrix to be optimised, and

$$\mathsf{U} = \left[-U_{\max},\, U_{\max}\right] + j\left[-U_{\max},\, U_{\max}\right] \tag{19.3}$$

defines the search range for each element of $\mathbf{U}$. The flowchart of the PSO algorithm is given in Fig. 19.1. A swarm of particles, $\{\mathbf{U}_i^{(l)}\}_{i=1}^{S}$, that represent potential solutions are evolved in the search space $\mathsf{U}^{N\times M}$, where S is the swarm size and index l denotes the iteration step. The details of the algorithm is now explained.

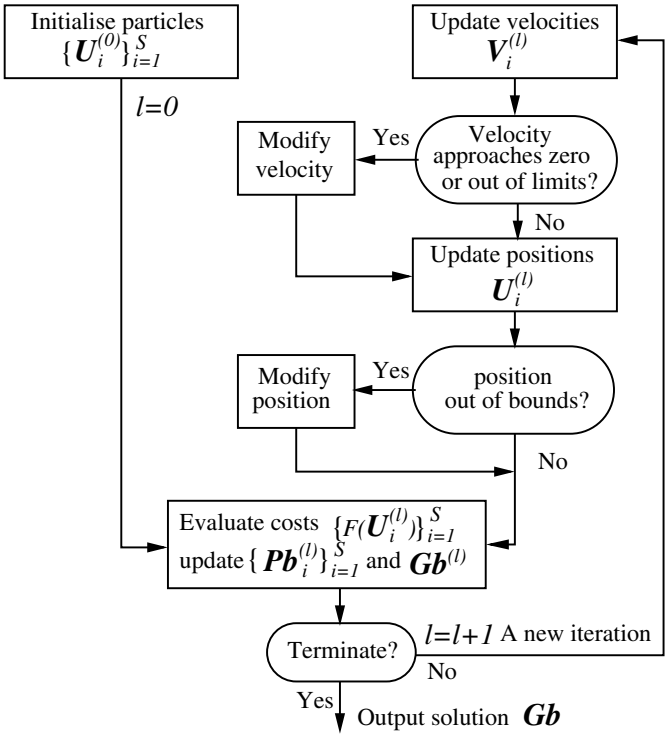


Fig. 19.1 Flowchart of the PSO algorithm

19.2.1 PSO Algorithm

a) The swarm initialisation. Set $l = 0$ and generate the initial particles, $\{\mathbf{U}_i^{(l)}\}_{i=1}^{S}$, in the search space $\mathsf{U}^{N\times M}$ with a prescribed way. Typically, the initial particles are randomly generated.

b) The swarm evaluation. For each particle $\mathbf{U}_i^{(l)}$, compute its associated cost $F\left(\mathbf{U}_i^{(l)}\right)$. Each particle $\mathbf{U}_i^{(l)}$ remembers its best position visited so far, denoted as $\mathbf{Pb}_i^{(l)}$, which provides the cognitive information. Every particle also knows the best position visited so far among the entire swarm, denoted as $\mathbf{Gb}^{(l)}$, which provides the social

information. The cognitive information $\{\mathbf{Pb}_i^{(l)}\}_{i=1}^{S}$ and the social information $\mathbf{Gb}^{(l)}$ are updated at each iteration:

For $(i = 1; i \leq S; i{+}{+})$
  If $(F(\mathbf{U}_i^{(l)}) < F(\mathbf{Pb}_i^{(l)}))$ $\mathbf{Pb}_i^{(l)} = \mathbf{U}_i^{(l)}$;
End for;
$i^* = \arg\min_{1\leq i\leq S} F(\mathbf{Pb}_i^{(l)})$;
If $(F(\mathbf{Pb}_{i^*}^{(l)}) < F(\mathbf{Gb}^{(l)}))$ $\mathbf{Gb}^{(l)} = \mathbf{Pb}_{i^*}^{(l)}$;

c) The swarm update. Each particle $\mathbf{U}_i^{(l)}$ has a velocity, denoted as $\mathbf{V}_i^{(l)}$, to direct its "flying" or search within the search space. The velocity and position of the ith particle are updated in each iteration according to:

$$\mathbf{V}_i^{(l+1)} = \xi * \mathbf{V}_i^{(l)} + c_1 * \varphi_1 * (\mathbf{Pb}_i^{(l)} - \mathbf{U}_i^{(l)}) + c_2 * \varphi_2 * (\mathbf{Gb}^{(l)} - \mathbf{U}_i^{(l)}), \quad (19.4)$$

$$\mathbf{U}_i^{(l+1)} = \mathbf{U}_i^{(l)} + \mathbf{V}_i^{(l+1)}, \quad (19.5)$$

where ξ is the inertia weight, c_1 and c_2 are the two empirically chosen acceleration coefficients, while $\varphi_1 = rand()$ and $\varphi_2 = rand()$ denotes the two random variables uniformly distributed in $(0,\ 1)$.

In order to avoid excessive roaming of particles beyond the search space [18], a velocity space $\mathsf{V}^{N\times M}$ with

$$\mathsf{V} = \left[-V_{\max},\ V_{\max}\right] + j\left[-V_{\max},\ V_{\max}\right] \quad (19.6)$$

is imposed so that each element of $\mathbf{V}_i^{(l+1)}$ is within the search range V defined in (19.6), namely,

If $(\Re[\mathbf{V}_i^{(l+1)}|_{p,q}] > V_{\max})$ $\Re[\mathbf{V}_i^{(l+1)}|_{p,q}] = V_{\max}$;
If $(\Re[\mathbf{V}_i^{(l+1)}|_{p,q}] < -V_{\max})$ $\Re[\mathbf{V}_i^{(l+1)}|_{p,q}] = -V_{\max}$;
If $(\Im[\mathbf{V}_i^{(l+1)}|_{p,q}] > V_{\max})$ $\Im[\mathbf{V}_i^{(l+1)}|_{p,q}] = V_{\max}$;
If $(\Im[\mathbf{V}_i^{(l+1)}|_{p,q}] < -V_{\max})$ $\Im[\mathbf{V}_i^{(l+1)}|_{p,q}] = -V_{\max}$;

Moreover, if $\mathbf{V}_i^{(l+1)}$ approaches zero, it is reinitialised proportional to $V_{\max}$ with a small control factor γ according to:

If $(\Re[\mathbf{V}_i^{(l+1)}|_{p,q}] == 0)$
  If$(rand() < 0.5)$
    $\Re[\mathbf{V}_i^{(l+1)}|_{p,q}] = \varphi_v * \gamma * V_{\max}$;
  Else
    $\Re[\mathbf{V}_i^{(l+1)}|_{p,q}] = -\varphi_v * \gamma * V_{\max})$;
  End if;
Else if $(\Im[\mathbf{V}_i^{(l+1)}|_{p,q}] == 0)$
  If$(rand() < 0.5)$
    $\Im[\mathbf{V}_i^{(l+1)}|_{p,q}] = \varphi_v * \gamma * V_{\max}$;

Else
$\Im[\mathbf{V}_i^{(l+1)}|_{p,q}] = -\varphi_v * \gamma * V_{\max})$;
End if;
End if;

where $\varphi_v = rand()$ is another uniform random variable in $(0,\ 1)$.

Similarly, each $\mathbf{U}_i^{(l+1)}$ is checked to ensure that it stays inside the search space $\mathsf{U}^{N\times M}$. This can be done for example with the rule:

If $(\Re[\mathbf{U}_i^{(l+1)}|_{p,q}] > U_{\max})$ $\Re[\mathbf{U}_i^{(l+1)}|_{p,q}] = U_{\max}$;
If $(\Re[\mathbf{U}_i^{(l+1)}|_{p,q}] < -U_{\max})$ $\Re[\mathbf{U}_i^{(l+1)}|_{p,q}] = -U_{\max}$;
If $(\Im[\mathbf{U}_i^{(l+1)}|_{p,q}] > U_{\max})$ $\Im[\mathbf{U}_i^{(l+1)}|_{p,q}] = U_{\max}$;
If $(\Im[\mathbf{U}_i^{(l+1)}|_{p,q}] < -U_{\max})$ $\Im[\mathbf{U}_i^{(l+1)}|_{p,q}] = -U_{\max}$;

An alternative rule is, if a particle is outside the search space, it is moved back inside the search space randomly, rather than forcing it to stay at the border as the previous rule does. That is,

If $(\Re[\mathbf{U}_i^{(l+1)}|_{p,q}] > U_{\max})$ $\Re[\mathbf{U}_i^{(l+1)}|_{p,q}] = rand() * U_{\max}$;
If $(\Re[\mathbf{U}_i^{(l+1)}|_{p,q}] < -U_{\max})$ $\Re[\mathbf{U}_i^{(l+1)}|_{p,q}] = -rand() * U_{\max}$;
If $(\Im[\mathbf{U}_i^{(l+1)}|_{p,q}] > U_{\max})$ $\Im[\mathbf{U}_i^{(l+1)}|_{p,q}] = rand() * U_{\max}$;
If $(\Im[\mathbf{U}_i^{(l+1)}|_{p,q}] < -U_{\max})$ $\Im[\mathbf{U}_i^{(l+1)}|_{p,q}] = -rand() * U_{\max}$;

This is similar to the checking procedure given in [18].

d) Termination condition check. If the maximum number of iterations, $I_{\max}$, is reached, terminate the algorithm with the solution $\mathbf{U}_{\text{opt}} = \mathbf{Gb}^{(I_{\max})}$; otherwise, set $l = l + 1$ and go to Step *b)*.

19.2.2 Complexity of PSO Algorithm

Let the computational complexity of one cost function evaluation be C_{single}. Given the swarm size S, assume that the algorithm converges in $I_{\max}$ iterations. Then the total number of cost function evaluations is simply $N_{\text{total}} = S \times I_{\max}$, and the complexity of the algorithm is given by

$$C = N_{\text{total}} \times C_{\text{single}} = S \times I_{\max} \times C_{\text{single}}. \tag{19.7}$$

19.2.3 Choice of PSO Algorithmic Parameters

We now comment on the choices of PSO algorithmic parameters. The search bound $U_{\max}$ is specified by the optimisation problem considered, while the velocity limit $V_{\max}$ is typically related to the value of $U_{\max}$. The swarm size S depends on how hard the optimisation problem (19.1) is. For small to medium size optimisation problems, a standard choice recommended in the literature is $S = 20$ to 50. The maximum

number of iterations, $I_{\max}$, is generally determined by experiment. In our experiments we choose the optimal swarm size S to minimise the total complexity C of (19.7).

It was reported in [35] that a time varying acceleration coefficient (TVAC) enhances the performance of PSO. In this TVAC mechanism [35], c_1 for the cognitive component is reduced from 2.5 to 0.5 and c_2 for the social component varies from 0.5 to 2.5 respectively during the iterative procedure according to

$$\left.\begin{aligned} c_1 &= (0.5-2.5)*l/I_{\max}+2.5 \\ c_2 &= (2.5-0.5)*l/I_{\max}+0.5 \end{aligned}\right\} \tag{19.8}$$

The reason given for this TVAC mechanism is that at the initial stages, a large cognitive component and a small social component help particles to wander around or exploit better the search space and to avoid local minima. In the later stages, a small cognitive component and a large social component help particles to converge quickly to a global minimum.

We also experiment an alternative TVAC mechanism in which c_1 is varies from 0.5 to 2.5 and c_2 changes from 2.5 to 0.5 during the iterative procedure according to

$$\left.\begin{aligned} c_1 &= (2.5-0.5)*l/I_{\max}+0.5 \\ c_2 &= (0.5-2.5)*l/I_{\max}+2.5 \end{aligned}\right\} \tag{19.9}$$

Which TVAC mechanism to choose is decided by empirical performance in our applications.

Several choices of the inertia weight can be considered, including the zero inertia weight $\xi = 0$, a constant inertia weight ξ or a random inertia weight $\xi = rand()$. In our applications, empirical experience suggests that $\xi = 0$ is appropriate. An appropriate value of the control factor γ in reinitialising zero velocity found empirically for our applications is $\gamma = 0.1$.

19.3 PSO Aided Semi-blind Joint ML Estimation

Our first application of PSO to multiple-input multiple-output (MIMO) transceiver design involves the PSO-aided semi-blind joint maximum likelihood (ML) channel estimation and data detection for MIMO receiver.

19.3.1 MIMO System Model

We consider a MIMO system consisting of n_T transmitters and n_R receivers, which communicates over flat fading channels [42]. The system is described by the well-known MIMO model [32]

$$\mathbf{y}(k) = \mathbf{H}\mathbf{x}(k) + \mathbf{n}(k), \tag{19.10}$$

where k is the symbol index, $\mathbf{H}$ denotes the $n_R \times n_T$ complex-valued MIMO channel matrix, $\mathbf{x}(k) = [x_1(k)\ x_2(k) \cdots x_{n_T}(k)]^T$ is the transmitted symbols vector of the n_T transmitters with the symbol energy given by $E\left[|x_m(k)|^2\right] = \sigma_x^2$ for $1 \le m \le n_T$, $\mathbf{y}(k) = [y_1(k)\ y_2(k) \cdots y_{n_R}(k)]^T$ denotes the received signal vector, and $\mathbf{n}(k) = [n_1(k)\ n_2(k) \cdots n_{n_R}(k)]^T$ is the complex-valued Gaussian white noise vector associated with the MIMO channels with $E\left[\mathbf{n}(k)\mathbf{n}^H(k)\right] = 2\sigma_n^2 \mathbf{I}_{n_R} = \mathrm{N_o}\mathbf{I}_{n_R}$. The signal-to-noise ratio (SNR) of the system is defined by $\mathrm{SNR} = \mathrm{E_b}/\mathrm{N_o} = \sigma_x^2/2\sigma_n^2$.

More specifically, the narrowband MIMO channel matrix is defined by $\mathbf{H} = [h_{p,m}]$, for $1 \le p \le n_R$ and $1 \le m \le n_T$, where $h_{p,m}$ denotes the channel coefficient linking the mth transmitter to the pth receiver. The fading is assumed to be sufficiently slow, so that during the time period of a short block of L symbols, all the entries in the MIMO channel matrix $\mathbf{H}$ may be deemed unchanged. From frame to frame, the channel impulse response (CIR) taps $h_{p,m}$ are independently and identically distributed (i.i.d.) complex-valued Gaussian processes with zero mean and $E\left[|h_{p,m}|^2\right] = 1$. Note that frequency selective MIMO channels can be made narrowband using for example the orthogonal frequency division multiplexing (OFDM) technique [19]. We also assume that the modulation scheme is the quadrature phase shift keying (QPSK) and, therefore, the transmitted symbol takes the value from the symbol set

$$x_i(k) \in \mathscr{X} = \{\pm 1 \pm j\}. \tag{19.11}$$

All the results discussed here are equally applicable to higher-throughput modulation schemes, such as the quadrature amplitude modulation (QAM) [20], with increased complexity.

19.3.2 Semi-blind Joint ML Channel Estimation and Data Detection

Let us consider the joint channel estimation and data detection based on the observation vector $\mathbf{y}(k)$ over a relatively short length of L symbols. First define the $n_R \times L$ matrix of the received data as $\mathbf{Y} = [\mathbf{y}(1)\ \mathbf{y}(2) \cdots \mathbf{y}(L)]$ and the corresponding $n_T \times L$ matrix of the transmitted symbols as $\mathbf{X} = [\mathbf{x}(1)\ \mathbf{x}(2) \cdots \mathbf{x}(L)]$. Then the probability density function (PDF) of the received data matrix $\mathbf{Y}$ conditioned on the MIMO channel matrix $\mathbf{H}$ and the transmitted symbol matrix $\mathbf{X}$ can be written as

$$p(\mathbf{Y}|\mathbf{H},\mathbf{X}) = \frac{1}{(2\pi\sigma_n^2)^{n_R \times L}} e^{-\frac{1}{2\sigma_n^2}\sum_{k=1}^{L}\|\mathbf{y}(k)-\mathbf{H}\mathbf{x}(k)\|^2}. \tag{19.12}$$

The ML estimation of $\mathbf{X}$ and $\mathbf{H}$ can be obtained by jointly maximising $p(\mathbf{Y}|\mathbf{H},\mathbf{X})$ over $\mathbf{X}$ and $\mathbf{H}$. Equivalently, the joint ML estimation is obtained by minimsing the cost function

$$J_{\mathrm{ML}}(\check{\mathbf{X}},\check{\mathbf{H}}) = \frac{1}{n_R \times L}\sum_{k=1}^{L}\left\|\mathbf{y}(k) - \check{\mathbf{H}}\,\check{\mathbf{x}}(k)\right\|^2, \tag{19.13}$$

which is a function of the symbol matrix $\check{\mathbf{X}} = [\check{\mathbf{x}}(1)\ \check{\mathbf{x}}(2)\cdots\check{\mathbf{x}}(L)]$ and the channel matrix $\check{\mathbf{H}}$. Thus the joint ML channel and data estimation is obtained as

$$(\hat{\mathbf{X}}, \hat{\mathbf{H}}) = \arg\left\{ \min_{\check{\mathbf{X}}, \check{\mathbf{H}}} J_{\mathrm{ML}}(\check{\mathbf{X}}, \check{\mathbf{H}}) \right\}. \tag{19.14}$$

The joint ML optimisation defined in (19.14) is computationally prohibitive. The complexity of this optimisation process may be reduced to a tractable level, if it is decomposed into an iterative search carried out over all the possible data symbols first and then over the channel matrices as

$$(\hat{\mathbf{X}}, \hat{\mathbf{H}}) = \arg\left\{ \min_{\check{\mathbf{H}}} \left[\min_{\check{\mathbf{X}}} J_{\mathrm{ML}}(\check{\mathbf{X}}, \check{\mathbf{H}}) \right] \right\}. \tag{19.15}$$

At the inner-level optimisation we can use the optimised hierarchy reduced search algorithm (OHRSA) based ML detector [2] to find the ML data estimate for the given channel. The detailed implementation of the OHRSA-aided ML detector can be found in [2] and will not be repeated here. In order to guarantee a joint ML estimate, the search algorithm used at the outer or upper-level optimisation should be capable of finding a global optimal channel estimate efficiently. A joint ML solution is achieved with the following iterative loop.

Outer-level Optimisation: A search algorithm searches the MIMO channel parameter space to find a global optimal estimate $\hat{\mathbf{H}}$ by minimising the mean square error (MSE)

$$J_{\mathrm{MSE}}(\check{\mathbf{H}}) = J_{\mathrm{ML}}(\hat{\mathbf{X}}(\check{\mathbf{H}}), \check{\mathbf{H}}), \tag{19.16}$$

where $\hat{\mathbf{X}}(\check{\mathbf{H}})$ denotes the ML estimate of the transmitted data for the given channel $\check{\mathbf{H}}$.

Inner-level Optimisation: Given $\check{\mathbf{H}}$ the OHRSA detector finds the ML estimate of the transmitted data and feeds back the ML metric $J_{\mathrm{MSE}}(\check{\mathbf{H}})$ to the upper level.

Pure blind joint data and channel estimation converges very slowly and suffers from an inherent permutation and scaling ambiguity problem [40]. To resolve this permutation and scaling ambiguity, a few training symbols are employed to provide an initial least square channel estimate (LSCE) for aiding the outer-level search algorithm. Let the number of training symbols be K, and denote the available training data as $\mathbf{Y}_K = [\mathbf{y}(1)\ \mathbf{y}(2)\cdots\mathbf{y}(K)]$ and $\mathbf{X}_K = [\mathbf{x}(1)\ \mathbf{x}(2)\cdots\mathbf{x}(K)]$. The LSCE based on $\{\mathbf{Y}_K, \mathbf{X}_K\}$ is readily given by

$$\check{\mathbf{H}}_{\mathrm{LSCE}} = \mathbf{Y}_K \mathbf{X}_K^H \left(\mathbf{X}_K \mathbf{X}_K^H\right)^{-1}. \tag{19.17}$$

To maintain the system throughput, we only use the minimum number of training symbols, namely, $K = n_T$, which is equal to the rank of the MIMO system. The training symbol matrix $\mathbf{X}_K$ should be designed to yield the optimal estimation performance [4]. Specifically, $\mathbf{X}_K$ is designed to have n_T orthogonal rows. This yields the most efficient estimate and removes the need for matrix inversion.

19.3.3 PSO Aided Semi-blind Joint ML Scheme

The above semi-blind joint ML data and channel estimation is a very expensive optimisation problem. Firstly, let us exam the inner-level optimisation. For a given channel $\check{\mathbf{H}}$, the ML data detection solution $\hat{\mathbf{X}}(\check{\mathbf{H}})$ must be calculated. Note that the data matrix $\check{\mathbf{X}}$ has $\mathscr{M}^{L\times n_R}$ legitimate combinations, where $\mathscr{M}=4$ is the size of the QPSK symbol set (19.11). A exhausted search would require to calculate the cost function (19.13) $\mathscr{M}^{L\times n_R}$ times and to find the data matrix that attains the minimum value of the cost function. This is obviously prohibitive. The OHRSA-aided ML detector [2] manages to reduce dramatically the complexity required for attainding the ML solution $\hat{\mathbf{X}}(\check{\mathbf{H}})$. Even so the OHRSA detector is by no means low-complexity and is in fact inherently expensive owing to the nature of the optimal ML detection. The detailed complexity analysis can be found for example in [46] and is beyond the scope of this contribution. Now consider the outer-level optimisation, which has to search through the $(2n_R)\times(2n_T)$ dimensional real-valued channel space. Each point evaluated requires to call the OHRSA detector once. Any search algorithm will require a large number of OHRSA evaluations in order to attain the joint ML solution $\hat{\mathbf{X}}(\hat{\mathbf{H}})$.

In the previous work [1], we have applied the repeated weighted boosting search (RWBS) algorithm [7] to perform the outer-level optimisation search of the joint ML iterative loop. The results shown in [1] demonestrate that the RWBS-aided semi-blind joint ML scheme performs well and is efficient in terms of its convergence speed. In this contribution, we show that by invoking the PSO method as the outer-level search algorithm, further performance enhancement can be achieved in terms of reduced complexity. The cost function for the PSO algorithm to optimise in this case is $F(\check{\mathbf{H}})=J_{\rm MSE}(\check{\mathbf{H}})$ with the dimensions of the search space specified by $N=n_R$ and $M=n_T$.

In Step *a) The swarm initialisation*, the initial particles are chosen as $\check{\mathbf{H}}_1^{(0)}=\check{\mathbf{H}}_{\rm LSCE}$ and

$$\check{\mathbf{H}}_i^{(0)}=\check{\mathbf{H}}_{\rm LSCE}+\varphi_h(\mathbf{1}_{n_R\times n_T}+j\mathbf{1}_{n_R\times n_T}),\ 2\le i\le S, \tag{19.18}$$

where φ_h is a uniformly distributed random variable defined in the range $[-\alpha,\ \alpha]$. Appropriate value for α is determined by experiment.

In Step *c) The swarm update*, we adopt the zero inertia weight $\xi=0$ and the TVAC mechanism (19.9). For any particle wandering outside the search space, we force it back to stay at the border of the search space. These provisions are found to be appropriate for this application empirically.

Let $C_{\rm OHRSA}(L)$ be the complexity of the OHRSA algorithm to decode the L-symbol data matrix $\mathbf{X}$ and let $N_{\rm OHRSA}$ be the number of calls for the OHRSA algorithm required by the PSO algorithm to converge. Then the complexity of the proposed semi-blind method is expressed as

$$C=N_{\rm OHRSA}\times C_{\rm OHRSA}(L), \tag{19.19}$$

where $C_{\rm OHRSA}(L)$ is given in [46], and $N_{\rm OHRSA}=S\times I_{\max}$ with $I_{\max}$ being the maximum number of iterations and S the swarm size. It can be seen that the

computational complexity of the PSO aided semi-blind joint ML estimation scheme is characterised by the number of OHRSA cost function evaluations N_{OHRSA}. Obviously, the number of iterations that the PSO algorithm requires to converge is $I_{\max}$, and the value of $I_{\max}$ depends on the choice of S. It is easily seen that the optimal choice of the swarm size S should lead to the minimum value of N_{OHRSA}.

19.3.4 Simulation Study

A simulation study was carried out to investigate the PSO aided semi-blind joint ML channel estimation and data detection scheme. We considered the benchmark MIMO system with $n_T = 4$ and $n_R = 4$ used in [1]. The achievable performance was assessed in the simulation using three metrics, and these were the MSE defined in (19.16), the mean channel error (MCE) defined as

$$J_{\text{MCE}}(\check{\mathbf{H}}) = \|\mathbf{H} - \check{\mathbf{H}}\|^2, \tag{19.20}$$

where $\mathbf{H}$ denotes the true MIMO channel matrix and $\check{\mathbf{H}}$ the channel estimate, and the bit error rate (BER). All the simulation results were averaged over 50 different channel realisations of $\mathbf{H}$.

We set the population size to $S = 20$, which led to the maximum number of evolutionary steps $I_{\max} = 50$. This choice of S appeared to be adequate for this application as it resulted in the smallest N_{OHRSA} for the algorithm to converge. Thus, the complexity of the PSO based semi-blind scheme was determined by $N_{\text{OHRSA}} = 1000$. Since $\Re[h_{p,q}]$ and $\Im[h_{p,q}]$ of each MIMO channel tap $h_{p,q}$ were Gaussian distributed with a variance 0.5, we chose the search space bound to be $U_{\max} = 1.8$

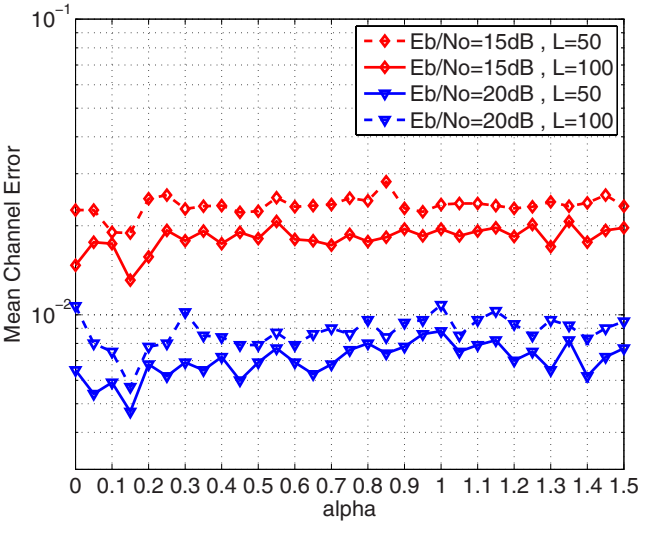


Fig. 19.2 Mean channel error average over 50 different channel realisations as a function of α after 1000 OHRSA evaluations, for two values of $\text{E}_\text{b}/\text{N}_\text{o}$ and two values of L

which lay between 2 to 3 standard deviations of the true tap distribution. We also set the velocity limit to $V_{\max} = 1.0$ which was confirmed in simulation to be a suitable value for this application. The control factor γ in reinitialising zero velocity was found empirically to be $\gamma = 0.1$. The optimal value for the control parameter α in the channel population initiation (19.18) was first found empirically. Fig. 19.2 shows the MCE performance after 1000 OHRSA evaluations over a range of α values. It can be seen from Fig. 19.2 that the optimal value of α in this case was 0.15. This value of α was used in all the other simulations.

Fig. 19.3 depicts the BER performance of the PSO based semi-blind scheme having a frame length $L = 100$ after 1000 OHRSA evaluations and averaging over 50 different channel realisations, in comparison with the performance of the training-based OHRSA detector having $K = 4$, 8 and 16 training symbols for the LSCE, respectively, as well as with the case of perfect channel knowledge. It can be observed from Fig. 19.3 that, for the training-based scheme to achieve the same BER performance of the PSO-aided semi-blind one having only 4 pilot symbols, the number of training symbols had to be more than 16. This example was identical to the MIMO system investigated in [1]. The BER performance of the PSO-based semi-blind scheme depicted in Fig. 19.3 was slightly better than the BER of the RWBS-based semi-blind scheme shown in [1]. Moreover, the performance of the PSO-aided scheme was achieved after 1000 OHRSA evaluations, while the performance of the RWBS-based scheme reported in [1] was obtained after 1200 OHRSA evaluations. Thus, for this 4×4 MIMO benchmark, the computational saving achieved by the

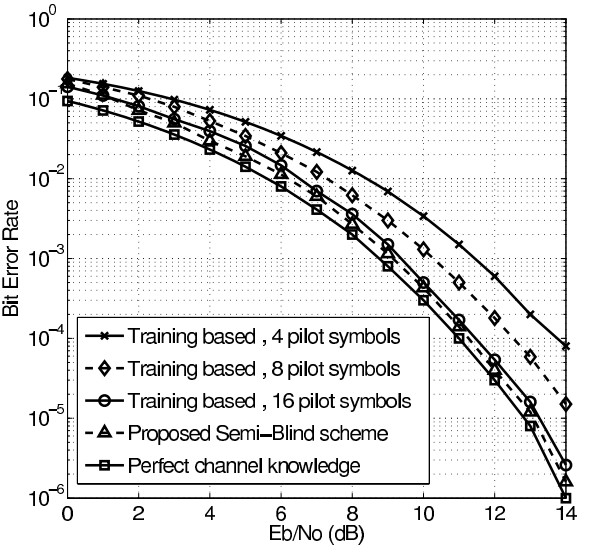


Fig. 19.3 BER of the PSO aided semi-blind scheme with frame length $L = 100$ after 1000 OHRSA evaluations and average over 50 different channel realisations in comparison with the training-based cases using 4, 8 and 16 pilot symbols as well as the case of perfect channel knowledge

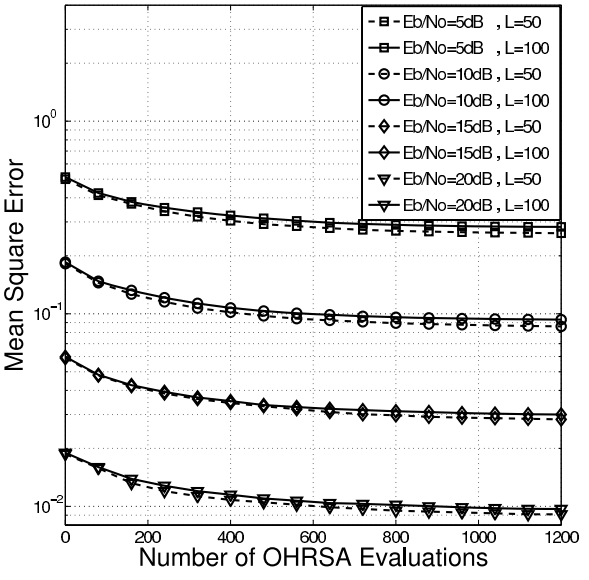


Fig. 19.4 Mean square error convergence performance of the PSO aided semi-blind scheme averaged over 50 different channel realisations for different values of E_b/N_o and L

proposed PSO-based semi-blind method over the previous RWBS-based scheme was approximately

$$\frac{1200 \times C_{\text{OHRSA}}(L) - 1000 \times C_{\text{OHRSA}}(L)}{1000 \times C_{\text{OHRSA}}(L)} = 20\%. \tag{19.21}$$

Figs. 19.4 and 19.5 depict the convergence performance of the proposed PSO-aided semi-blind joint ML channel estimation and data detection scheme averaged over 50 different channel realisations in terms of the MSE and MCE, respectively, for different SNR values as well as for two frame lengths $L = 50$ and 100. It can be seen from Fig. 19.4 that the MSE converged to the noise floor. The MCE performance shown in Fig. 19.5 was seen to be slightly better and converging faster than the results obtained by the RWBS-based semi-blind joint ML scheme shown in [1].

19.4 PSO Based MBER Multiuser Transmitter Design

In this second application, we adopt the PSO for designing the minimum BER (MBER) multiuser transmission (MUT) for the downlink of a space-division multiple-access (SDMA) induced MIMO system.

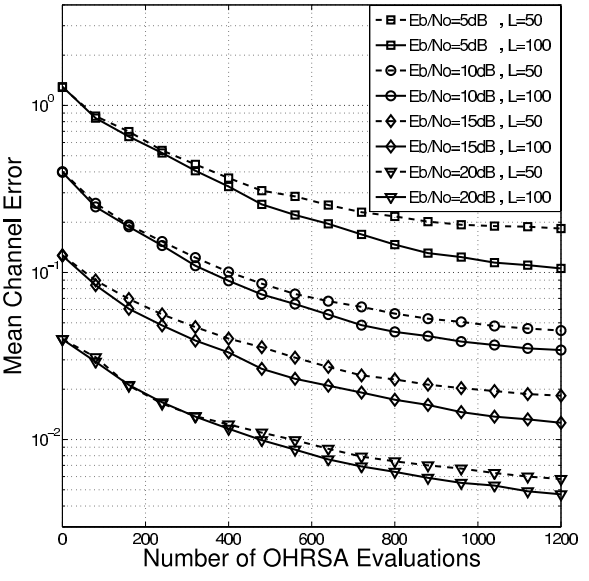


Fig. 19.5 Mean channel error convergence performance of the PSO aided semi-blind scheme averaged over 50 different channel realisations for different values of $\mathrm{E_b/N_o}$ and L

19.4.1 Downlink of SDMA Induced MIMO System

In the downlink of the SDMA induced MIMO system, the base station (BS) equipped with n_T transmit antennas communicates over flat fading channels with n_R mobile terminals (MTs), each employing a single-receive antenna. Again we point out that frequency selective channels can be converted to a multiplicity of parallel narrowband channels using the OFDM technique [19]. Let the vector of n_R information symbols transmitted in the downlink be $\mathbf{x}(k) = [x_1(k)\ x_2(k) \cdots x_{n_R}(k)]^T$, where k denotes the symbol index, $x_m(k)$ denotes the transmitted symbol to the mth MT, and the symbol energy is given by $E[|x_m(k)|^2] = \sigma_x^2$, for $1 \le m \le n_R$. The modulation scheme is again assumed to be the QPSK of the symbol set (19.11), but the extension to the generic QAM modulation scheme can be achieved by considering the minimum symbol error rate criterion [9]. The $n_T \times n_R$ precoder matrix $\mathbf{C}$ of the BS's MUT is defined by

$$\mathbf{C} = [\mathbf{c}_1\ \mathbf{c}_2 \cdots \mathbf{c}_{n_R}], \tag{19.22}$$

where $\mathbf{c}_m$, $1 \le m \le n_R$, is the precoder's coefficient vector for pre-processing the mth user's data stream. Given a fixed total transmit power $\mathrm{E_T}$ at the BS, an appropriate scaling factor should be used to fullfill this transmit power constraint, which is defined as

$$\rho = \sqrt{\mathrm{E_T}/E[\|\mathbf{C}\mathbf{x}(k)\|^2]}. \tag{19.23}$$

Thus, the signal vector to be launched from the n_T transmit antennas is $\rho\mathbf{C}\mathbf{x}(k)$.

The downlink of the SDMA system is specified by its channel matrix $\mathbf{H}$, which is given by

$$\mathbf{H} = [\mathbf{h}_1\ \mathbf{h}_2 \cdots \mathbf{h}_{n_R}], \tag{19.24}$$

where $\mathbf{h}_m = [h_{1,m}\ h_{2,m} \cdots h_{n_T,m}]^T$, $1 \le m \le n_R$, is the mth user's spatial signature. The channel taps $h_{i,m}$ for $1 \le i \le n_T$ and $1 \le m \le n_R$ are independent of each other and obey the complex-valued Gaussian distribution with $E[|h_{i,m}|^2] = 1$. At the receiver, the reciprocal of the scaling factor, namely ρ^{-1}, is used to scale the received signal to ensure unity-gain transmission, and the baseband model of the system can be described as

$$\mathbf{y}(k) = \rho^{-1}\mathbf{H}^T \rho\, \mathbf{C}\mathbf{x}(k) + \rho^{-1}\mathbf{n}(k) = \mathbf{H}^T \mathbf{C}\mathbf{x}(k) + \rho^{-1}\mathbf{n}(k), \tag{19.25}$$

where $\mathbf{n}(k) = [n_1(k)\ n_2(k) \cdots n_{n_R}(k)]^T$ is the channel additive white Gaussian noise vector, $n_m(k)$, $1 \le m \le n_R$, is a complex-valued Gaussian random process with zero mean and $E[|n_m(k)|^2] = 2\sigma_n^2 = \mathrm{N_o}$, and $\mathbf{y}(k) = [y_1(k)\ y_2(k) \cdots y_{n_R}(k)]^T$ denotes the received signal vector. Note that $y_m(k)$, $1 \le m \le n_R$, constitutes sufficient statistics for the mth MT to detect the transmitted data symbol $x_m(k)$. The SNR of the downlink is defined as $\mathrm{SNR} = \mathrm{E_b}/\mathrm{N_o}$, where $\mathrm{E_b} = \mathrm{E_T}/(n_T \log_2 \mathscr{M})$ is the energy per bit per antenna for $\mathscr{M}$-ary modulation. In our case, $\mathscr{M} = 4$.

19.4.2 MBER MUT Design

The minimum mean square error (MMSE) MUT design, denoted as $\mathbf{C}_{\mathrm{TxMMSE}}$, is popular owing to its appealing simplicity [44, 51], but it does not minimise the achievable system's BER. The average BER of the in-phase component of $\mathbf{y}(k)$ at the receiver is given by [8]

$$P_{e_I}(\mathbf{C}) = \frac{1}{n_R \mathscr{M}^{n_R}} \sum_{q=1}^{\mathscr{M}^{n_R}} \sum_{m=1}^{n_R} Q\left(\frac{\mathrm{sgn}(\Re[x_m^{(q)}])\Re[\mathbf{h}_m^T \mathbf{C}\mathbf{x}^{(q)}]}{\sigma_n} \right), \tag{19.26}$$

where $Q(\ )$ is the standard Gaussian error function, $\mathscr{M}^{n_R} = 4^{n_R}$ is the number of equiprobable legitimate transmit symbol vectors $\mathbf{x}^{(q)}$ for QPSK signalling (19.11) and $x_m^{(q)}$ the mth element of $\mathbf{x}^{(q)}$, with $1 \le q \le \mathscr{M}^{n_R}$. Similarly, the average BER of the quadrature-phase component of $\mathbf{y}(k)$ can be shown to be [8]

$$P_{e_Q}(\mathbf{C}) = \frac{1}{n_R \mathscr{M}^{n_R}} \sum_{q=1}^{\mathscr{M}^{n_R}} \sum_{k=1}^{n_R} Q\left(\frac{\mathrm{sgn}(\Im[x_m^{(q)}])\Im[\mathbf{h}_m^T \mathbf{C}\mathbf{x}^{(q)}]}{\sigma_n} \right). \tag{19.27}$$

Thus the average BER of the MUT with the precoder matrix $\mathbf{C}$ is given by

$$P_e(\mathbf{C}) = \big(P_{e_I}(\mathbf{C}) + P_{e_Q}(\mathbf{C})\big)/2, \tag{19.28}$$

Table 19.1 Computational complexity per iteration of two MBER MUT designs for QPSK signalling, where n_T is the number of transmit antennas, n_R the number of mobile terminals, $\mathscr{M} = 4$ is the size of symbol constellation and S is the swarm size

Algorithm	Flops
SQP	$n_R \times (8 \times n_T^2 \times n_R^2 + 6 \times n_T \times n_R + 6 \times n_T + 8 \times n_R + 4) \times \mathscr{M}^{n_R}$ $+\mathscr{O}(8 \times n_T^3 \times n_R^3) + 8 \times n_T^2 \times n_R^2 + 16 \times n_T \times n_R^2 + 8 \times n_T^2 \times n_R$ $+12 \times n_T \times n_R + 6 \times n_R^2 - 2 \times n_T^2 + n_T - 2 \times n_R + 11$
PSO	$((16 \times n_T \times n_R + 7 \times n_R + 6 \times n_T + 1) \times \mathscr{M}^{n_R} + 20 \times n_T \times n_R + 2) \times S + 8$

and the solution of the average MBER MUT is defined as

$$\begin{aligned} \mathbf{C}_{\text{TxMBER}} &= \arg\min_{\mathbf{C}} P_e(\mathbf{C}) \\ &\text{s.t. } E[\|\mathbf{C}\mathbf{x}(k)\|^2] = \mathrm{E_T}. \end{aligned} \tag{19.29}$$

The optimisation problem (19.29) is a constrained nonlinear optimisation problems, and it is typically solved by an iterative gradient based optimisation algorithm known as the SQP [21, 23, 39]. The computational complexity per iteration of the SQP-based MBER MUT, quoted from [39], is listed in Table 19.1 for QPSK modulation, where $\mathscr{O}(8 \times n_T^3 \times n_R^3)$ stands for order of $8 \times n_T^3 \times n_R^3$ complexity and we assume that the complexity of a real-valued multiplication is equal to a real-valued addition. Note that $\mathscr{O}(8 \times n_T^3 \times n_R^3)$ is the complexity for matrix inversion required by the SQP algorithm, and the exact value of $\mathscr{O}(8 \times n_T^3 \times n_R^3)$ depends on the inversion algorithm employed. The total computational complexity equals the number of iterations that the algorithm required to arrive at a global optimal solution multiplied by this complexity per iteration.

19.4.3 PSO Aided MBER MUT Design

In practice, the computational complexity of the SQP based MBER-MUT solution may be excessively high for high-rate systems [23]. In this contribution, we invoke the PSO algorithm to solve the MBER-MUT design (19.29) in order to bring down the computational complexity to a practically acceptable level. A penalty function approach is adopted to convert the constrained optimisation process (19.29) into the unconstrained one and to automatically perform power allocation in order to meet the transmit power constraint. Let us define the cost function for the PSO algorithm to minimise as

$$F(\mathbf{C}) = P_e(\mathbf{C}) + G(\mathbf{C}) \tag{19.30}$$

with the penalty function given by

$$G(\mathbf{C}) = \begin{cases} 0, & E[\|\mathbf{C}\mathbf{x}(k)\|^2] - \mathrm{E_T} \le 0, \\ \lambda(E[\|\mathbf{C}\mathbf{x}(k)\|^2] - \mathrm{E_T}), & E[\|\mathbf{C}\mathbf{x}(k)\|^2] - \mathrm{E_T} > 0. \end{cases} \tag{19.31}$$

With an appropriately chosen penalty factor λ, the MBER-MUT design (19.29) can be obtained as the solution of the following unconstrained optimisation

$$\mathbf{C}_{\text{TxMBER}} = \arg\min_{\mathbf{C}} F(\mathbf{C}). \tag{19.32}$$

The value of λ is linked to the value of SNR. Since the BS has the knowledge of the downlink SNR, it is not difficult at all to assign an appropriate λ value. The dimensions of the search space for the PSO optimisation are specified by $N = n_T$ and $M = n_R$.

In Step *a) The swarm initialisation*, we set $\mathbf{C}_1^{(0)} = \mathbf{C}_{\text{TxMMSE}}$, the MMSE MUT solution, and randomly generate the rest of the initial particles, $\{\mathbf{C}_i^{(0)}\}_{i=2}^{S}$, in the search space $\mathsf{U}^{n_T \times n_R}$.

In Step *c) The swarm update*, we adopt the zero inertia weight $\xi = 0$ and the TVAC mechanism (19.8). If a particle wanders outside the search space, we move it back inside the search space randomly rather than forcing it to stay at the border of the search space. These measures are tested empirically to be appropriate for this application.

The computational complexity per iteration for the PSO-aided MBER-MUT scheme is also listed in Table 19.1. We will demonstrate that the PSO-aided MBER MUT design imposes a considerably lower complexity than the SQP based MBER MUT design. This is owing to the fact that the designed PSO algorithm is very efficient in searching through the precoder's parameter space to find an optimal solution, as demonstrated in the following simulation study.

19.4.4 Simulation Study

We considered the downlink of a multiuser system that employed $n_T = 4$ transmit antennas at the BS to communicate over the 4×4 flat Rayleigh fading MIMO channels to $n_R = 4$ single-receive-antenna QPSK MTs. The size of the swarm was chosen to be $S = 20$, and the corresponding maximum number of iterations for the PSO algorithm to arrive at the MBER performance was in the range of $I_{\max} = 20$ to 40, depending on the value of the downlink SNR. The choice of $S = 20$ was appropriate in this application as it led to the lowest computational cost for the algorithm to converge. Our empirical results suggested that the search limit $U_{\max} = 1.0$ and the velocity bound $V_{\max} = 1.0$ were appropriate for this application. The control factor γ in avoiding zero velocity was found to be $\gamma = 0.1$ by experiments. All the simulation results were obtained by averaging over 100 different channel realisations.

We first assumed the perfect channel state information (CSI) at the BS. Fig. 19.6 compares the BER performance of the MMSE-MUT scheme with that of the PSO-based MBER-MUT scheme. It can be seen from Fig. 19.6 that, given the perfect CSI, the PSO-aided MBER-MUT provided an SNR gain of 3 dB over the MMSE-MUT scheme at the target BER level of 10^{-4}. The robustness of the PSO-aided MBER-MUT design to channel estimation error was next investigated by adding a complex-valued Gaussian white noise with a standard deviation of 0.05 per

dimension to each channel tap $h_{i,m}$ to represent channel estimation error. The BERs of the MMSE-MUT and the PSO-based MBER-MUT under this channel estimation error are also plotted in Fig. 19.6. It can be seen that the PSO-aided MBER-MUT design was no more sensitive to channel estimation error than the MMSE-MUT design. The convergence performance and computational requirements of the PSO-aided MBER-MUT design were investigated, using the SQP-based MBER-MUT counterpart as the benchmark. Fig. 19.7 compares the convergence performance of the SQP-based and PSO-aided MBER MUT schemes, operating at the SNR values of $\mathrm{E_b/N_o} = 10$ dB and 15 dB, respectively.

At the SNR of 10 dB, it can be seen from Fig. 19.7 that the SQP algorithm converged to the MBER-MUT solution after 100 iterations, while the PSO counterpart arrived at the same MBER-MUT solution after 20 iterations. Fig. 19.8 shows the computational complexities required by the SQP-based and PSO-aided MBER-MUT designs, respectively, to arrive at the MBER MUT solution, in term of (a) the total number of operations (Flops) and (b) the total run time (seconds) recorded. In deriving the number of operations required by the SQP algorithm, we had approximated $\mathcal{O}(8 \times n_T^3 \times n_R^3)$ by $8 \times n_T^3 \times n_R^3$. It can be observed from Fig. 19.8 (a) that the SQP-based algorithm needed 229,351,100 Flops to converge to the MBER-MUT solution, while the PSO-aided algorithm converged to the same MBER-MUT solution at the cost of 34,561,760 Flops. Therefore, the PSO-aided MBER-MUT design imposed an approximately seven times lower complexity than the SQP counterpart

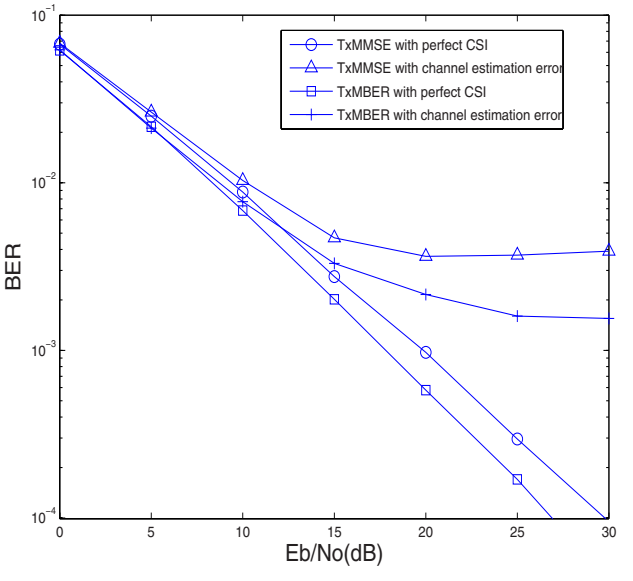


Fig. 19.6 BER versus SNR performance of the PSO-aided MBER-MUT communicating over flat Rayleigh fading channels using $n_T = 4$ transmit antennas to support $n_R = 4$ QPSK MTs, in comparison with the benchmark MMSE-MUT

for this scenario. From Fig. 19.8 (b), it can be seen that the SQP-based design required 1730.6 seconds to converge to the optimal MBER-MUT solution, while the PSO-aided design only needed 257.3 seconds to arrive at the same optimal MBER-MUT solution. This also confirms that the PSO-aided MBER-MUT scheme was approximately seven times faster than the SQP-based counterpart in this case.

From Fig. 19.7 it can also been that, with the SNR of 15 dB, the SQP based algorithm converged after 140 iterations, which required a total cost of 321,091,540 Flops, while the PSO-aided scheme archived the convergence after 40 iterations, which required a total cost of 63,541,120 Flops. Thus, the PSO-aided design imposed an approximately five times lower complexity than the SQP counterpart in this scenario.

Further investigation showed that the convergence results obtained for SNR$<$ 10 dB were similar to the case of SNR$=$ 10 dB, while the convergence results obtained under SNR$>$ 15 dB agreed with the case of SNR$=$ 15 dB. Thus, we may conclude that for this 4×4 MIMO benchmark the PSO-aided MBER-MUT design imposed approximately five to seven times lower complexity than the SQP-based MBER-MUT counterpart.

Finally, we showed that why the choice of the swarm size $S = 20$ was optimal in this application. Fig. 19.9 illustrates the convergence performance and the total required complexity for the PSO-aided algorithm with the different swarm sizes of $S = 10, 20, 30$ and 40 at the SNR value of 15 dB. It is clear that $S = 10$ was too small

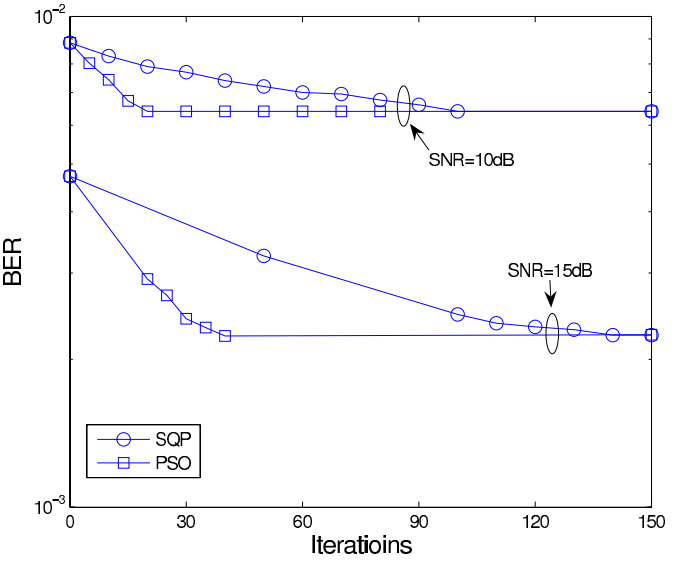
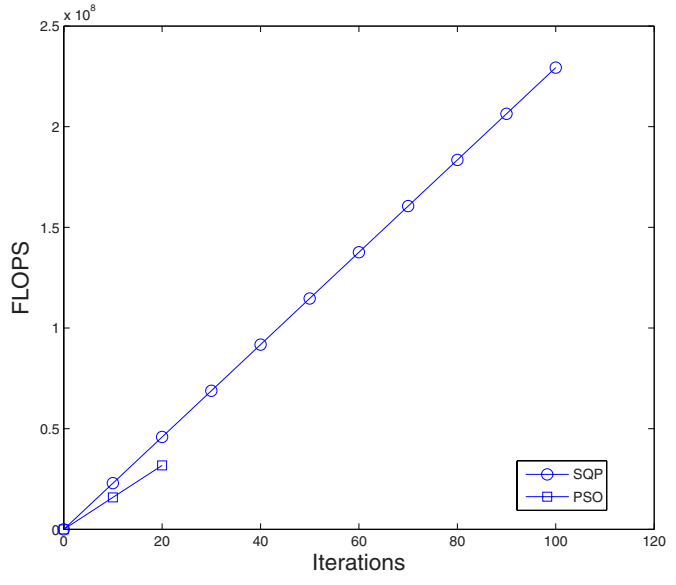
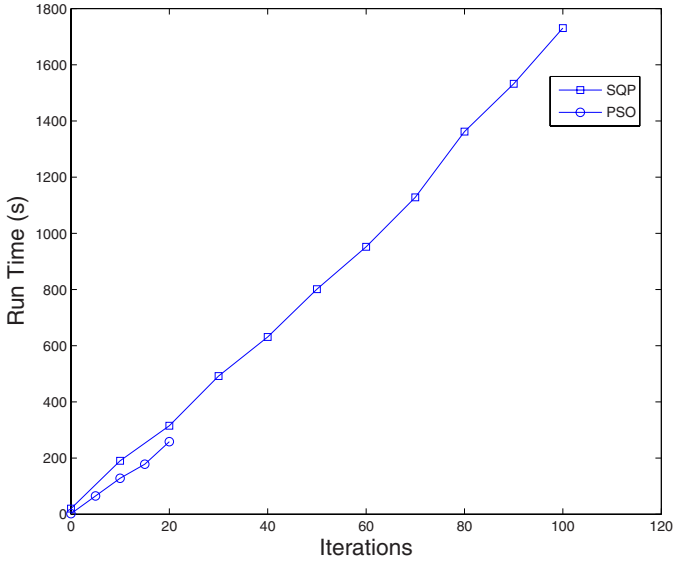


Fig. 19.7 Convergence performance of the SQP-based and PSO-aided MBER-MUT schemes for the system employing $n_T = 4$ transmit antennas to support $n_R = 4$ QPSK MTs over flat Rayleigh fading channels at $\mathrm{E_b}/\mathbf{N_o} = 10$ dB and 15 dB, respectively

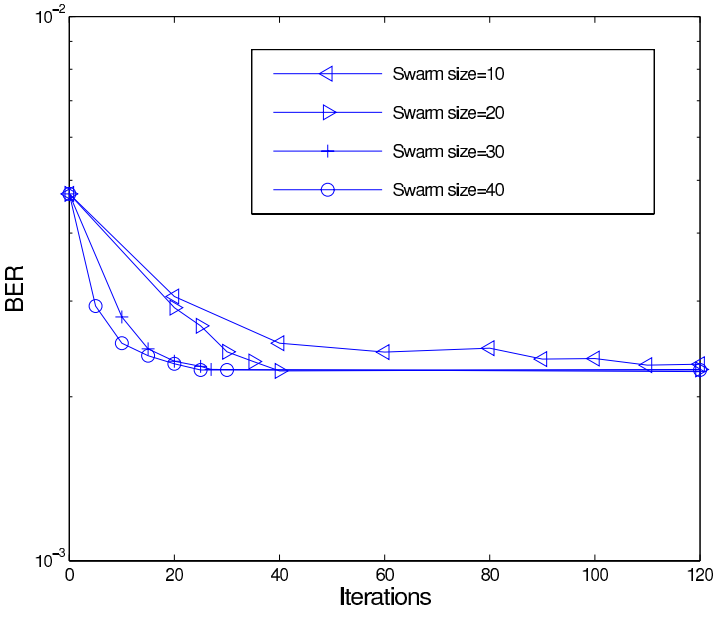


(a)

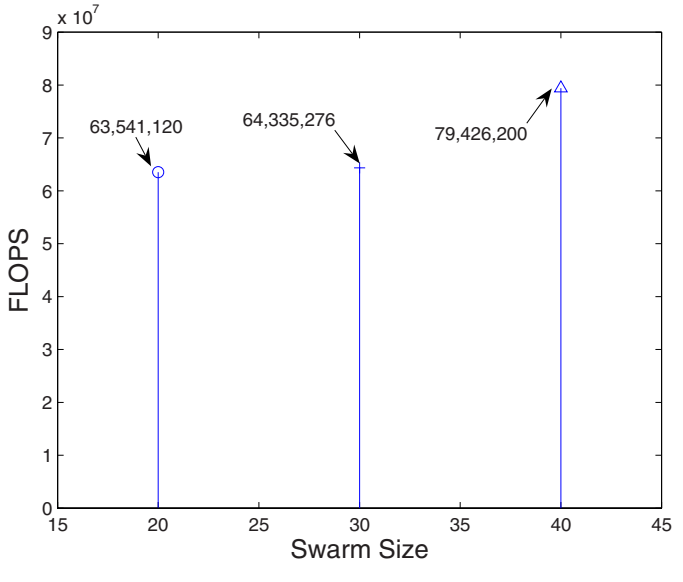


(b)

Fig. 19.8 Complexity comparison of the SQP-based and PSO-aided MBER-MUT schemes for the system employing $n_T = 4$ transmit antennas to support $n_R = 4$ QPSK MTs over flat Rayleigh fading channels at $\mathrm{E_b}/\mathbf{N}_\mathrm{o} = 10$ dB, in terms of (a) number of FLOPs, and (b) run time (seconds)



(a)



(b)

Fig. 19.9 Convergence performance (a) and required total complexity (b) of the PSO-aided MBER-MUT scheme with different swarm sizes for the system employing $n_T = 4$ transmit antennas to support $n_R = 4$ QPSK MTs over flat Rayleigh fading channels at $\mathrm{E_b}/\mathbf{N_o} = 15$ dB

for the algorithm to converge to the optimal MBER-MUT solution in this case. The results of Fig. 19.9 also show that with $S = 20$ the algorithm took 40 iterations to converge at the cost of 63,541,120 Flops, and with $S = 30$ it needed 27 iterations at the cost of 64,335,276 Flops, while the algorithm given $S = 40$ only required 25 iterations to converge but its cost was 79,426,200 Flops. Thus the choice of $S = 20$ led to the lowest computational cost for the algorithm to converge in this application.

19.5 Conclusions

State-of-the-art MIMO transceiver designs impose expensive optimisation problems, which require the applications of sophisticated and advanced optimisation techniques, such as evolutionary computation methods, in order to achieve the optimal performance offered by MIMO technologies at practically affordable cost. In this contribution, we have demonstrated that the PSO provides an efficient tool for aiding MIMO transceiver designs. Specifically, we have applied the PSO algorithm to the semi-blind joint ML channel estimation and data detection for MIMO receiver, which offers significant complexity saving over an existing state-of-the-art RWBS-based scheme. Furthermore, we have employed the PSO to design the MBER MUT scheme for the downlink of a SDMA induced MIMO system, which imposes much lower computational complexity than the available SQP-based MBER MUT design.

The Communication Research Group at the University of Southampton has actively engaged in research of state-of-the-art MIMO transceiver designs using various powerful evolutionary computation methods for a long time. In particular, we have extensive experience using the genetic algorithm [3, 5, 6, 22, 24, 53] and the ant colony optimisation [47, 48, 49] for MUD designs. Further research is warranted to further investigate various evolutionary computation methods in benchmark MIMO designs and to study their performance-complexity trade-offs with the aim of providing useful guidelines for aiding practical MIMO system designs.

References

1. Abuthinien, M., Chen, S., Hanzo, L.: Semi-blind joint maximum likelihood channel estimation and data detection for MIMO systems. IEEE Signal Processing Letters 15, 202–205 (2008)
2. Akhtman, J., Wolfgang, A., Chen, S., Hanzo, L.: An optimized-hierarchy-aided approximate Log-MAP detector for MIMO systems. IEEE Trans. Wireless Communications 6(5), 1900–1909 (2007)
3. Alias, M.Y., Chen, S., Hanzo, L.: Multiple antenna aided OFDM employing genetic algorithm assisted minimum bit error rate multiuser detection. IEEE Trans. Vehicular Technology 54(5), 1713–1721 (2005)
4. Biguesh, M., Gershman, A.B.: Training-based MIMO channel estimation: A study of estimator tradeoffs and optimal training signals. IEEE Trans. Signal Processing 54(3), 884–893 (2006)

5. Chen, S., Wu, Y., McLaughlin, S.: Genetic algorithm optimisation for blind channel identification with higher-order cumulant fitting. IEEE Trans. Evolutionary Computation 1(4), 259–266 (1997)
6. Chen, S., Wu, Y.: Maximum likelihood joint channel and data estimation using genetic algorithms. IEEE Trans. Signal Processing 46(5), 1469–1473 (1998)
7. Chen, S., Wang, X.X., Harris, C.J.: Experiments with repeating weighted boosting search for optimization in signal processing applications. IEEE Trans. System, Man and Cybernetics, Part B 35(4), 682–693 (2005)
8. Chen, S., Hanzo, L., Ahmad, N.N., Wolfgang, A.: Adaptive minimum bit error rate beamforming assisted receiver for QPSK wireless communication. Digital Signal Processing 15(6), 545–567 (2005)
9. Chen, S., Livingstone, A., Du, H.-Q., Hanzo, L.: Adaptive minimum symbol error rate beamforming assisted detection for quadrature amplitude modulation. IEEE Trans. Wireless Communications 7(4), 1140–1145 (2008)
10. Das, S., Konar, A.: A swarm intelligence approach to the synthesis of two-dimensional IIR filters. Engineering Applications of Artificial Intelligence 20(8), 1086–1096 (2007)
11. El-Mora, H.H., Sheikh, A.U., Zerguine, A.: Application of particle swarm optimization algorithm to multiuser detection in CDMA. In: Proc. 16th IEEE Int. Symp. Personal, Indoor and Mobile Radio Communications, Berlin, Germany, September 11-14, vol. 4, pp. 2522–2526 (2005)
12. Fang, W., Sun, J., Xu, W.-B.: Design IIR digital filters using quantum-behaved particle swarm optimization. In: Proc. 2nd Int. Conf. Natural Computation, Part II, Xian, China, September 24-28, pp. 637–640 (2006)
13. Feng, H.-M.: Self-generation RBFNs using evolutional PSO learning. Neurocomputing 70(1-3), 41–251 (2006)
14. Foschini, G.J.: Layered space-time architecture for wireless communication in a fading environment when using multiple antennas. Bell Labs Tech. J. 1(2), 41–59 (1996)
15. Foschini, G.J., Gans, M.J.: On limits of wireless communications in a fading environment when using multiple antennas. Wireless Personal Communications 6(3), 311–335 (1998)
16. Guerra, F.A., Coelho, L.S.: Multi-step ahead nonlinear identification of Lorenz's chaotic system using radial basis function neural network with learning by clustering and particle swarm optimiszation. Chaos, Solitons and Fractals 35(5), 967–979 (2008)
17. Guo, Z., Xiao, Y., Lee, M.H.: Multiuser detection based on particle swarm optimization algorithm over multipath fading channels. IEICE Trans. Communications E90-B(2), 421–424 (2007)
18. Guru, S.M., Halgamuge, S.K., Fernando, S.: Particle swarm optimisers for cluster formation in wireless sensor networks. In: Proc. 2005 Int. Conf. Intelligent Sensors, Sensor Networks and Information Processing, Melbourne, Australia, December 5-8, pp. 319–324 (2005)
19. Hanzo, L., Münster, M., Choi, B.J., Keller, T.: OFDM and MC-CDMA for Broadband Multi-User Communications, WLANs and Broadcasting. John Wiley and IEEE Press, Chichester (2003)
20. Hanzo, L., Ng, S.X., Keller, T., Webb, W.: Quadrature Amplitude Modulation: From Basics to Adaptive Trellis-Coded, Turbo-Equalised and Space-Time Coded OFDM, CDMA and MC-CDMA Systems. John Wiley and IEEE Press, Chichester (2004)
21. Hjørungnes, A., Diniz, P.S.R.: Minimum BER prefilter transform for communications systems with binary signaling and known FIR MIMO channel. IEEE Signal Processing Letters 12(3), 234–237 (2005)

22. Hua, W.: Interference Suprression in Single- and Multi-Carrier CDMA Systems. PhD Thesis, School of Electronics and Computer Science, University of Southampton, Southampton, UK (2005)
23. Irmer, R.: Multiuser Transmission in Code Division Multiple Access Mobile Communication Systems. PhD Thesis, Technique University of Dresden, Dresden, Germany (2005)
24. Jiang, M.: Hybrid Multi-user OFDM Uplink Systems Using Multiple Antennas. PhD Thesis, School of Electronics and Computer Science, University of Southampton, Southampton, UK (2005)
25. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. 1995 IEEE Int. Conf. Neural Networks, Perth, Australia, November 27-December 1, vol. 4, pp. 1942–1948 (1995)
26. Kennedy, J., Eberhart, R.: Swarm Intelligence. Morgan Kaufmann, San Francisco (2001)
27. Leong, W.-F., Yen, G.G.: PSO-based multiobjective optimization with dynamic population size and adaptive local archives. IEEE Trans. Systems, Man and Cybernetics, Part B 38(5), 1270–1293 (2008)
28. Liu, H., Li, J.: A particle swarm optimization-based multiuser detection for receive-diversity-aided STBC systems. IEEE Signal Processing Letters 15, 29–32 (2008)
29. Lu, Z., Yan, S.: Multiuser detector based on particle swarm algorithm. In: Proc. 6th IEEE CAS Symp. Emerging Technologies: Frontiers of Mobile and Wireless Communication, Shanghai, China, May 31-June 2, vol. 2, pp. 783–786 (2004)
30. Marzetta, T.L., Hochwald, B.M.: Capacity of a mobile multiple-antenna communication link in Rayleigh flat fading. IEEE Trans. Information Theory 45(1), 139–157 (1999)
31. Nocedal, J., Wright, S.J.: Numerical Optimization. Springer, New York (1999)
32. Paulraj, A., Nabar, R., Gore, D.: Introduction to Space-Time Wireless Communications. Cambridge University Press, Cambridge (2003)
33. Paulraj, A.J., Gore, D.A., Nabar, R.U., Bölcskei, H.: An overview of MIMO communications – A key to gigabit wireless. Proc. IEEE 92(2), 198–218 (2004)
34. Pham, D., Pattipati, K.R., Willet, P.K., Luo, J.: An improved complex sphere decoder for V-BLAST systems. IEEE Signal Processing Letters 11(9), 748–751 (2004)
35. Ratnaweera, A., Halgamuge, S.K., Watson, H.C.: Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans. Evolutionary Computation 8(3), 240–255 (2004)
36. Soo, K.K., Siu, Y.M., Chan, W.S., Yang, L., Chen, R.S.: Particle-swarm-optimization-based multiuser detector for CDMA communications. IEEE Trans. Vehicular Technology 56(5), 3006–3013 (2007)
37. Sun, J., Xu, W.-B., Liu, J.: Training RBF neural network via quantum-behaved particle swarm optimization. In: King, I., Wang, J., Chan, L.-W., Wang, D. (eds.) ICONIP 2006. LNCS, vol. 4233, pp. 1156–1163. Springer, Heidelberg (2006)
38. Sun, T.-Y., Liu, C.-C., Tsai, T.-Y., Hsieh, S.-T.: Adequate determination of a band of wavelet threshold for noise cancellation using particle swarm optimization. In: Proc. CEC 2008, Hong Kong, China, June 1-6, pp. 1168–1175 (2008)
39. Tan, S.: Minimum Error Rate Beamforming Transceivers. PhD Thesis, School of Electronics and Computer Science, University of Southampton, Southampton, UK (2008)
40. Tang, L., Liu, R.W., Soon, V.C., Huang, Y.F.: Indeterminacy and identifiability of blind identification. IEEE Trans. Circuits and Systems 38(5), 499–509 (1991)
41. Telatar, I.E.: Capacity of multi-antenna Gaussian channels. European Trans. Telecommunications 10(6), 585–595 (1999)

42. Tse, D., Viswanath, P.: Fundamentals of Wireless Communication. Cambridge University Press, Cambridge (2005)
43. Vandenameele, P., van Der Perre, L., Engels, M.: Space Division Multiple Access for Wireless Local Area Networks. Kluwer, Boston (2001)
44. Vojčić, B.R., Jang, W.M.: Transmitter precoding in synchronous multiuser communications. IEEE Trans. Communications 46(10), 1346–1355 (1998)
45. Winters, J.H.: Smart antennas for wireless systems. IEEE Personal Communications 5(1), 23–27 (1998)
46. Wolfgang, A.: Single-Carrier Time-Domain Space-Time Equalization Algorithms for the SDMA Uplink. PhD Thesis, School of Electronics and Computer Science, University of Southampton, Southampton, UK (2007)
47. Xu, C., Yang, L.-L., Hanzo, L.: Ant-colony-based multiuser detection for MC DS-CDMA systems. In: Proc. VTC 2007-Fall, Baltimore, USA, September 30-October 2, pp. 960–964 (2007)
48. Xu, C., Yang, L.-L., Maunder, R.G., Hanzo, L.: Near-optimum soft-output ant-colony-optimization based multiuser detection for the DS-CDMA. In: Proc. ICC 2008, Beijing, China, pp. 795–799 (2008)
49. Xu, C., Hu, B., Yang, L.-L., Hanzo, L.: Ant-colony-based multiuser detection for multi-functional antenna array assisted MC DS-CDMA systems. IEEE Trans. Vehicular Technology 57(1), 658–663 (2008)
50. Yang, L.-L.: Design of linear multiuser transmitters from linear multiuser receivers. In: Proc. ICC 2007, Glasgow, UK, June 24-28, pp. 5258–5263 (2007)
51. Yang, D., Yang, L.-L., Hanzo, L.: Performance of SDMA systems using transmitter preprocessing based on noisy feedback of vector-quantized channel impulse responses. In: Proc. VTC2007-Spring, Dublin, Ireland, pp. 2119–2123 (2007)
52. Yao, W., Chen, S., Tan, S., Hanzo, L.: Particle swarm optimisation aided minimum bit error rate multiuser transmission. In: Proc. ICC 2009, Dresden, Germany, 5 pages (2009)
53. Yen, K.: Genetic Algorithm Assisted CDMA Multiuser Detection. PhD Thesis, School of Electronics and Computer Science, University of Southampton, Southampton, UK (2001)

Chapter 20
Optimal Design of a Common Rail Diesel Engine Piston

Teresa Donateo

Abstract. This chapter analyzes the main challenges in the application of "simulation optimization" to the design of engine components, with particular reference to the combustion chamber of a Direct Injection Diesel engine evaluated via Computational Fluid Dynamic (CFD) codes.

20.1 Presentation

This chapter analyzes the main challenges in the application of "simulation optimization" to the design of engine components, with particular reference to the combustion chamber of a Direct Injection Diesel engine evaluated via Computational Fluid Dynamic (CFD) codes.

The chapter starts with a description of the advantages of simulation optimization with respect to traditional trial and error approaches. The state of the art and the recent spreading of such techniques also in industry will be considered. Then, the specific challenges of optimizing an internal combustion engine are analyzed: the large computational time required for the fluid dynamic simulation of the engine (depending on the resolution of the computational mesh used to represent the fluid domain), the necessity to take into account several operating conditions (each requiring time expensive simulations), the interaction between fluid and solid structure (requiring the combination of CFD and FEM), etc. Moreover, if the design parameters include the geometrical features of the engine (i.e. the shape of the combustion chamber), the computational three-dimensional mesh has to be parameterized so that it can be automatically generated according to the selected values of the design parameters. This is particularly challenging when using commercial CFD codes that usually have a specific pre-processor with specific requirements in terms of grid quality, volumes connection, boundary conditions, etc. Constraints, restrictions and

Teresa Donateo
Department of Engineering for Innovation, via per Arnesano, 73100 Lecce, Italy
e-mail: teresa.donateo@unisalento.it

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 513–541.
springerlink.com

limits that the designer must meet due to norms, regulations and functionalities are additional challenges in the design process. Another aspect to be considered is the multi-objective nature of the problem (the main goals to be achieved are the reduction of emissions, the containment of fuel consumption and CO2 emissions, etc) that has been addressed in different ways by the main research centers involved in this kind of application.

Among the available optimization methods, Genetic Algorithms (GAs) are usually chosen in this application for their robustness and their capability to deal with multi-objective optimizations. Furthermore, they are simple to use and to combine with existing simulation code without significant modifications and their efficiency is independent of the nature of the problem. Another advantage of genetic algorithms is their implicit parallel nature that makes possible to easily exploit the computational capability of high performance multi-processors server now available not only in academic but also in industrial research centres. The advantages and the criticism of distributing the computational load on inter regional computing grids like the south Italy SPACI Grid will be also underlined.

Finally, a test case will be presented, describing the application of simulation optimization to the design of a common rail direct injection diesel engine for automotive applications based on multi-objective genetic algorithms and CFD analysis with respect to four different engine operating conditions. Note that a multi-disciplinary approach is a key aspect to successfully apply the method. The achievements shown in the test case are the results of the collaboration with both automotive companies (CVIT - Bosch Research Center, Bari, Italy, Prototipo Group) and high performance computation academic research centres (HPC- Lecce). Moreover, the methodology described in the chapter has been extensively applied for the design of commercial diesel engines since 2003 so that a wide amount of data are available. The results have been also validated through experiments by building and testing the optimized pistons developed with the described method.

20.2 Introduction

In direct injection diesel engines, the combustion chamber is represented by the space defined, at each time, by the cylinder head and walls and the piston crown. Since the combustion and emissions mechanisms of formation are strongly affected by the flow field produced by the chamber shape, the optimization of the bowl profile is a strategic way to fulfill present day and future regulations about pollutant emissions and greenhouse gases, which depend on fuel consumption. A symmetrical cavity, called the bowl, is usually present in the piston to allow fuel to be injected, mixed with air and burned. This type of combustion system was firstly introduced in 1934 by a Swiss company named Adolph Saurer [1] and then adopted by several companies, such as Scania, Volvo, PSA, British Leyland, IVECO and many others. Lately, the increase of fuel injection pressure and the use of multiple injections improved their efficiency and reduced emissions. These results yielded the development of high-pressure electronically-controlled injection systems as unit injectors

and common rail systems. Due to the high flexibility of common rail in fulfilling arbitrary injection strategies, cars equipped with direct injection diesel engines and Common Rail injection systems are now widespread in the automotive market.

The necessity to fulfill more and more stringent limits of emissions, lead designers to abandon the traditional trial and error approach in the design of the combustion chamber and to search for innovative solutions. Since '90s, work has been done, mainly through experiments but also with the help of CFD simulations to study the effect of combustion chamber on engine performances and emissions by comparing a small number (two or three) of combustion chamber designs a ([2],[3], [4], [5], [6], [7], [8],[7]).

In the mean time, the academic world and in particular the ERC Research Center at the University of Madison-Wisconsin [9] and the CREA Research center of the University of Salento [10], introduced the use of simulation optimization in the design of engine combustion chambers. In these first applications, the optimization was performed by combining a standard genetic algorithm with the Kiva code, an open source three-dimensional engine simulation program. These works underlined one of the difficulties of optimizing the combustion chamber, that is the necessity to take into account a very large number of input parameters. The two main operating parameters of the engine are its torque and speed that change continuously when the vehicle is run according to the vehicle route, the user driving style, etc. For this reason, to assess the performance of a vehicle in terms of consumption and emissions, standard driving cycle are used. A driving cycle consists of a particular profile of velocity versus time that has to be executed under controlled conditions. Only the vehicles that fulfill specific regulation in terms of emissions along a standard driving cycle can be introduced in the market. The specification of the driving cycle changes form nation to nation and depends on the mass or the rated power of the vehicle.

In the preliminary design of the vehicle, however, the engine is not tested over the entire driving cycle but only over a certain number of operating conditions (torque and speed couples) named "modes" that are assumed to be representative of the cycle. The effect of a particular chamber shape over emission levels changes significantly when changing from one mode to another [6]. Thus, the combustion chamber optimization has to be performed according to several modes. Senecal et al. [11] applied the KIVA-GA optimization method to optimize chamber for two operating modes, de Risi et al. [12] considered up to four modes. Moreover, the performance of an engine with a particular combustion chamber shape depends on the control strategy chosen for the engine in terms of injection strategy (injection pressure, number of injection per cycle, quantity of fuel injected in each injection pulse), EGR (exhaust gas recirculation), boost pressure, etc. This problem is usually approached by separating the design parameters that are "common" across all modes (compression ratio, combustion chamber profile, etc.) from those that can vary from mode to mode (EGR, injection profile, boost pressure, etc.) and so can be considered "independent". To completely optimize engine performance and pollutant emissions, both "common" and "independent" parameters should be considered in the optimization. This approach, used by Reitz et al. [9], would find an absolutely optimized configuration of the input parameters but it is unlikely to explain why this

configuration works well. Moreover, by using this approach, the complexity of the system strongly increases when different modes are taken into account since each independent parameters has to be counted as many time as the number of modes. For this reason, the optimizations of engine geometry and control parameters were taken separated at the CREA. De Risi et al. [13] developed a two-step optimization methodology where both steps are based on genetic algorithms and CFD simulations. Firstly, the combustion chamber shape is optimized for a fixed injection strategy. Then, the shape of the combustion chamber is kept constant and the injection strategy is changed to identify the response of the chambers selected in the first step to different injection strategies.

In [14] to save computational time the Kriging response surface model is adopted to limit the number of CFD simulations to be performed. The Kriging method is used to develop an approximation model that is coupled with an optimization method to find the optimum. In this way, the computational time is cut by 95%. However, using RSM in optimization the global optimum can be missed because the estimated function values obtained with RSM includes errors at an unknown point.

Nowadays, the use of multi-objective optimization combined with CFD code is widespread in automotive industry that has become one of the main user of complex simulation and optimization tools. However, industry search for user-friendly optimization tools easily to couple with existing commercial CFD code. This is the reason of the success of optimization environment like ModeFRONTIER [15] and iSIGHT [16]. In these optimization environments, different optimization method (genetic algorithm, simulated annealing, etc.) can be chosen by the user. However, it is important to stress the difference between optimizing the engine in terms of operating conditions (injection strategy, EGR level, etc) and in terms of combustion chamber shape. In the first case, in fact, the design parameters are usually easy to modify since they are given as input to commercial CFD code by means of ASCII files. On the other hand, the shape of the combustion chamber is given through a complex three-dimension computational mesh that cannot be easily described by input parameters and automatically generated. For this reason the next chapter will analyze the automatic generator of mesh for open source and commercial CFD codes.

20.3 Automatic Definition of the Computational Mesh

In direct injection diesel engines, the combustion chamber consists of two communicating regions named squish and bowl and illustrated in figure 20.1. During the intake and exhaust phases, the combustion chamber communicates with the manifolds through the intake and exhaust valves. However, the flow field generated by the intake process is not affected by the shape of the combustion chamber [17] and the exhaust phase is important only for its effect on catalyst and turbo-charging systems. Thus, only the closed portion of the engine cycle is considered.

The squish region is the cylinder space defined at any time by the cylinder head, the cylinder wall and the piston head while the bowl is the cavity usually present

in the piston to allow fuel to be injected, mixed with air and burned. Note that the squish region grows and shrinks as the piston moves downward and upward between the top dead center (TDC) and the bottom dead center (BTD) while the bowl region keeps always the same shape and size. The minimum height of the squish region is dependent on the compression ratio of the engine and is named ”squish height”. To simulate the behavior of the combustion chamber with CFD code, a moving 3D computational mesh that describes the combustion space according with the piston movement is used.

In initial applications of simulation optimization to the combustion chamber, the engine was simulated with different versions of an open source CFD code named KIVA ([18],[19]). In the first investigation at the ERC [9] the bowl geometry was defined by three input variables (bowl diameter, bowl depth and central crown height of the piston) allowing only open chamber profiles to be investigated. Lately, a more general chamber profile generation tool was developed at the ERC[20]. where the chamber profile is defined according to the parameters shown in figure 20.2. This is performed with an automated grid generator named Kwickgrid that uses a reduced set of input parameters when compared with the standard KIVA grid generation code [18]. Kwickgrid uses up to five parameters to define the overall piston bowl shape and up to eight variables to generate Bezier curves that describe the desired piston

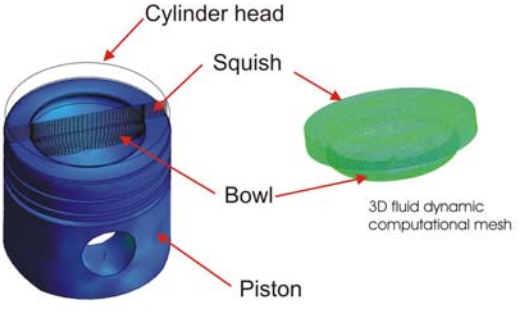


Fig. 20.1 An example of computational mesh

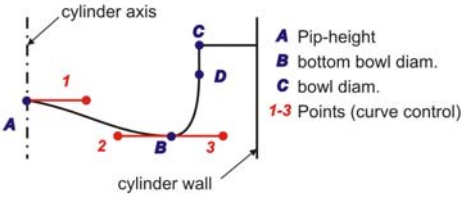


Fig. 20.2 The ERC profile generator, adapted from [20]

design and make it smooth for practical application. An approximating and iterative methodology is used to maintain the compression ratio and user-determined mesh size until a convergence criterion is reached.

In 2002, Senecal et al. [11] presented a general geometry parametrization where six parameters were used to define the bowl shape. By increasing or decreasing the number of parameters defining the profile it is possible to include more or less "wiggles" in the bowl profile. Once the bowl profile is determined, a grid generation program G-Smooth is used to create the mesh. A wide range of bowl shapes can be obtained with this technique but some of them are unsuitable for practical application. Senecal et al. [11] stress the importance of keeping the same mesh resolution for all geometries generated in the search. This ensures that differences in design performance are due to changes in geometry (and other eventual independent parameters) and not changes in mesh resolution. Like in the approach of the ERC center, bore, stroke, and compression ratio are constant. The compression ratio set by the user is maintained for all designs by changing the squish height.

In the investigation of Wakisaka et al. [21], the shape of the combustion chamber is defined with 10 design variable (see figure 20.3). Injection angle is also considered as design variable because diesel engine combustion largely depends on the injection angle.

In the investigations performed at the CREA the parametric schematization of figure 20.5 has been initially considered. Note that this schematization does not allow more than one inflexion point in the bowl profile. In the method of de Risi et al. [10] the volume of the bowl obtained with a particular combination of the parameters is calculated as algebraic sum of six volumes V_j with $j = 1,6$ as in figure 20.4. Once calculated the six volumes the volume of the bowl is given by:

$$V_{bowl} = \sum_{i=1}^{5} V_j - V_6 \tag{20.1}$$

Since volumes V_j with $j = 2,6$ depend on the position of point O, a non linear equation in x_0 is obtained that is solved with a standard iterative calculus procedure. Note

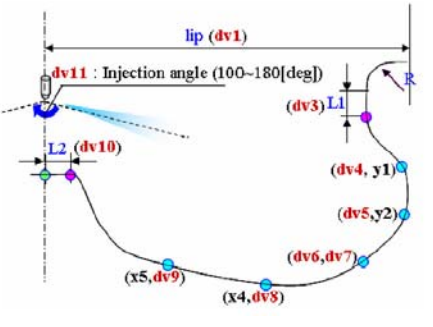


Fig. 20.3 The Wakisaka's profile generator, adapted from [21]

that in this way the achievement of the desiderate volume bowl (i.e. compression ratio) is obtained by moving point 0 in the horizontal direction and not by changing the squish height. In this way the bowl to squish ratio, that strongly affects the flow field, doesn't change when the parameters of the bowl are changed. The bowl profile obtained is then processed by a tool named "Meshmaker" that automatically write the simple input file which is required by the KIVA standard preprocessor, named K3prep, in order to generate a three-block structured mesh. The mesh consists of three blocks represented in 20.5. The first two blocks define the squish region while the third block describes the bowl. The spatial resolution is set equal for all chambers and the number of divisions along x, θ and z axes is automatically calculated for each chamber according to both engine size (bore, stroke and squish) and bowl depth and width. K3prep automatically adapts the shape of the computational cells of the third region to follow the profile of the bowl trying to avoid cells with a bad aspect ratio. If this is not achievable, depending on the shape of the profile, an error message is given by k3prep. In this case the computational mesh is not generated.

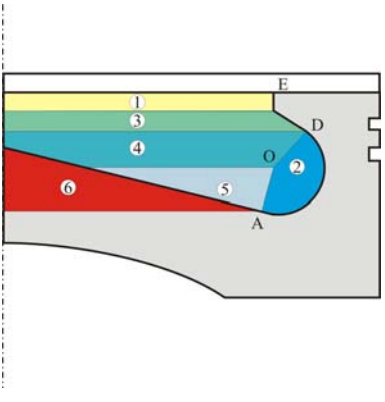


Fig. 20.4 The CREA bowl volume calculation

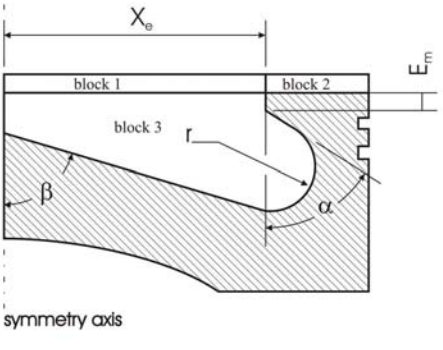


Fig. 20.5 The CREA profile generator

All the examples illustrated until now, refer to open source mesh generators where structured meshes are used and the implementation of the automatic generation of the chamber profile is quite simple. On the contrary, the implementation becomes more complex when using commercial CFD codes that usually consider unstructured meshes and have complex pre-processors with specific requirements in terms of grid quality, volumes connection, boundary conditions, etc. Moreover, they only accept input files with specific formats (Iges, Patran, STL, etc.). Thus, automatically writing the chamber profile for them is quite challenging. Recently, the CREA research center extended the optimization method to the use of commercial CFD codes in collaboration with the Nardo' Technical Center, Prototipo group [15] . In this application, the simulation of the engine was performed with the CFX software and the procedure of figure 20.6 was developed to generate an automatic computational mesh for this software. Moreover, the effect of the combustion chamber was evaluated not only with respect to its fluid-dynamic behavior but also with respect to the thermal and pressure stresses on the piston head. This could be done thanks to the capability of the CFX software in performing both CFD analysis of the fluid domain (combustion chamber) and FEM analysis of the piston (solid domain). The flow-chart of the optimization process is shown in figure 20.6.

The method has been implemented in the optimization environment ModeFrontier. The input files *Meshparam.txt* and *Enginemode.ccl* contain the geometrical parameters of the bowl and the operating conditions of the engine, respectively. These files are automatically generated in the optimization environment according to the design parameters selected by the optimization algorithm. The geometrical parameters defining the bowl are the input of the automatic profile and mesh generator ($Meshmaker - X$). The engine operating conditions (rpm, injected fuel, injection strategy, EGR, etc.) are used as boundary conditions for both the thermo-fluid dynamics analysis and the structural simulation with CFX. The results of the simulations are post-processed with CFX-Post and the main outputs are written in two

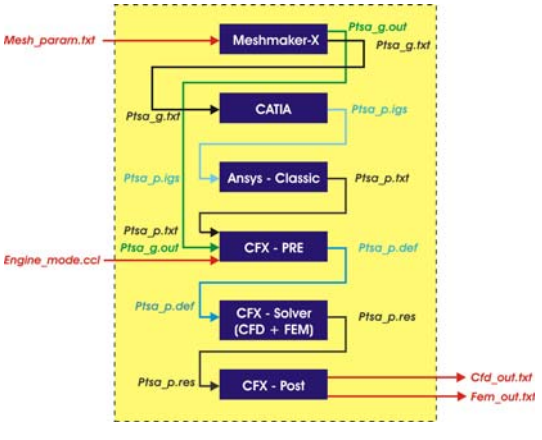


Fig. 20.6 Flow chart of the optimization with CFX

ASCII files that are returned to the optimization environment. The first one is named *Cfdout.txt* and contains the emission levels (mainly soot and NOx) and the fuel consumption of the engine with the proposed combustion chamber for the operating mode in *Enginemode.ccl*. The second one (*femout.txt*) contains information about the capability of the piston to sustain the predicted thermal load that will be used as constrain in the optimization process.

The bowl profile is obtained by the union of the two cubic Bezier curves (AB and BD) and the vertical lines OA and CD as illustrated in figure 20.7. Points 1,2,3,4 are control points that define the direction and the slope of the curves. By moving points A,B,C , D,1,2,3 and 4, a large variety of combustion chamber profiles can be obtained. However there are some constraints in the building of the profile. Point D can move only in the horizontal direction (x axis), $z_D = 0$, and its x coordinate must be the same of point C ($x_C = x_D$) while point A has to belong to the vertical axis z ($x_A = 0$). The two Bezier curves have in common the point B, thus the slope in that point has to be the same for the two curves. The achievement of the constant volume of the chamber is obtained by adjusting the coordinates of point B. In particular point B is moved parallel to the bisector of angle the defined by the prolongations of segments A1 and C4 until the volume of the chamber is equal (with a tolerance Δ_v) to the value chosen by the user. The numerical values of the parameters of figure 20.7 as described in table 20.1 are contained in the input file *Meshparam.txt*.

Once the profile has been generated, an unstructured computational mesh is defined for the fluid domain according to the resolution selected by the user (also included in the *Meshparam.txt*) and written in a PATRAN file to be read by the CFX solver. *Meshmaker* $-X$ also sets the boundary conditions on the mesh and check for the fulfillment of the CFX preprocessor requirements in terms of grid uniformity, cell shape, etc. Thanks to the combined CFD-FEM evaluation, chambers that are suitable from the CFD point of view but not able to sustain the thermal load are penalized in the optimization process. Details on the construction of the solid domain for the FEM analysis are not reported here for the sake of brevity. Note that the

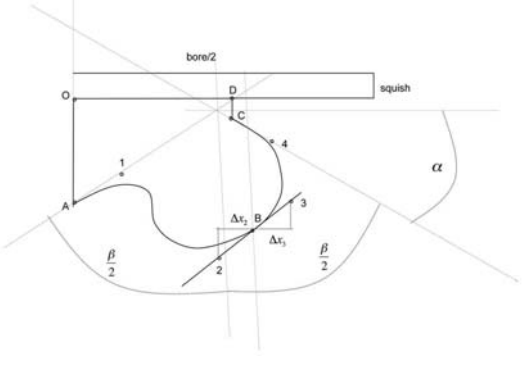


Fig. 20.7 Chamber parameterization for CFX

Table 20.1 Input parameters for Meshmaker-X

Variable	Description
z_A	z coordinate of point A
x_1	x coordinate of point 1
z_1	z coordinate of point 1
Δx_2	horizontal distance between points 2 and B
Δx_3	horizontal distance between points 3 and B
$s_{2,3}$	slope of the line 2 3 B
x_4	x coordinate of point 4
z_4	z coordinate of point 4
x_C	x coordinate of point C
z_C	z coordinate of point C
x_B	initial value of the x coordinate of point B
Δx_B	movement of point B along x direction
n_1	number of points defining the first Bezier curve (AB)
n_2	number of points defining the second Bezier curve (BC)
vol_f	selected volume of the bowl
Δ_v	maximum acceptable error of the bowl volume

computational time required for the FEM analysis of the piston is negligible with respect to the CFD simulation of the combustion chamber.

20.4 Single-objective and Multi-objective Approaches

The optimization of an internal combustion engine is a very difficult task due to the multiple and competitive goals to be achieved. A typical case is the contemporary reduction of the two main diesel engine emissions (soot and nitrogen oxides). Other engine outputs to be controlled in the optimization are the emission of unburned hydrocarbons (HC), specific fuel consumption (ISFC), mean effective pressure, exhaust temperature, peak of pressure, noise, etc. The initial approach of the ERC group was to combine all the engine optimization goals in a merit function. For example, the form of the merit function utilized in [22] is as follows:

$$merit = \frac{1000}{\left[\frac{NO_x+NC}{NO_{x,n}+HC_m}\right]^2 + \frac{ISFC}{ISFC_0} + \frac{WHEAT}{WHEAT_0}} \tag{20.2}$$

where $NO_x + HC$ are the nitrogen oxides and hydrocarbon emissions lumped together while $NO_{x,m} + HC_m$ are the target values to be achieved to fulfill specific engine emission regulation, $ISFC$ is the indicated specific fuel consumption and $ISFC_0$ is the same but calculated from simulation of the baseline operating case. $WHEAT$ is the total cylinder wall heat transfer and $WHEAT_0$ is the corresponding value for the baseline configuration.

A similar approach was also used by Senecal et al [11] to optimize the chamber with respect to two operating modes A and B, Senecal calculated the merit function with respect of each mode (f_A and f_B) and then considered the following expression:

$$f = \frac{1}{0.5\left(\frac{1}{f_A} + \frac{1}{f_B}\right)} \tag{20.3}$$

Compared with the simply averaging of the individual merit function values, this expression has the property of weighting the overall merit value more heavily by the lower of f_A and f_B. In this way, the formulation does not allow a design with a high value of f f_A and a very low value of f_B to falsely have a high value of the overall merit function. The individual merit function values from both operating conditions must be reasonable to obtain a high value of f.

Starting from 2008, the ERC group [23] considered a multi-objective approach by using the NSGA-II optimization algorithm. In [14] the EGOMOP (Efficient Global Optimization for multi-objective problem) approach is used, in which each objective function is converted into its Expected Improvement and this value is used as fitness in multi-objective optimization problem. The k-means clustering method is adopted for the efficient selection of additional sample points in high dimensional problem. In this way not all the Pareto points are used as additional sample points but only a reduced number is selected. In this study, the optimization was performed with four objective functions: soot, NO, CO and thermal efficiency.

In the approach followed at the CREA, the optimization goals, soot, NOx, HC and Indicate Mean Effective Pressure, are kept separate and a fitness component is evaluated for each of them:

$$F_1 = \sum_{i=1}^{N_{modes}} F_{1p}(i) \cdot w_1(i) \left(\frac{NO_x^0}{NO_x}\right)_i \tag{20.4}$$

$$F_2 = \sum_{i=1}^{N_{modes}} F_{2p}(i) \cdot w_2(i) \left(\frac{soot_0}{soot}\right)_i \tag{20.5}$$

$$F_3 = \sum_{i=1}^{N_{modes}} F_{3p}(i) \cdot w_3(i) \left(\frac{HC^0}{HC}\right)_i \tag{20.6}$$

$$F_4 = \sum_{i=1}^{N_{modes}} F_{4p}(i) \cdot w_4(i) \left(\frac{IMEP}{IMEP^0}\right)_i \tag{20.7}$$

Note that in this case the optimization of the combustion chamber is treated as a maximization problem. In these equations, subscript 0 refers to the values obtained with the baseline configuration while $w_j(i), j = 1,4$ represents the weight of mode i for the definition of objective j (1=NOx, 2=soot, 3=HC and 4=IMEP); N_{modes} is the total number of modes considered in the application and $F_j^p(i), j = 1,2,3$ is the value of the penalty function for the fitness component j calculated on mode i. Penalty functions are used to introduce inequality constraints in the optimization.

This approach also allows for secondary optimization goals to be taken into account as constraints instead of goal so that the complexity of the problem is not excessively increased by considering more than four fitness components. More details on this approach will be given in the last section of this chapter that describes a case study performed with this approach.

20.5 Optimization Algorithms

Genetic algorithms (GAs) are suitable for engine optimization thanks to their high robustness and their capability to deal with multi-objective optimization. Moreover, they are simple to use and to combine with existing simulation code without significant modifications. The implicit parallel nature of GAs makes it easy to exploit the growing parallel computing power. In fact, they work with a population of solutions, then multiple optimal individuals can be captured in a single run.

Before GAs, traditional optimization methodologies (e.g. gradient based methods) were used as optimization tools in engineering design. These methods are sensitive to derivatives of the parameters space (while the design spaces is generally non-smooth in engineering applications) and have difficulties both in finding global optimum and in addressing multi-objective problems.

A considerable number of Multi-objective Genetic Algorithm applied to industrial problems can be found in literature. The assessment of MOGAs toward the application of engine designs by Shi and Reitz [23] showed that NSGA II, compared with a micro-GA and an Adaptative Range Multi-Objective Genetic Algorithm performs better with respect to optimality and diversity of the optimal solutions when a large population size is applied. As far as the activity at CREA is concerned a multi-objective genetic algorithm named GA-CREA was developed. In the GA-CREA algorithm [10], a standard genetic algorithms structure is considered but instead of a single fitness function, an array of fitness values is assigned to each configuration as described in the previous paragraph. The components of the array are the fitness values calculated separately according to each objective for each individual in the current generation. Once the fitness arrays are calculated, non dominated solutions are ranked by using the dominance criterion. The method is illustrated in the *test case* section since it has also been implemented in the HiPeGEO algorithm.

At each GACREA iteration (i), before the new population (with size Np) is generated, the elitism module randomly select Ne individual from the non dominated solutions obtained in the generation (i-1), Pareto(i-1) and reproduce them, without any modifications, in the population pop(i). The remaining Np-Ne individuals of gen(i) are obtained by gen(i-1) after applying single-point crossover and mutation. Ne is set equal to the size of Pareto(i-1) if it is less than Np/2, otherwise it is set equal to Np/2. The execution of the optimization is stopped when a prefixed number of generations is completed and the Pareto front is extracted from all the individuals analyzed during the GA run.

The performance of GA-CREA were evaluated with respect to both mathematical test problems [24] and a typical engine optimization problem [25]: reducing soot

and NOx emissions and increasing IMEP by changing the injection pressure and the advance of the single injection. This problem is called ”Diesel dilemma” since retarding the injection NOx decrease while soot increases. On the other hand, by increasing injection pressure soot emissions and IMEP improve but NOx emissions get worse. GA-CREA was compared with three optimization tools available in the ModeFrontier environment: MOGAII, MOSA and MACK.

For the quantitative analysis of the optimization algorithms, four metrics available in literature [26] were considered and applied to the Pareto front obtained with GACREA, MOGAII, MOSA and MACK. The results of this test showed that the optimization algorithms GACREA performs very well in terms of distribution and definition of the Pareto front. Even if the number of points on the Pareto front is lower than in the case of MOGAII and MOSA, only a small percentage of these points are dominated by the solutions found with MOGAII, MOSA and MACK.

Another genetic algorithm named HiPeGEO (High Performance Genetic Algorithm for Engine Optimization) was developed by the CREA in collaboration with the High Performance Computing group at the University of Salento,. The HiPeGEO algorithm will be described in the test case section since it was specifically developed for that application.

20.5.1 Implementations on Computation Platform and Grids

Thanks to the implicit parallel nature of genetic algorithms, a UNIX shell script is usually sufficient to parallelize a GA and distribute the computational task on multiprocessor servers. Recently, some works have been devoted to analyze the use of grid computing for industrial optimization problems [27]. Based on the popularity of the Internet and the availability of a large amount of geographically disperse computational resources, the grid computing allows the use of the computing power offered by such platforms to tackle complex problems which involve intensive computing tasks. An implementation of the micro-GA HiPeGEO using a grid system to distribute the function evaluations is presented in [12] and applied to the optimization of the combustion chamber according to the approach developed at the CREA. Grid technologies were implemented in a grid portal named DESGrid which consists of three essential modules:

- a web interface to access the system transparently;
- a second module for the management and the execution of the HiPeGEO;
- a grid resource manager to optimize the use of the available computational resources.

As further example concerning heuristic methods for multi-objective optimization and grid computing is described in the work of [27]. In this investigation, authors do not just distribute the function evaluations (e.g., the algorithm remains as in sequential) but develop a new grid-based model of search based on the PAES algorithm.

However, some drawbacks of the use of computational grid for engine optimization via numerical simulation have to be underlined. Firstly, the numerical results of the simulation are affected by the architecture where they are run over. This means

that difference in the performance of the analyzed chambers could derive not only from the chamber specification but also on the architecture used to simulate that specific chamber. A possible way to solve this problem is to use scaled objective function as in 20.4 - 20.7 where the baseline data used for the comparison are calculated on the same architecture where the current design is being evaluated. A second difficulty arise when commercial codes are used. In this case, the possibility to distribute the computational load over the grid is limited by the number of available licenses. Thus, each resource of the grid has to be connected with the licence manager server and ask for the availability of a license to run an evaluation with the simulation code.

20.6 Test Case

A direct injection diesel engine with four valves per cylinder, a zero-offset bowl and a centrally disposed seven-hole injector has been considered. The engine specifications are reported in table 20.2. To optimize the bowl profile, the simple parametric schematization of figure 20.5 was considered. The range of variation of the parameters for the test case are shown in table 20.3.

Table 20.2 Engine specification

specification	unit	value
Displacement	$[cm^3]$	420
Compression ratio		17.2
Intake valve closing, crank angle BTDC	$[deg]$	134
Injection system		Common Rail
Holes diameter	$[mm]$	0.145
Number of holes		7

Table 20.3 Input parameters for the test case

Parameter	unit	lower limit	upper limit
X_e	$[mm]$	15.0	34.0
α	[deg]	-90	90
β	$[deg]$	45	90
r	$[mm]$	2.0	14.0
E	$[mm]$	1.0	5.0

For the present application, bore, stroke, squish volume and compression ratio were kept constant and equal to the baseline configuration chosen for reference. Therefore, the bowl volume was the same for all the analyzed combustion chambers.

The simulation of the engine behavior when changing the bowl shape was performed with the CFD code KIVA3V-CREA while the optimization was performed with the HiPeGEO algorithm, specifically developed for this application.

20.6.1 Problem Specifications

The injection parameters were assumed to be the same for all investigated configurations and an innovative injection strategy with a large quantity of fuel injected 60 before TDC was considered (early injection) at low speed and load while a single injection was assumed for mode 3 and 4. The optimization was aimed at reducing pollutant emissions at low speed and load and increasing engine performance at high load. Thus, three fitness functions were defined according to the main pollution produced by diesel engines: particulate, NOx and Unburned Hydrocarbons (HC) while a fourth function was used to take into account the Indicated Mean Pressure (IMEP). The behavior of each configuration was evaluated by comparing its emission levels and IMEP value with those produced by the baseline chamber for the same operating mode. The comparison was performed for all modes and a weight was assigned to each mode on the basis of its influence on the definition of engine emissions and performance. The four objective functions have been already reported in equations 20.4 - 20.7.

The four modes of table 20.4 were used in the optimization. The same weight was assigned to mode 1 and 2 in the calculation of the fitness components because they were considered equally important in the definition of total engine emissions. In the same way, a weight equal to 0.5 was assigned to mode 3 and 4 for IMEP maximization. Note that mode 4 represents the maximum allowed values of speed and load.

Table 20.4 Operating conditions for the optimization

Mode	1	2	3	4
Engine speed $[rpm]$	1500	2000	3000	5300
Indicated Mean Pressure IMEP$[bar]$	4.3	8.0	25.0	20.5
Weight for emission calculation (w1=w2=w3)	0.5	0.5	0	0
Weight for performance calculation (w4)	0	0	0.5	0.5

The optimization process was performed by means of numerical simulations; thus, the injected fuel mass was necessarily an input parameter to be set the same for all configurations. When engine speed and injected mass are kept constant, the highest IMEP values correspond to the lowest fuel consumptions so that the capability of each solution in reducing specific consumption can be evaluated by analyzing the IMEP levels.

The IMEP was considered as the main objective at full load but it has to be taken into account at low load and speed too. In fact, IMEP values can be very low, fuel injected being the same, if the completeness of the combustion processes is prevented somehow or other. In spite of their low performance, these chambers could be considered good solutions by GA because they produce very low levels of NOx. For this reason, a penalty function, was used to penalize chamber configurations with low IMEP values at low speed and load. If the current chamber gives an IMEP value higher than the baseline configuration, the penalty function is set equal to 1 and no penalization is given to the chamber. Otherwise, the chamber is slightly penalized if the reduction of IMEP is inferior to 8% while penalization is much higher when the reduction is greater than 8% with respect to the baseline case. The same criterion was applied at full load to penalize chamber configurations with soot emissions higher than a prefixed threshold value, in the present investigation the value $Soot_{ths} = 0.78 g/kgf$ was considered (baseline value at operating mode 4). Mathematically, the penalty functions used in this test case can be both described in the following way:

$$F_p(i) = \begin{cases} 1 & p \geq 1.0 \\ 1.0 + log(p)/10p & 0.92 \leq p \leq 1.0 \\ p + 0.07 & p \leq 0.92 \end{cases} \tag{20.8}$$

where p is the penalty parameter, i.e. $p = IMEP/IMEP_0$ at modes 1 and 2 (applied to fitness components 1 to 3) and $p = soot_{ths}/soot$ at modes 3 and 4 (applied to the fourth fitness component). The choice of this formulation derives from experience in the automotive industry according to which variations up to 8% can be regained by re-mapping the engine.

20.6.2 Engine Simulation Code

KIVA3V is a transient, three-dimensional, multiphase, multicomponent code for the analysis of chemically reacting flows with sprays. The code uses discretizes space using the finite-volume technique and uses an implicit time-advancement with the exception of the advective terms that are cast in an explicit but second-order monotonicity-preserving manner. Arbitrary numbers of species and chemical reactions are allowed. A stochastic particle method is used to calculate evaporating liquid sprays, including the effects of droplet collisions and aerodynamic breakups. The code is specifically designed for performing internal combustion engine calculations. However, in this investigation a modified version of the KIVA3V code named KIVA3V-CREA [28] was used since improved models for spray and combustion are needed to predict the behavior of modern direct injection diesel engines when geometrical and control parameters like injection strategy and EGR are changed.

20.6.3 HIPEGEO

In the HiPeGEO algorithm, the micro-GA technique proposed by Coello's and Pulido's [29] is implemented. HiPeGEO automatically executes the preprocessor, the solver and the postprocessor of KIVA3V-CREA, as described in the text subparagraphs. HiPeGEO has been developed specifically for this application and uses Grid Technologies to reduce computational time. The complexity of the system is hidden to the final user who accesses the Grid using a user-friendly web interface to solve a specific optimization problem. The flowchart of HiPeGEO is shown in fig. 20.8.

HiPeGEO is structured on two levels: an external level, where a certain number of macro-iterations are performed, and an internal one which is represented by a micro-GA cycle. The external iteration uses a large population divided in a replaceable and a non-replaceable portion (which is generated randomly only once at the begin of

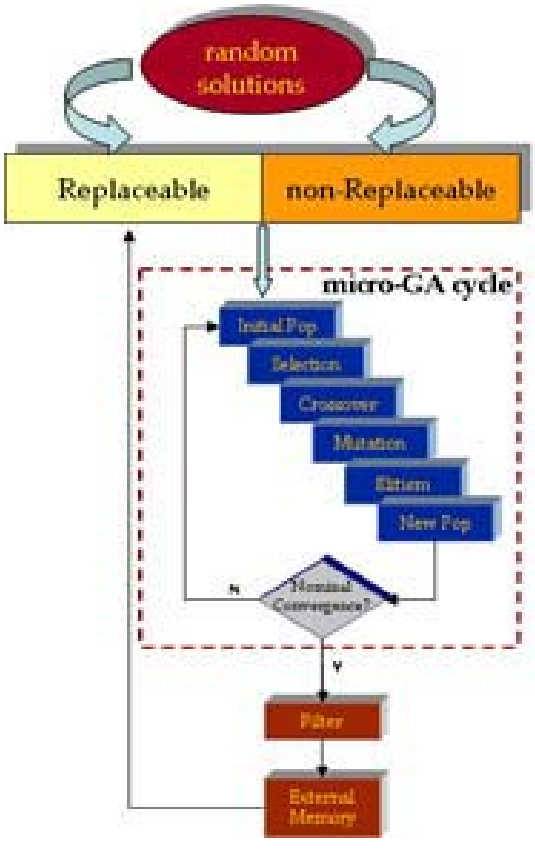


Fig. 20.8 HiPeGEO architecture

the optimization). The non replaceable portion is never upgraded and represents the source of diversity for the micro-GA cycles.

At each iteration, a micro population is randomly extracted from both the replaceable and non-replaceable portions and a micro-GA cycle is performed until the nominal convergence is reached. Then, the external memory containing the Pareto front (initially empty) is upgraded (as described in the *elitism module* section) with the nominal solution of the micro-GA and a new iteration is started. The size of the external memory is limited (*filter* block of figure 20.8) to a threshold value selected by the user with the clustering procedure described in the *clustering module* section. The micro-GA cycle is perfomed through the generation, fitness, rank, crossover, mutation, nominal convergence and elitism modules described next.

20.6.3.1 Generation Module

The *generation module* produces a list of randomly generated chromosomes of five genes representing the geometric parameters of fig.20.5:

```
xe=0.361762   alfa=0.695257   beta=1.547302   r=0.361379   em=1.108602
xe=0.863116   alfa=4.800000   beta=0.695257   r=0.937501   em=0.361379
xe=0.361762   alfa=1.326748   beta=4.800000   r=0.361379   em=0.863116
xe=1.547302   alfa=4.800000   beta=4.800000   r=4.800000   em=0.408767
.....
```

For each gene, a suitability analysis is performed to verify that the corresponding geometrical parameter is included in the range of variation of table 20.3.

20.6.3.2 Fitness Module

The *fitness module* calculate the fitness values corresponding to a list of the chromosomes. For the test case, the fitnessmodule executes four runs with the KIVA3V code (one for each mode of table 20.4 and produces a list of chromosomes and their fitness values as follows:

```
xe=0.361762   alfa=0.695257   beta=1.547302   r=0.361379   em=1.108602
fit1=15.766697  fit2=0.001982   fit3=0.404997   fit4=0.078914

xe=0.863116   alfa=4.800000   beta=0.695257   r=0.937501   em=0.361379
fit1=7.673714   fit2=0.001166   fit3=0.106391   fit4=0.160870

xe=0.361762   alfa=1.326748   beta=4.800000   r=0.361379   em=0.863116
fit1=0.000000   fit2=0.000000   fit3=0.000000   fit4=0.000000

xe=1.547302   alfa=4.800000   beta=4.800000   r=4.800000   em=0.408767
fit1=18.446617  fit2=0.002777   fit3=0.000000   fit4=0.210047
.....
```

The fitness functions are calculated with the following procedure.

For each chromosome, the fitness module generates a file, named *iprep*, using the *meshmaker* tool described above. Then, the module calls the K3PREP grid generator that writes a mesh file for KIVA3V-CREA conforming to the needed specifications. The spatial resolution is set equal for all chambers and the number of divisions along x, and z axes is automatically calculated for each chamber according to both engine size (bore, stroke and squish) and bowl depth and width. In the present application

the maximum cell size was set equal to 2.2 mm, 1.4 mm and 2 deg after a sensitivity analysis performed to test the influence of grid resolution on KIVA3V results ([13]).

If the K3PREP pre-processor is able to generate a structured mesh with a good aspect ratio for the candidate chamber, an output file named *itape17*, containing geometric solution information, is passed to the KIVA3V application which simulates the combustion process and generates the fitness file, named *fort.65*. On the contrary, if a good mesh cannot be obtained, the corresponding chamber profile is excluded from the optimization process. Other input files are needed for KIVA3V execution, such as *itape5erc*, *itapei*, *itape5*, etc., containing initial conditions, constants of spray combustion models and so on.

At the end of each KIVA3V-CREA run, the emissions levels and the IMEP are stored and uses at the end of the four runs to calculate the penalty function and the fitness functions (eqs. 20.4- 20.7).

20.6.3.3 Rank Module

The *rank module* provides a list of rank values for the individuals belonging to a population to be ranked, as required by the *selection module*. To rank individuals, the approach developed by Fonseca [30] has been followed in HiPeGEO. In particular, the rank $r(j)$ of an individual j is defined by the number of fitness vectors by which $F(j)$ is dominated, increased by 1. If $F(x)$ is the fitness vector associated to solution x, $F(y)$ is the fitness vector of individual y, and the goal is the maximization of all the fitness components of F, than $F(y)$ is said to dominate $F(x)$ if the condition 20.9 is verified:

$$\forall i \quad (F_i(x) \leq F_i(y)) \quad \wedge \quad \exists i \quad (F(x_i) < F(y_i)) \tag{20.9}$$

If a vector is not dominated by any other, it is called non-dominated or not inferior and the corresponding design is said to belong to the Pareto front.

In this way the Pareto solutions, which are the "best" individuals in a multi-objective problem, have rank equal to one and while the "worst" solutions has a rank equal to the population size (). The pseudo-code of the rank module for the case study, characterized by 4 fitness functions to be maximized, is the following:

```
For i=1 to N
    Rank(i)=1
    For m=1 to N
        If ((F1(i)-F1(m)<=0) and (F2(i)-F2(m)<=0) and
           (F3(i)-F3(m)<=0) and (F4(i)-F4(m)<=0))
            If ((F1(i)-F1(m)<0) or (F2(i)-F2(m)<0) or
               (F3(i)-F3(m)<0) or (F4(i)-F4(m)<0))
                Individual i is dominated
                Rank(i)=Rank(i)+1
    next m
next i
```

20.6.3.4 Selection Module

The *selection module* provides a list of couples of selected chromosomes according to the rank selection method. Generally speaking, the rank selection method is preferable to the roulette wheel when there are big differences among the fitness

values of a population. In fact, with the rank method all the chromosomes have a chance to be selected, also those with all zeros values in the fitness because unable to generate a good quality mesh. Of course, the rank method is particular suitable for this application where the solutions are ranked in a multi-objective fashion.

20.6.3.5 Crossover and Mutation Module

The *crossover and mutation module* provides a list of offspring generated after crossover and mutation operations. Like the *generation module*, it verifies the suitability of offsprings. The selection of crossover and mutation techniques to be applied, is a crucial factor. For the present case study, the uniform crossover has been chosen since experience has showed its stability compared with other crossover methods and its efficiency if applied to chromosomes with few genes. The initial crossover probability P_{ci} is specified by the user (as a parameter of the HiPeGEO application). After the first iteration, the probability changes taking into account the degree of stability of the Pareto front and the number of iterations already executed. The crossover probability $P_c(i)$, referred to the i^{th} iteration is given by (20.10):

$$P_c(i) = \begin{cases} P_{ci} & i = 1 \\ P_{ci} - (a \times S_{pf}(i) + b \times i) & i > 1 \end{cases} \tag{20.10}$$

where a and b are two positive coefficients selected by the user and $S_{pf}(i)$ measures the stability of the Pareto front.

The distance between the fronts at i^{th} and $(i-1)^{th}$ iterations, $d(i)$, is defined as the average of Euclidean distance between couples of points with the same index j, belonging to the fronts (20.11):

$$\sum_{j=1}^{N(i)} \frac{\overline{P_j(i)P_j(i-1)}}{N(i)} \tag{20.11}$$

If $\Delta N(i)$ is the difference between the number of points belonging to the two fronts, and d is the optimal distance (fixed by the user) between two fronts in order to consider them very similar, then $S_{pf}(i)$ is given by eq. 20.12:

$$S_{pf}(i) = \begin{cases} 0 & \Delta N(i) \neq 0 \\ \left[1 - \frac{d(i)-d}{d(i)}\right] \times i & \Delta N(i) = 0, \quad d(i) \geq d \\ i & \Delta N(i) = 0, \quad d(i) < d \end{cases} \tag{20.12}$$

When the user specifies P_{ci} and a and b coefficients, the system verifies that $P_c(i)$ belongs to the $[0,1]$ interval $\forall i$, that is $(a+b) \leq \frac{P_{ci}}{i}\ \forall i$, so that $(a+b) \leq \frac{P_{ci}}{N_{max}}$, being N_{max} the maximum number of iterations to be executed.

The uniform mutation is then applied. It is known that the mutation probability has to be chosen as a compromise between the necessity to preserve the building blocks and the necessity to explore the search space for new building blocks. For this reason, a mutation probability increasing with the number of executed iterations $P_m(i)$ is considered (equation (20.13)):

$$P_m(i) = P_{mi} + c \times \frac{i}{N_{max}} \tag{20.13}$$

where P_{mi} is the initial mutation probability and c is a positive coefficient, both selected by the user. When the user specifies P_{mi} and c coefficient, the system verifies that $P_m(i)$ belongs to the $[0,1]$ interval $\forall i$, that is $c \leq (1 - P_{mi}) \times \frac{N_{max}}{i} \ \forall i$, so that $c \leq (1 - P_{mi})$, being N_{max} the maximum value of i.

20.6.3.6 Nominal Convergence Module

The *nominal convergence module* provides the nominal solution of the micro-GA if the convergence has been achieved.

In the present work, both convergence criteria suggested by Coello and Pulido have been implemented. The module verifies the similarity among the chromosomes belonging to the new population provided as input. For each geometric parameter (gene), a range of similarity is fixed by the user. The nominal convergence can be considered reached when the difference among the same genes is within the range of similarity for all analyzed individuals. Once the similarity criterion is satisfied, a representative individual is selected as nominal solution. However, if convergence is not reached after a fixed number of cycles, the micro-GA execution stops and the individuals are ranked to select the nominal solution. If two or more solutions have the same rank, one of them is randomly selected.

The nominal solution is then is copied in the external memory.

20.6.3.7 Elitism Module

The elitism module performs a tournament between the individual to be preserved and an individual randomly chosen from the population of destination. The winner of the match will remain in the population, while the loser will be excluded. This module is called at each micro-iteration of the micro-GA cycle to preserve the best individual (according to the rank method) . However, the same module is also used to perform two other elitism models suggested by Coello's and Pulido's and implemented at the end of the micro-GA cycle to drive the evolution of the macro-population:

1. the representative solution of the micro-GA cycle is preserved in the replaceable memory (is it wins the tournament);
2. at a fixed number of iterations some of the non-dominated solutions are used to update the replaceable portion.

20.6.3.8 Clustering Module

After each iteration, the *clustering module* verifies if the number of solutions stored into the external memory, which contains the Pareto front, exceeds the maximum size (n) defined by the user during the submission of the optimization process. When this happens, it is necessary to exclude some individuals, preserving a uniform

distribution of solutions. The procedure to perform the uniformity of the Pareto front is described in the following steps:

1. definition of a matrix D of size $m \times m$, where the generic element d_{ij} is the distance between i^{th} and j^{th} individuals;
2. search of the pair of points which have the minimum distance;
3. deleting of the point in the selected pair which has minimum distance from a third point;
4. if the number of excluded points is lower than $m - n$, go to step 2.

20.6.4 Required Computational Time

For the test case, the optimization was performed with the HiPeGEO parameters of table 20.5 on an AlphaServer where the computational time of a single KIVA3V simulation is equal to 35 minutes. The time required to run the optimization over an Alpha Server with 16 processors was 20 days, a value quite negligible with respect to the design time required with the standard trial and error approach (usually six months). The use of Itanium clusters of the SPACI (Southern Partnership for Advanced Computational Infrastructure) could allow reducing the constitutional load. On the Itanium server, where the computational time of each run is only 9.5 minutes, the optimization can be performed in 5 days. Moreover, the use of the 128-processors server could allow the number of modes included in the optimization to be increased up to 32, still keeping the same computational time (5 days).

Table 20.5 HiPeGEO parameters for the test case

Parameter	Value
Maximum Number of Iterations	100
Population Memory Size	50
Replaceable Portion (%)	70
Initial Population Size	5
Initial Mutation Probability	0.05
External Memory Size (%)	100

20.6.5 Analysis of the Results

The plots of figure 20.9 shows the final Pareto Front together with the values corresponding to the baseline chamber. Note that very small improvement on soot and HC were obtained with the HiPeGEO, as can be easily assessed by considering figure 20.9c) where the points representing the baseline configuration are very close to the origin of the plot. On the contrary, a strong reduction of NOx with respect to the baseline chamber can be obtained with almost all the optimized chambers. The improvement in IMEP are below 15% for all modes, as can be expected. Since

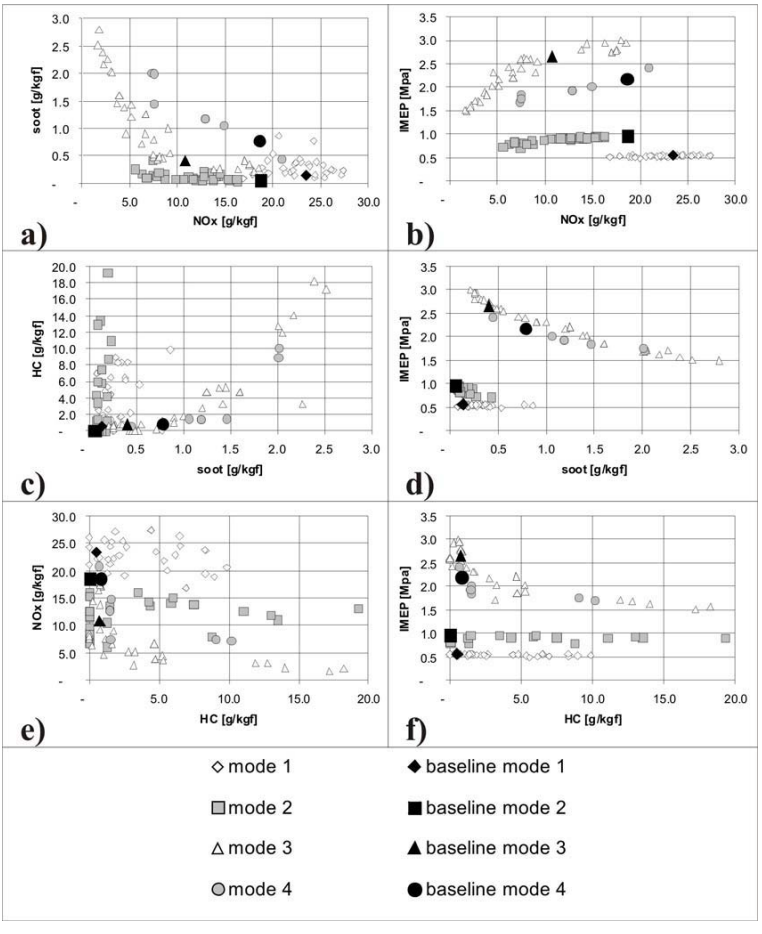


Fig. 20.9 Results of HiPeGEO

the injected mass is the same for all the chambers at the same operating mode, an increase of IMEP by 15% corresponds to reducing fuel consumption by the same amount. To better analyze the results, a clustering process has been performed on the Pareto front with the same algorithm described in the *nominal convergence module* section.

20.6.5.1 Clustering of the Results

The final Pareto Front has been clustered in five groups and the outcome of the clustering process is shown in fig. 20.10. The data in 20.10 can be used to analyze the effect of combustion chamber not only on each output parameter but also on each operating condition.

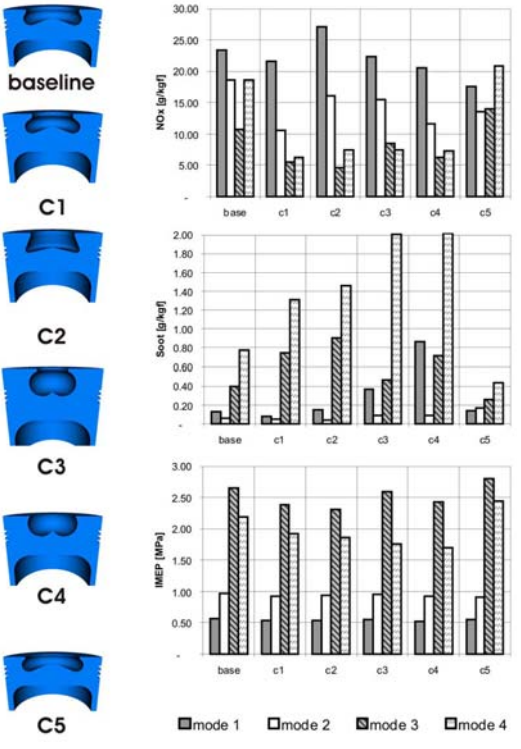


Fig. 20.10 Clustering of the Pareto chambers

Note that the first four combustion chambers (C1-C4), very deep and with a narrow throat, allow a strong reduction of NOx emissions on almost all the operating modes, while the fifth chamber is worse than the baseline chamber at high load (mode 3 and 4). On the other hand, soot emissions seem to be more affected by the mode. In fact, the behavior of the selected chambers changes with engine speed and load. In particular, chambers C1 and C2 (with a narrow throat and a large bottom) performs better than the baseline chamber at mode 1 and 2. At high load and speed (mode 3 and mode 4) a very large and shallow chamber is required to reduce particulate (see results of chamber C5). Chambers C3 and C4 are characterized by very high values of soot emission on all modes.

As far as IMEP is concerned, note that this output value is weakly influenced by the combustion chamber shape at mode 1 and 2. The best results are obtained with chamber C5 which is very similar to the baseline configuration, except for the shape of the throat lip and the high of the central protrusion. Among the five chambers of fig. 20.10, chambers C1 and C5 can be said to guarantee a better compromise in the optimization goals; for this reason they were selected for further investigation so as to explain the influence of bowl geometry on emissions and performance. The results of this investigation can be found in [12]. The choice between C1 and

C5 depends on the relative importance given by the user to the emissions in urban driving cycle (mode 1 and 2) with respect to high load conditions. In the first case chambers C1 is to be preferred while C5 is the best solution at high speed and load.

20.6.6 Conclusions

This chapter analyzes different approaches described in literature for the optimization of a direct injection diesel combustion chambers by means of evolutionary algorithms and Computational Fluid Dynamic (CFD) codes. The chapter focused principally on the investigations of two academic research centers: the ERC at the Madison-Wisconsin University and the CREA at the University of Salento, Italy. The different approaches of these centers were compared in terms of specification of the algorithm used for the optimization, kind of parameterization of the chamber profile, treatment of the competitive goals, etc. Finally, a test case developed at the CREA is presented to illustrate how these aspects have been practically addressed in the design of a common rail diesel engine piston in collaboration with a major European automotive company.

From the analysis of the test case presented here and from the application of the method to several other engine optimization problems, the following general considerations can be drawn.

The key factor in the optimization of a complex system like a diesel engine is, of course, the capability of the simulation model to capture the behavior of the engine. Even if this aspect is not directly connected with the optimization process, it must be always kept in mind when addressing the optimization of an engine. In particular, the optimization process must be always preceded by a validation of the model with appropriate experimental data.

Of course, a compromise must be looked for between accuracy and computational load. The behavior of an engine when changing its control parameter (like EGR, injection rate, etc.) can be captured with an acceptable accuracy also with simplified models like ANNs and 1-D simulation codes. On the other hand, when the goal of the optimization is to change the design parameter (spray orientation with respect to the cylinder axys, compression ratio, squish ratio, bowl profile), simple models cannot be used since this aspects can be only addressed by complex CFD codes. The experience suggests that small differences in the bowl parameters (and often on the bowl profile) can produce very large differences in the behavior of the engine in terms of pollutant emissions.

Moreover, the optimization of the engine design requires the development of specific mesh generator tools for each CFD simulation codes. In literature, the level of complexity of these tools as been increased in time to allow a larger variety in the shape of the bowl, a better quality of the mesh in terms of resolution and aspect ratio, the possibility to generate specific file formats like IGES, PATRAN, etc, the necessity to simulate also the solid domain with FEM codes, etc.

Another proof of the particular challenges in optimizing the design parameters has been obtained at the CREA when one of the chamber optimized for a specific

engine was tested on a second engine. When adapted to a slightly larger cylinder diameter and a smaller compression ratio, that bowl profile produced higher emission levels than the baseline chamber of the second engine. Note that this was not due to lack of accuracy of the simulation code since it was able to predict the behavior of the two engines without any tuning of the models. Thus, the optimization was to be repeated for the second engine to obtain a reduction of emissions for that engine.

The validation of the CFD model also includes a sensitivity study to assess the lowest mesh resolution compatible with the desired accuracy and this resolution has to be preserved for all the investigated chamber designs. Once the model has been validated, it is preferable to orient the optimization to the improvements achievable with respect to a baseline case and not to the achievement of specific target values of the goals. In fact, even if validated, the model always introduce some errors in the evaluation of the engine behavior when changing the boundary conditions with respect to those considered in the validation. Thus, the absolute values of the output variables can be meaningless. On the other hand, a good engine model is expected to capture the trend of the output when changing the value of the design variables. For this reasons, the author suggests to compare the output of each design with those of the baseline conditions as simulated with the same simulation code and on the same computing architecture. In this way, it is also possible to directly monitor the optimization process by analyzing in real time the improvement achieved with respect to the initial configuration.

An experimental validation of the method is also useful at the end of the optimization. For the test case presented in this chapter, one of the optimized chamber was built and tested at operating conditions similar to those used in the optimization. The experimental results showed that the optimized chamber was effective in reducing soot and HC emissions and the measured reduction of soot was higher than the calculated one (up to 50% for mode 2). As far as NOx emissions are concerned, a better NOx-soot trade-off was also obtained. However, the experimental validation also revealed another important aspect to be kept in mind. As discussed in the introduction, the author choose to keep constant the control parameters, in particular the injection strategy, and to focus the optimization only on the design parameter of the combustion chamber. When tested with injection strategies different from those used in the optimization, the optimized chamber didn't perform better that the baseline one with the same strategy. This stresses the strong interaction between the air flow field (generated by the chamber profile) and the spray distribution (resulting from the injection specification) on the pollutant mechanisms of formation.

In the case of mechanical components like the engine piston is also mandatory to consider not only the fluid dynamic behavior but also the mechanical response of the component to the stresses generated by the pressure flow field. A shape of the bowl that is able to reduce emissions but strongly decrease the mechanical resistance of the piston cannot be accepted. The mechanical resistance is one of the secondary output values that can be included in the optimization as a penalty functions if it can be evaluated with a specific model. These results of the optimization also put in evidence the importance of keeping separated the fitness components. In fact, in the choice of the final configuration to be built, a preference has been given

to the reduction of soot (or particulate) since the NOx emissions could be controlled with the use of EGR. Some months after the construction of the optimized chamber, the massive introduction in the diesel automotive market of particulate filters completely changed the optimization scenario making preferable reducing NOx. With the multi-objective approach followed at CREA, the solution for the new scenario was already available without the necessity to perform a new optimization.

A final consideration can be drawn with respect to the criterion used for the choice of the final configuration. In this investigation, a clustering of the Pareto solutions was performed with respect to the design parameter (genotype) in order to group the chambers with similar geometric characteristics. This approach allowed the effect of the overall combustion chamber aspect to be analyzed with respect to each optimization goals. However, as underlined before, the output values can change significantly also for chambers belonging to the same cluster. From this point of view, it is better to use Multi-Criteria-Decision-Making techniques to perform the final choice of the chamber with respect to the output values (phenotype).

References

[1] Saurer, H.: Improvements in and relating to internal combustion engines of the liquid fuel injection type. Patent N. GB421101 (December 1934)
[2] Heywood, J.B.: Internal Combustion Engine Fundamentals. Mc Graw-Hill, New York (1988)
[3] Tsao, K.C., Dong, Y., Xu, Y.: Investigation of flow field and fuel spray in a direct-injection diesel engine via kiva-ii program. SAE Technical Paper 901616 (1990)
[4] Zhang, L., Ueda, T., Takatsuki, T., Yokota, K.: A study of the effect of chamber geometries on flame behavior in a di diesel engine. SAE Technical Paper 952515 (1995)
[5] Mahakul, B., Bolis, D.A., Crane, G.E.: Deep angle injection nozzle and piston having complementary combustion bowl. Patent N. US5868112 (February 1999)
[6] De Risi, A., Manieri, D., Laforgia, D.: A theoretical investigfation on the effects of combustion chamber geometry and engine speed on soot and nox emissions. In: ASME-ICE1999, Book No. G1127A, vol. 33-1, pp. 51–59 (1999)
[7] Kidoguchi, Y., Sanda, M., Miwa, K.: Experimental and theoretical optimization of combustion chamber and fuel distribution for the low emission di diesel engine. In: 2001 ICE Spring Technical Conference, ASME 2001, vol. ICE-36-2 (2001)
[8] Lisbona, M.G., Olmo, L., Rindone, G.: Analysis of the effect of combustin bowl geometry of a di diesel engine on efficiency and emissions. In: Desantes, J.-M., Whitelaw, J.H., Payri, F. (eds.) Thermo- and Fluid-dynamic Processes in Diesel Engines: Selected Papers from the THIESEL 2000 Conference Held in Valencia, Spain, September 13-15 (2000)
[9] Senecal, P.K., Reitz, R.D.: Simultaneous reduction of engine emissions and fuel consumption using genetic algorithms and multidimensional spray and combustion modeling. SAE Technical Paper 2000-01-1890 (2000)
[10] De Risi, A., Donateo, T., Laforgia, L.: Optimization of the combustion chamber of direct injection diesel engines. SAE Technical Paper 2003-10-1064 (2003)

[11] Senecal, P.K., Pomraning, E., Richards, K.: Multi-mode genetic algorithm optimization of combustion chamber geometry for low emissions. SAE Technical Paper 2002-01-0958 (2002)
[12] De Risi, A., Donateo, T., Laforgia, D., Aloisio, G., Blasi, E.: An evolutionary methodology for the design of a d.i. combustion chamber for diesel engines. In: THIESEL 2004 Conference on Thermo-and Fluid-Dynamic Processes in Diesel Engines (2004)
[13] De Risi, A., Donateo, T., Laforgia, D.: A new advanced approach to design diesel engines. International Journal of Vehicle Design 41(1/2/3/4), 165–187 (2006)
[14] Jeong, S., Minemura, Y., Obayashi, S.: Optimization of combustion chamber for diesel engine using kriging model. Journal of Fluid Science and Tecnology 1(2), 138–146 (2006)
[15] De Risi, A., Donateo, T., Nobile, F., Vadacca, G., Vedruccio, D.: Fluid dynamics and structural behavior of optimized combustion chamber profiles. In: International Conference on CAE and Computational Tecnologies for Industry, Mestre Italy, October 16-17 (2008)
[16] Padula, S.L., Korte, J.J., Dunn, H.J., Salas, A.O.: Multidisciplinary optimization branch experience using isight software. In: 1999 International iSIGHT Users' Conference (1999)
[17] Carrozza, R., Donateo, T., Laforgia, D.: Effect of the combustion chamber profile on the in-cylinder flow field in a direct injection diesel engine. In: 61 Congresso Nazionale ATI, Perugia (2006)
[18] Amsden, A.A., Rourke, P.J.O., Butler, T.D.: Kiva ii - a computer program for chemically reactive flows with sprays. Los Alamos National Labs, LA - 11560 - MS (1989)
[19] Amsden, A.A.: Kiva 3 - a kiva program with block-sructured mesh for complex geometries. Los Alamos National Labs (1989)
[20] Genzale, C., Wickman, D., Reitz, R.D.: An advanced optimization methodology for understanding the effects of piston bowl design in late injection low-temperature diesel combustion. In: International Multidimensional Engine Modeling User's Group Meeting (2006)
[21] Wakisaka, T., Takeuchi, S., Imamura, F., Ibaraki, K., Isshiki, T.: Numerical analysis of diesel spray impinging on combustion chamber walls by means of a discrete droplet liquid-film model. In: Proceeding of COMODIA, pp. 462–492 (1998)
[22] Subramaniam, M., Ruman, M., Reitz, R.D.: Reduction of emissions and fuel consumption in a 2- stroke direct injection engine with multidimensional modelling and an evolutionary search technique. SAE Technical Paper 2003-01-0544 (2003)
[23] Shi, Y., Reitz, R.D.: Assessment of optimization methodologies to study the effects of bowl geometry, spray targeting and swirl ratio for a heavy-duty diesel engine operated at high-load. SAE Technical paper 2008-01-0949 (2008)
[24] De Risi, A., Donateo, T., Laforgia, D.: Optimization of high pressure common rail electro-injector using genetic algorithms. SAE Technical Paper 2001-10-1980 (2003)
[25] De Risi, A., Donateo, T., Laforgia, D.: Choosing an evolutionary algorithm to optimize diesel engines. In: International Conference on CAE and Computational Tecnologies for Industry, Lecce, Italy (2005)
[26] Lee, S., Von Allmen, P., Fink, W., Petropoulos, A.E., Terrile, R.J.: Comparison of multi-objective genetic algorithms in optimizing q-law low-thrust orbit transfers. In: GECCO 2005, June 25-29 (2005)
[27] Luna, F., Nebro, A.J., Alba, E.: Observations in using grid-enabled technologies for solving multi-objective optimization problems. Parallel Computing 32(5), 377–393 (2006)

[28] De Risi, A., Donateo, T., Zurlo, S., Laforgia, D.: 3d simulation and experimental validation of high egr-phcci combustion. In: Proceedings of ICE-Capri 2007 (2007)
[29] Coello, C.A., Pulido, G.T.: Multiobjective optimization using micro-genetic algorithm. In: Genetic and Evolutionary Computation Conference (GECCO 2001), pp. 274–282. Morgan Kaufmann, San Francisco (2001)
[30] Fonseca, C.M., Fleming, J.: Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Fifth International Conference on Genetic Algorithm, pp. 416–423 (1993)

Chapter 21
Robust Preliminary Space Mission Design under Uncertainty

Massimiliano Vasile and Nicolas Croisard

Abstract. This chapter presents the design of a space mission at a preliminary stage, when uncertainties are high. At this particular stage, an insufficient consideration for uncertainty could lead to a wrong decision on the feasibility of the mission. Contrary to the traditional margin approach, the methodology presented here explicitly introduces uncertainties in the design process. The overall system design is then optimised, minimising the impact of uncertainties on the optimal value of the design criteria. Evidence Theory, used as the framework to model uncertainties, is presented in details. Although its use in the design process would greatly improve the quality of the design, it increases significantly the computational cost of any multidisciplinary optimisation. Therefore, two approaches to tackle an Optimisation Problem Under Uncertainties are proposed: (a) a direct solution through a multi-objective optimisation algorithm and (b) an indirect solution through a clustering algorithm. Both methods are presented, highlighting the techniques used to reduce the computational time. It will be shown in particular that the indirect method is an attractive alternative when the complexity of the problem increases.

21.1 Introduction

In the early phase of the design of a space mission, it is generally desirable to investigate as many feasible alternative solutions as possible. At this particular stage, an insufficient consideration for uncertainty would lead to a wrong decision on the feasibility of the mission. Traditionally, a system margin approach is used in order to take into account the inherent uncertainties related to the computation of the system budgets. The reliability of the mission is then independently computed in parallel. An iterative, though integrated, process between the solution design and the reliability assessment should finally converge to an acceptable solution. This chapter describes a way to model uncertainties and introduce them explicitly in the design process. The overall system design is then optimised, minimising the impact of uncertainties on the optimal value of the design criteria. The minimisation of the

Massimiliano Vasile · Nicolas Croisard
University of Glasgow, Department of Aerospace Engineering,
James Watt Building, Glasgow, G12 8QQ, United Kingdom
e-mail: {m.vasile,n.croisard}@aero.gla.ac.uk

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 543–570.
springerlink.com

impact of uncertainties in the design process is generally known as robust design and the associated optimisation process robust optimisation.

In the last few decades, robust design has been gaining wide attention and its applications have been extended from improving the quality of individual components to the design of complex engineering systems. The methods for robust design have progressed from the initial Taguchi's 'parameter design method' (Taguchi 1950) to recent nonlinear programming methods that formulate robust design problems as nonlinear optimisation problems with multiple objectives subject to feasibility robustness.

Most of these approaches are based on gradient methods for multiobjective optimisation and to a treatment of uncertainty with probability theory [10]. More recently, other approaches based on Evidence Theory have been proposed [1, 2, 23, 28]. Some of them use gradient methods in combination with response surfaces [1, 23], others evolutionary computation [28]. Evidence Theory, also know as the Dempster-Shafer theory [8, 25], represents an advantage with respect to probability theory in that it allows modelling both aleatory and epistemic uncertainties, coming from a poor or incomplete knowledge of the design parameters.

Epistemic uncertainties are typical during the preliminary phase of the design of a space mission since the specialists responsible for each of the subsystems composing a spacecraft (structure, propulsion, power, etc.) are asked to provide reasonable estimations regarding the size, mass, power consumption or performance of individual components. The whole design process can be mathematically represented as a multidisciplinary optimisation problem in which a number of design parameters are uncertain or their values derive from opinions or estimations.

In Evidence Theory, the values of uncertain or vague design parameters are expressed by means of intervals with associated belief (degree of confidence in the range of values). Each expert participating in the design, assigns an interval and a belief according to their opinion or rare experimental data. Evidence Theory treats these epistemic uncertainties better than probability theory since there is no reason to prefer one distribution function over another. Ultimately, all the pieces of information associated to each interval are fused together to yield two cumulative values, Belief and Plausibility, that express the confidence range in the optimal design point. In particular the value of Belief expresses the lower limit on the probability that the selected design point remains optimal (and/or feasible) even under uncertainties. More precisely it represents the lowest level of confidence in the computed value of the cost function (and/or the constraints).

Although introducing epistemic and aleatory uncertainties in the design process would greatly improve the quality of the design (and would give a measure of the reliability of the result), it increases significantly the computational cost of any multidisciplinary optimisation. In particular, if the evaluation of the cost function (and/or the constraints) associate to each discipline of a multidisciplinary problem is already computationally expensive. In this chapter we present two approaches to tackle an Optimisation Problem Under Uncertainties (OUU): (a) a direct solution through a multi-objective optimisation algorithm and (b) an indirect solution through a clustering algorithm.

In these two approaches the solution of the OUU problem is addressed through a global optimisation procedure based on evolutionary computation. In particular, in the direct approach the OUU problem is directly solved in the attempt to reconstruct the set of all the Pareto optimal solutions (or a good approximation to it) that maximise the Belief and optimise the cost functions for all the disciplines. The indirect approach is proposed to mitigate the computational cost related to the use of Evidence Theory (in particular the exponential growth of the combinations of uncertainty intervals). The indirect approach tries to find at first the sets of solutions for which the system budgets are within some required values, then it intersects these sets with the interval of uncertainties for the design parameters. The resulting set is a superset of the Pareto optimal one (i.e. it contains the Pareto one).

The preliminary design is here performed by using reduced models for trajectory analysis and system design. When a reduced model is not available we propose the use of surrogate models made of Kriging response surfaces (e.g. the shape of a heat shield may need running a CFD code, in this case every variation in shape would require minutes to hours of computational time). Although the methodology is of general applicability to mission design problems it is here intended for preliminary combined design of interplanetary trajectories in which system level parameters play a consistent role, such as low-thrust or low-thrust multi gravity assist transfers. The chapter will present the solution of the OUU problem associated to the robust design of a real space mission.

21.2 Uncertainties in Space Mission Design

Uncertainties are usually classified in two distinct categories, aleatory and epistemic uncertainty. According to Helton [14, 15], the definition of each type is:

Aleatory Uncertainty arises from what is considered to be an inherent randomness in the behavior of the system under study.
also known as: Stochastic uncertainty, Type A uncertainty, Irreducible uncertainty, Variability, Objective uncertainty

Epistemic Uncertainty arises from a lack of knowledge about a quantity that is assumed to have a fixed value in the context of a particular analysis.
also known as: Subjective uncertainty, Type B uncertainty, Reducible uncertainty, State of Knowledge uncertainty, Ignorance

W.L. Oberkampf considers a third category, **Error**, also called numerical uncertainty, which "*is defined as a recognizable deficiency in any phase or activity of modelling and simulation that is not due to lack of knowledge*" [1]. Such uncertainties are well-known, and a good estimation of the error is generally available. This point distinguishes errors from epistemic uncertainties. Aleatory uncertainties are due to the random nature of input data while epistemic ones are generally linked to incomplete modelling of the physical system, the boundary conditions, unexpected failure modes, etc.

In the case of preliminary space mission design, analysts face both types of uncertainty. For example, the initial velocity of the spacecraft, the gravity model or the solar radiation, all present aleatory uncertainties. On the other hand, a good deal of

the parameters defining the characteristics of spacecraft subsystems are not known a priori and their value cannot be computed because it depends on other unknown parameters. Therefore their value has to be first estimated on the basis of previous experience or educated guesses by a group of experts. The uncertainty associated to those parameters is therefore epistemic.

The classical way to treat uncertainty is through probability theory. A probability density function is well suited to mathematically model aleatory uncertainties, as far as enough data (experimental for instance) are available [1]. Even though, the analyst still has to assume the distribution function and estimate its parameters. Moreover, Bae et al. [3] pointed out that aleatory uncertainty could be in fact epistemic uncertainty when "*insufficient data are available to construct a probability distribution*". In this situation, alternative distributions can represent the uncertainty as the mean, variance and shape are unknown [16].

However, probability fails to represent epistemic uncertainties because there is no reason to prefer one distribution function over another [23]. Indeed, the probability applies only if one can identify a sample of independent, identically-distributed observations of the phenomenon of interest [24]. When uncertainties are express by means of intervals, based on experts opinion or rare experimental data, as it is the case in space mission design, this representation becomes questionable. As pointed out by Helton et al. [15], there is a large conceptual difference between saying that all that is known about a quantity is that its value belongs to an interval [a,b] and saying that the probability distribution of that quantity is uniform on [a,b]. The latter statement, in fact, implies an additional piece of knowledge on where the value of that quantity is located in [a,b].

A few modern theories exist to better represent epistemic uncertainties, without the need to make additional assumptions. These include for example interval analysis [13, 19], Possibility Theory [29], Fuzzy Set Theory [11], Theory of Paradoxical Reasoning [26] or Theory of Clouds [22]. Evidence Theory is an extension of Possibility and Fuzzy Set Theory [18], therefore we propose the use of Evidence Theory in the framework of preliminary space mission design.

21.3 Modelling Uncertainties through Evidence Theory

Evidence Theory was developed by Shafer [25] based on Dempster's original work [9] and is appealing because: it does not require additional assumptions when the available information is poor or incomplete; can model aleatory and epistemic uncertainty; offers a consistent way of combining several sources of information. In the remainder of this section we will give a brief introduction to Evidence Theory.

21.3.1 Frame of Discernment $\mathcal{U}$, Power Set $2^{\mathcal{U}}$ and Basic Probability Assignment

The frame of discernment $\mathcal{U}$, also know as the universal set, is "*a set of mutually exclusive elementary propositions*" [3]. In most engineering applications of Evidence Theory, experts express their belief of an uncertain parameter u being within a given set of intervals. For example, $u \in [a,b]$ is an elementary proposition, thus

an element of $\mathcal{U}$. The frame of discernment can be viewed as the counterpart of the finite sample space in probability theory.

The power set of $\mathcal{U}$, $2^{\mathcal{U}}$, is a set of subsets of $\mathcal{U}$. The level of confidence an expert has on an element of $2^{\mathcal{U}}$ is quantified using the Basic Probability Assignment (BPA) also called mass (m). A BPA satisfies the following three axioms:

$$m(E) > 0, \forall E \in 2^{\mathcal{U}} \tag{21.1}$$

$$m(E) = 0, \forall E \notin 2^{\mathcal{U}} \tag{21.2}$$

$$m(\varnothing) = 0 \tag{21.3}$$

$$\sum_{E \in 2^{\mathcal{U}}} m(E) = 1 \tag{21.4}$$

Therefore, the BPA is a function that maps the power set into $[0,1]$. The elements of $2^{\mathcal{U}}$ are solely defined by their associated BPA being strictly positive, and are commonly called focal elements (FE).

While the set of subsets of the finite sample space in the probability theory constitutes a σ-algebra, the power set $2^{\mathcal{U}}$ does not. This distinguishes fundamentally Evidence Theory from the probability theory. Unlike probability theory, unions and intersections of subsets of $\mathcal{U}$ are not necessarily included in the power set. This means that evidence on the event $\{A \; or \; B\}$ or $\{A \; and \; B\}$ does not imply/require information on either events $\{A\}$ and $\{B\}$. Moreover, the complement of an element of $\mathcal{U}$ is not necessarily in the power set. While $P(\overline{A}) = 1 - P(A)$ is true in probability theory, it does not hold for Evidence Theory.

Therefore, the power set $2^{\mathcal{U}}$ and the BPA are less structured than their counterparts of probability theory. They aim at representing all and only the piece of information available to the analyst. This characteristic is fundamental when the analyst needs to make decisions based on poor or incomplete information.

When more than one parameter are considered uncertain (e.g. u_1 and u_2), the power set is composed of the cartesian products of all the elements of the power sets of each parameter's frame of discernment: $2^{(\mathcal{U}_1,\mathcal{U}_2)} = 2^{\mathcal{U}_1} \times 2^{\mathcal{U}_2}$. Thus the BPA of a given element of $2^{(\mathcal{U}_1,\mathcal{U}_2)}$ is the product of the BPA of the two corresponding focal elements:

$$\forall (FE_1, FE_2) \in 2^{\mathcal{U}_1} \times 2^{\mathcal{U}_2}, \; m_{1 \otimes 2}(FE_1 \times FE_2) = m_1(FE_1) * m_2(FE_2) \tag{21.5}$$

The number of focal elements increases exponentially with the number of uncertain parameters and the number of focal elements of their respective power sets. If N parameters are considered uncertain and n_k represents the number of focal elements of the power set of the k$^{\text{th}}$ uncertain parameter, the total number of focal elements is given by:

$$n_{FE} = \prod_{k=1}^{N} n_k \tag{21.6}$$

This expression is based on the assumption that the different uncertain parameters are independent. This chapter considers that this assumption holds true. For the case of dependant parameters, which is beyond the scope of the present publication, the reader can refer to the work of Ferson et al. [12].

It is worth mentioning that, in general, the pieces of evidence can come from different sources and need to be combined. As highlighted in [23], "*the results of an uncertainty analysis can strongly depend on which combination method is chosen for use*". The choice of the combination rule should be driven principally by the context of the information to be combined. However, in this chapter we consider only the case in which the sources of evidence have already been combined.

21.3.2 Belief and Plausibility Functions

While Probability theory uses a single value for quantifying uncertainty, Evidence Theory uses two values: the lower and upper bounds of the uncertainty quantification. The lower bound is called Belief (*Bel*) and the upper bound Plausibility (*Pl*) and are defined as follows:

$$Bel(A) = \sum_{\substack{FE \subset A \\ FE \in 2^{\mathcal{U}}}} m(FE) \tag{21.7}$$

$$Pl(A) = \sum_{\substack{FE \cap A \neq \varnothing \\ FE \in 2^{\mathcal{U}}}} m(FE) \tag{21.8}$$

Thus, all the propositions with a not null intersection with the set **A** contribute to the *Pl* value while only the propositions included in **A** contribute to the *Bel* value. For example, Fig. 21.1 represents a BPA structure of two uncertain parameters u_1 and u_2. Parameter u_1 can belong to any of the four intervals $[a_1,b_1]$, $[b_1,c_1]$, $[c_1,d_1]$ and $[d_1,e_1]$ while the parameter u_2 can belong to the three intervals $[a_2,b_2]$, $[b_2,c_2]$ and $[c_2,d_2]$. Thus there is a total of twelve focal elements $FE_1,\ldots,FE_{12}$. Let us define the proposition **A** as the area within the dashed curve $\mathscr{C}$. Only the focal elements FE_1, FE_6 and FE_{10} (gray in the figure) are entirely included in $\mathscr{C}$. In addition, FE_2, FE_3, FE_5, FE_7, FE_9 and FE_{11} are partly inside $\mathscr{C}$ (dotted in the figure), therefore only partially implying the proposition **A**. The belief and plausibility of **A** are then:

$$\begin{aligned} Bel(\mathbf{A}) &= m(FE_1) + m(FE_6) + m(FE_{10}) \\ Pl(\mathbf{A}) &= m(FE_1) + m(FE_2) + m(FE_3) + m(FE_5) + m(FE_6) \\ &\quad + m(FE_7) + m(FE_9) + m(FE_{10}) + m(FE_{11}) \end{aligned}$$

If the pair (u_1,u_2) takes their value within $[b_1,c_1]\times[c_2,d_2]$, it fulfils the proposition **A**. However, if it is inside $[c_1,d_1]\times[b_2,c_2]$, it may verify **A** but also may not. Therefore, the belief represents our confidence in **A** being always true while the plausibility is our confidence in **A** being possibly true.

Three important and meaningful relations between belief and plausibility functions come directly from the fact that all basic assignments must sum to 1.

$$Bel(A) + Bel(\overline{A}) \leq 1 \tag{21.9}$$

$$Pl(A) + Pl(\overline{A}) \geq 1 \tag{21.10}$$

$$Pl(A) + Bel(\overline{A}) = 1 \tag{21.11}$$

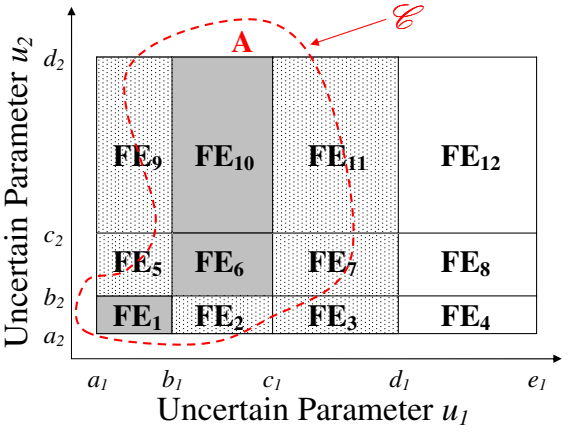


Fig. 21.1 Belief and Plausibility of proposition A in a given *BPA* structure of two uncertain parameters

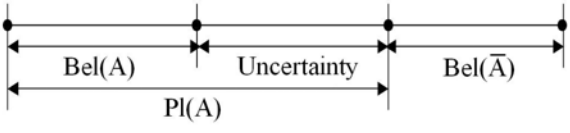


Fig. 21.2 Interpretation of the relation between Belief, Plausibility and uncertainty

where $(\overline{A})$ represents the complement of A. The two first relations shows that, on the contrary of probability, the belief (resp. plausibility) assigned to an event does not uniquely determine the belief (resp. plausibility) of its complement. The last relation means that *Pl* considers the uncertainty, while *Bel* does not (cf. Fig. 21.2).

21.3.3 Cumulative Functions: CBF, CCBF, CPF, CCPF

Whatever the system to be analysed is, it can be represented by a function f giving for a set of input variables $\mathbf{u}$ a quantity y characteristic of it. When the input variables are subject to uncertainties, the analyst is interested in propagating the uncertainties into the output domain. For this purpose, and similarly to probability, the Evidence Theory defines cumulative and complementary cumulative functions to summarise the uncertainty in y.

As Evidence Theory defines two functions to quantify the uncertainty, two pairs of cumulative functions are available to the analyst: (i) the cumulative belief function (CBF) and complementary cumulative belief function (CCBF) related to the belief and (ii) the cumulative plausibility function (CPF) and complementary cumulative plausibility function (CCPF) related to the plausibility. They are defined as follows:

$$CBF : y^* \in \mathcal{Y} \rightarrow Bel_Y\left(y \leq y^*\right) = Bel_U\left(f^{-1}\left(\mathcal{Y}^*\right)\right) \tag{21.12}$$

$$CCBF : y^* \in \mathcal{Y} \rightarrow Bel_Y\left(y > y^*\right) = Bel_U\left(f^{-1}\left(\overline{\mathcal{Y}^*}\right)\right) \tag{21.13}$$

$$CPF : y^* \in \mathcal{Y} \rightarrow Pl_Y\left(y \leq y^*\right) = Pl_U\left(f^{-1}\left(\mathcal{Y}^*\right)\right) \tag{21.14}$$

$$CCPF : y^* \in \mathcal{Y} \rightarrow Pl_Y\left(y > y^*\right) = Pl_U\left(f^{-1}\left(\overline{\mathcal{Y}^*}\right)\right) \tag{21.15}$$

where $\mathcal{Y}^*$ is the set of value of $\mathcal{Y}$ that are lower than y^*:

$$\mathcal{Y}^* = \{y \in \mathcal{Y} \mid y \leq y^*\} \tag{21.16}$$

and:

$$Bel_U\left(f^{-1}\left(\mathcal{Y}^*\right)\right) = \sum_{\substack{FE \in 2^{\mathcal{U}} \\ \forall \mathbf{u} \in FE,\ f(\mathbf{u}) \leq y^*}} m(FE) \tag{21.17}$$

$$Bel_U\left(f^{-1}\left(\overline{\mathcal{Y}^*}\right)\right) = \sum_{\substack{FE \in 2^{\mathcal{U}} \\ \forall \mathbf{u} \in FE,\ f(\mathbf{u}) > y^*}} m(FE) \tag{21.18}$$

$$Pl_U\left(f^{-1}\left(\mathcal{Y}^*\right)\right) = \sum_{\substack{FE \in 2^{\mathcal{U}} \\ \exists \mathbf{u} \in FE,\ f(\mathbf{u}) \leq y^*}} m(FE) \tag{21.19}$$

$$Pl_U\left(f^{-1}\left(\overline{\mathcal{Y}^*}\right)\right) = \sum_{\substack{FE \in 2^{\mathcal{U}} \\ \exists \mathbf{u} \in FE,\ f(\mathbf{u}) > y^*}} m(FE) \tag{21.20}$$

Fig. 21.3 shows an illustration on typical cumulative and complementary cumulative functions in the frame of Evidence Theory.

Computational Complexity. From the definitions Eqs. (21.12–21.15), it can be deduced that the computational time required to compute a cumulative function can

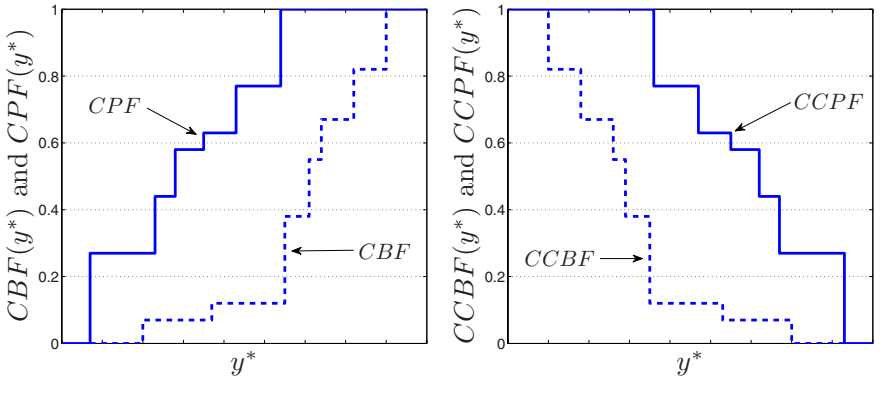


Fig. 21.3 Examples of cumulative belief and plausibility functions (left) and complementary cumulative belief and plausibility functions (right)

become quickly prohibitive as the number of uncertain parameters and the number of intervals per parameter increase. In fact, the total number of focal elements n_{FE} grows exponentially, according to Eq. (21.6), with the number of uncertain parameters N, in particular if the number of focal elements is the same for every uncertain parameter $n_{FE} = n^N$. Furthermore, in order to identify the focal elements included (or intersecting) $f^{-1}(\mathcal{Y}^*)$, the maximum of f over every focal element in $2^{\mathcal{U}}$ has to be computed and compared to y^*. In the event that the system function is convex, this maximum lies at one of the vertices of the focal element, otherwise, an optimisation problem has to be solved over every focal element. Therefore, a generic algorithm that attempted the direct calculation of the Belief and Plausibility values starting from the calculation of the focal elements, would have a computational cost that increases exponentially with the number of uncertain parameters.

In the literature, there exist some approximation methods that attempt a reduction of the number of focal elements, such as (k-l-x)-approximation [27] or the D1-approximation [4], in order to improve the speed of computation of the cumulative functions. Other, more recent methods, instead try to reduce the number of uncertain parameters by evaluating their impact on the value of the cumulative function through a sampling approach [15]. All these techniques could be used as a preprocessing stage to simplify the computation of the cumulative functions by approximating it.

In this chapter, however, we are addressing the OUU without any a priori reduction of the number of focal elements or uncertain parameters. Note that existing approaches like the one proposed by Agarwal et al. [1] address a reliability optimisation problem in which the belief in the satisfaction of the constraints has to be higher than a given value. In this case the construction of the entire CBF is not required, furthermore in the works of Agarwal et al. no specific technique to mitigate the exponential growth of the computational cost is considered.

21.4 Solution of the OUU Problem

21.4.1 Problem Formulation

Let us consider a cost function $f(\mathbf{u},\mathbf{d})$:$\mathbb{R}^{M+N} \longrightarrow \mathcal{Y}$ of some uncertain parameters $\mathbf{u} = [u_1, u_2, \ldots, u_N]$ and design variables $\mathbf{d} = [d_1, d_2, \ldots, d_M]$. The cost function f represents the system or the subsystem budget in the design process (e.g. the overall mass of the spacecraft). Then, let us associate a BPA structure to the frame of discernment $\mathcal{U}$ of the uncertain parameters $\mathbf{u}$. Now, the design variables can be adjusted at will by the designer within a feasibility domain $\mathcal{D}$ to optimise the cost function f. Furthermore, for a given constant y^*, named the threshold, such that $f < y^*$, the generic problem of optimisation under uncertainty (OUU) can be defined as follows:

$$\max_{\mathbf{d}\in\mathcal{D}} \quad CBF_{\mathbf{d}}(y^*) \tag{21.21}$$

The subscript $()_{\mathbf{d}}$ highlights the dependency of the CBF value on the design vector $\mathbf{d}$. Without loss of generality, we assumed here that the cost function is minimised. Note that the use of Belief corresponds to a strict requirement on the actual feasibility of the mission. On the other hand, if the analyst was interested in the possibility of having the mission in some conditions, then Plausibility should be used in Eq. (21.21) instead of Belief.

Although, the solution to problem (21.21) gives a measure of the maximum confidence in the proposition $f < y^*$, it does not give a measure of the best achievable system budget. The simultaneous optimisation of the CBF and of f can be formulated as a bi-objective optimisation problem, such that:

$$\begin{cases} \max\limits_{y^* \in \mathcal{Y}, \mathbf{d} \in \mathcal{D}} & CBF_{\mathbf{d}}(y^*) \\ \min\limits_{y^* \in \mathcal{Y}, \mathbf{d} \in \mathcal{D}} & y^* \end{cases} \tag{21.22}$$

A solution to Problem (21.22) corresponds to a pair $[\mathbf{d}, y^*]$ such that y^* is minimal and the CBF is maximal. Therefore, a pair $[\mathbf{d}, y^*]$ can be said to be Pareto optimal if there is no other pair for which the corresponding CBF is higher and y^* is lower. The image of the set of solutions that are Pareto optimal is called Pareto front. Two sample CBF curves corresponding to two design points are represented in Fig. 21.4. Note that, for different y^*, different $\mathbf{d}$ can be optimal. Furthermore, the following two considerations apply to Problem (21.22):

- for each value of the threshold, one or more design points can maximise the belief.
- an ideal design point $\mathbf{d}_1$ is such that the CBF associated to it is better than the CBF associated to any another design point $\mathbf{d}_2$ for any threshold.

$$\neg \exists y^* \in \mathcal{Y} | CBF_{\mathbf{d}_1}(y^*) < CBF_{\mathbf{d}_2}(y^*)$$

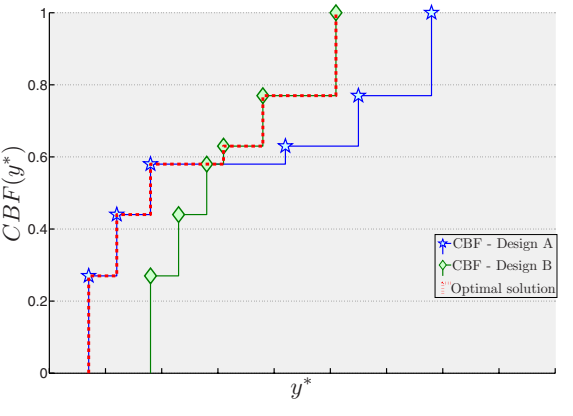


Fig. 21.4 Typical solution of the optimisation under uncertainty problem (dash). The CBF of 2 of the dominating designs are represented ($\star$ and $\diamond$)

The latter point is particularly interesting because defines the optimality of a set (the entire CBF curve) over another. According to this principle, the optimality of a design point can be redefined by saying that a design point $\mathbf{d}_1$ dominates another design point $\mathbf{d}_2$ if every point in the image space corresponding to $\mathbf{d}_1$ is better, lower y^* and higher CBF, than every point in the image space corresponding to $\mathbf{d}_2$. As described in the next section, this definition of optimality will lead to a particular formulation of the OUU.

21.4.2 Direct Solution through a Population-Based Genetic Algorithm

Problem (21.22) is nonlinear and non-differentiable, furthermore it can present multiple locally optimal Pareto sets, therefore for its solution we used the population-based genetic algorithm NSGA-II [7]. The approaches presented in this chapter, however, are independent of the choice of the multiobjective optimiser and any other evolutionary computation algorithm could be used instead. Other examples of the use of NSGA-II to solve the OUU can be found in the work of Limbourg et al. [20]. We propose here two approaches:

- **Bi-objective approach.** This approach is tackling directly the bi-objective formulation problem of Eq. (21.22) based on a direct formulation of the OUU. In this formulation, the number of objectives is limited to 2. However, the number of optimisation variables is increased by one as the threshold is seen as both objective and optimisation variable. Moreover, the typical solution of the OUU problem (cf. Fig. 21.4) is seen in this formulation as a Pareto front.
- **Multi-belief approach.** This approach consists in computing the CBF curve every time a design vector is selected. It dominates the others if there exist at least one belief level for which its corresponding threshold is minimum. Therefore, we can see the problem as a multiobjective optimisation problem where the objectives are to minimise all the minimum thresholds corresponding to the given belief levels.

  The belief levels could be uniformly distributed between 0 and 1. However, we can have access to a few (if not all) achievable belief levels by computing the CBF curve for a specific design vector (e.g. the middle of the design domain $\mathcal{D}$). If we name **bl** the vector of the n_{bel} characteristic belief levels identified, we thus have to solve the following n_{bel}-objectives optimisation problem:

$$\begin{cases} \min\limits_{\mathbf{d}\in\mathcal{D}} \; y^*_{\mathbf{d}}(\mathbf{bl}(1)) \\ \min\limits_{\mathbf{d}\in\mathcal{D}} \; y^*_{\mathbf{d}}(\mathbf{bl}(2)) \\ \qquad\vdots \\ \min\limits_{\mathbf{d}\in\mathcal{D}} \; y^*_{\mathbf{d}}(\mathbf{bl}(n_{bel})) \end{cases} \tag{21.23}$$

  where $y^*_{\mathbf{d}}(\mathbf{bl}(k)) = min\,(y^* \mid CBF_{\mathbf{d}}(y^*) = \mathbf{bl}(k))$ corresponds to the minimal threshold for which the design $\mathbf{d}$ is at belief $\mathbf{bl}(k)$. All the n_{bel} minimal thresholds for a

Algorithm 1. Computing the belief in the bi-objective formulation

Inputs : y^*, $\mathbf{d}$, CBF_{opt}
Outputs: $y^*_{\mathbf{d}}$, $Bel_{\mathbf{d}}$

```
/* Initialise outputs                                          */
Bel_d ← 0;
y*_d ← −∞;
/* Identify the current optimal belief corresponding to y*     */
Bel_opt ← CBF_opt(y*) ;
/* Initialise local variables                                  */
i ← 1 ;                                          /* Counter    */
n_FE ← numel(FE) ;              /* Number of focal elements    */
achBel ← 1 ;                        /* Achievable CBF value    */
/* Main loop                                                   */
while achBel ≥ Bel_opt and i <= n_FE do
    /* Compute the maximum of f on the i^th focal element      */
    y_max ← max_{u∈FE_i} f(d,u) ;
    /* Update the achievable CBF value or the outputs          */
    if y_max ≤ y* then
        Bel_d += m(FE(i));
        y*_d ← max(y_max, y*_d);
    else
        achBel −= m(FE(i));
    endif
    i += 1 ;                                  /* Increase counter */
endw
```

given design are known as soon as the entire belief curve is computed, which is done each time a design is selected.

In the case of the bi-objective approach, the solution vector is $\mathbf{x} = [\mathbf{d}, y^*]$. We firstly rank all the focal elements according to their BPA and compute a complete belief curve for a randomly selected design point. This curve, called CBF_{opt} represents the current best estimate of the optimal CBF. We then start the optimisation process. When evaluating an agent a_i, corresponding to a pair $[\mathbf{d}, y^*]$ (i.e. a new design point and a new threshold), we use Algorithm 1 to compute efficiently the belief associated to it. Because we ranked the focal elements, we add them up starting from the ones with higher BPA value. If the focal elements with higher BPA value are discarded because f is above y^* and the sum of all the remaining BPA's would not allow to improve the current CBF_{opt} value for that particular y^*, then we stop the computation of the belief value associated to that particular solution vector. This is done via the achievable belief (variable *achBel* in Algorithm 1) that tracks the value of the maximum belief that can be achieved during the computation. Furthermore, once a value is assigned to the threshold y^*, the maximisation of the system function f over each focal element is stopped as soon as a value is found above the threshold.

Finally, if the $Bel(\mathbf{d}, y^*)$ value associated to a pair $[\mathbf{d}, y^*]$ is better than $CBF_{opt}(y^*)$, we compute the minimum threshold y^*_{min} such that $Bel(y^*_{min}) = Bel(\mathbf{d}, y^*)$ and update the CBF_{opt}. This guided search for the optimal belief curve is summarised in Algorithm 1. Note that the multiobjective optimisation algorithm had to be slightly modified to make the CBF_{opt} available throughout the computation. Such a modification does not modify the performance of the optimiser. The computational cost of Algorithm 1 is dictated by n_{FE} the number of focal elements. In fact, a maximum of n_{FE} optimisation problems need to be solved every time a design point is evaluated.

In the case of the multi-belief approach, the solution vector is simply $\mathbf{x} = \mathbf{d}$. For each selected design vector the complete belief curve is computed. Though this is more computationally expensive than computing a single belief value, it has the benefit of having only the design vector as optimization variable. Therefore, each design needs to be evaluated once and only once. Additionally, in Algorithm 1, the known maxima of f over all focal elements evaluated during the loop are lost. Thus, while the information was available, it is not used to identify if the current design is dominating for lower belief levels (or identically lower thresholds). By computing the whole belief curve instead we preserve this information.

A more elegant implementation of this approach would consist in redefining the dominance index. If the classical Pareto dominance index:

$$I_i = \left| \left\{ j \mid CBF_{\mathbf{d}_j}(y^*_j) > CBF_{\mathbf{d}_i}(y^*_i) \wedge y^*_j < y^*_i \;\; j = 1, \ldots, n_{pop} \wedge j \neq i \right\} \right| \tag{21.24}$$

is used to define the Pareto optimality of a design vector $\mathbf{d}_i$, where $|.|$ denotes the cardinality of a set, the optimiser cannot evaluate correctly the local Pareto optimality of a point on the $CBF - y^*$ plane since for each design there is a whole curve of points in the $CBF - y^*$ plane. If the Pareto dominance index were defined as in equation Eq. (21.25)

$$I_i = n_{bel} - \left| \left\{ k \in [i, n_{bel}] \mid \forall j \in [1, n_{pop}], y^*_i(\mathbf{bl}(k)) > y^*_j(\mathbf{bl}(k)) \right\} \right| \tag{21.25}$$

then a design with a dominance index lower than n_{bel} is dominating all the others for at least one of the belief levels $\mathbf{bl}$. Therefore leading to the same result as the formulation of Eq. (21.23) and standard dominance index (Eq. (21.24)).

21.4.3 Indirect Solution Approach

The direct computation of the Belief and Plausibility curves for every feasible design point can be a very computationally expensive operation, due to the complexity of the computation of the cumulative functions even after one of the approximation techniques is applied. In order to reduce such a complexity, the idea is to identify, at first, within the cartesian product of the uncertain parameters domain and the design domain, the set $\mathbb{S}_{y^*}$ of subdomains where the system function verifies the proposition $f < y^*$. For a design vector $\mathbf{d}$, an approximation $\widetilde{CBF}_{\mathbf{d}}(y^*)$ of the cumulative belief

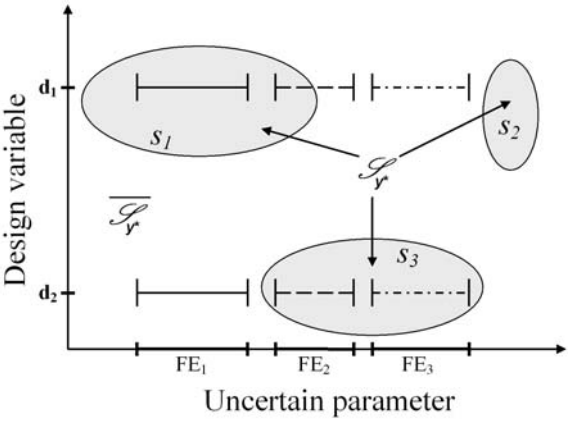


Fig. 21.5 Illustration of the cluster method with 3 focal elements FE_1, FE_2 and FE_3. The proposition $f < y^*$ is true only within the subdomains s_1, s_2 and s_3. Two examples of design point $\mathbf{d}_1$ and $\mathbf{d}_2$ are given

function can then be cheaply computed by adding the mass of the focal elements included in any element of $\mathcal{S}_{y^*}$.

$$\widetilde{CBF}_{\mathbf{d}}(y^*) = \sum_{\substack{(d,FE)\subset s \\ s\in\mathcal{S}_{y^*}}} m(FE) \tag{21.26}$$

Fig. 21.5 illustrates the proposed method. In this example, there are only three focal elements FE_1, FE_2 and FE_3. The set of subdomains where the system function verifies the proposition is $\mathcal{S}_{y^*} = \{s_1, s_2, s_3\}$. Two different design points $\mathbf{d}_1$ and $\mathbf{d}_2$ are represented. Their respective approximation of CBF are:

$$\widetilde{CBF}_{\mathbf{d}_1}(y^*) = m(FE_1); \quad \widetilde{CBF}_{\mathbf{d}_2}(y^*) = m(FE_2) + m(FE_3)$$

Algorithm. To compute the approximation of the CBF function, the set $\mathcal{S}_{y^*}$ of subdomains is computed for increasing values of the threshold until a belief of 1 is found. At each step, sample points verifying the proposition $f(\mathbf{d},\mathbf{u}) < y^*$ are identified, then classified in clusters. The points of a given cluster defines one subdomain s_i of $\mathcal{S}_{y^*}$. Then, the design maximising the approximation of $\widetilde{CBF}(y^*)$ is selected. The algorithm used here is described in Algorithm 2.

To speed up the computation, Axis-Aligned Box (AAB) are used. Each subdomain s_i is associated with its outer AAB (called also the Axis-Aligned Boundary Box) and an inner AAB. If s_i is defined by the set of points of $\mathbb{R}^L$ $(\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_p})$, then its axis-aligned boundary box $oAAB(s_i)$ is defined as:

$$oAAB(s_i) = \left\{\mathbf{y} \in \mathbb{R}^L \mid \forall k, 1 \le k \le L, \min_{1\le j\le L} x_j(k) \le y(k) \le \max_{1\le j\le L} x_j(k)\right\} \tag{21.27}$$

Algorithm 2. Algorithm of the cluster approximation method

Inputs : y^*, $step$
/* Fix a low value for threshold y^* */
Output: Matrix *Out* where each row corresponds to a step. The i[th] row of *Out* is composed of the value of the threshold, the optimum design vector and the maximum approximated cumulative belief found at the i[th] step.

/* Initial sample points */
$X \leftarrow \{[\mathbf{d},\mathbf{u}] \mid f(\mathbf{d},\mathbf{u}) \leq y^*\}$;
$X_{new} \leftarrow \{\}$;
/* Initialise $\widetilde{Bel}_{max}$ */
$\widetilde{Bel}_{max} \leftarrow 0$;
/* Main loop */
while $\widetilde{Bel}_{max} < 1$ **do**
  /* Update the threshold */
  $y^* \leftarrow y^* - step$;
  /* New sampling points */
  $X_{new} \leftarrow \{[\mathbf{d},\mathbf{u}] | (y^* - step) < f(\mathbf{d},\mathbf{u}) \leq y^*\}$;
  /* Update the set of valid sampled point */
  $X \leftarrow \{X, X_{new}\}$;
  /* Identify the valid subdomains */
  Partition in *clusters* the sample points X ;
  foreach *cluster* **do**
    Compute the associated convex hull ;
    Compute the oAAB and an iAAB ;
  endfch
  /* Find the design point giving the highest $\widetilde{CBF}$ */
  $[\widetilde{CBF}_{opt}(y^*), \mathbf{d}_{opt}] \leftarrow \max_{\mathbf{d}\in\mathcal{D}} \widetilde{CBF}_{\mathbf{d}}(y^*)$;
  /* Add a line at the end of the output matrix and save the results */
  $Out(end+1,:) = [y^*, \widetilde{CBF}_{opt}(y^*), \mathbf{d}_{opt}]$;
  /* Update the optimum belief variable */
  $\widetilde{Bel}_{max} \leftarrow \widetilde{CBF}_{opt}(y^*)$;
endw

The inner AAB is an axis-aligned box that is contained within the subdomain s_i. As opposite to the outer AAB, the definition of the inner AAB is not unique. It has been chosen here to centre the inner AAB on the barycentre of the sample points defining s_i and to maximise its relative size such that it remains within s_i.

The idea behind the inner and outer AABs is that it is extremely cheap to check if a focal element is outside or inside an AAB. The focal elements that are outside the outer AAB are guaranteed not to be within $f^{-1}(\mathcal{Y}^*)$ and the one inside the inner AAB are guaranteed to be within $f^{-1}(\mathcal{Y}^*)$. Once this selection process done, only

the focal elements that do not enter in any of those categories need to be checked to compute $\widetilde{CBF}(y^*)$.

Convex hull. In order to identify if any of the remaining focal elements fulfils the proposition $f(\mathbf{d},\mathbf{u}) < y^*, \forall \mathbf{u} \in FE$, one only need to check if its vertices are within the same subdomain s_i. In our implementation s_i is the convex hull of the sample points of the i$^{\text{th}}$ cluster. If $\mathbf{v}$ is a point of $\mathbb{R}^p$, we have:

$$\mathbf{v} \in s_i \Longleftrightarrow \exists \lambda \in \left(\mathbb{R}^+\right)^p \mid \quad (\mathbf{v} = \sum_{k=1}^{p} \lambda(k) * \mathbf{x_k}) \wedge (\sum_{k=1}^{p} \lambda(k) = 1) \tag{21.28}$$

Thus the phase 1 of the revised simplex method used to find a feasible solution to a linear programming problem has been implemented in order to determine whether or not such a vector λ exits [6][5].

It is important to highlight that in this method, no assumptions are made on the convexity of the system function f. Only the subdomains s_i are considered as convex which in the practical application related to space design is reasonable. Another advantage of this method is that it shall identify all the locally optimal design regions and thus identifying various classes of interesting design (as in the direct solution). Finally the global optimum is likely to be found using a simple local optimiser, starting for instance from the barycentre of each cluster.

Pixelisation. A more efficient possibility to identify the subdomains s_i is based on the partition into pixels the cartesian product of the uncertain parameters domain and the design domain. This pixelisation technique replaces the use of the convex hull to identify the subdomains s_i.

It is done by creating first the list of the pixels containing sample points verifying the proposition $f(\mathbf{d},\mathbf{u}) < y^*$, then pruning this list by eliminating the pixels containing at least one sample point violating the proposition. It can be proven that this operation is polynomial with the number of dimensions and subdivisions of each dimension. A focal element is thereafter said valid if all the pixels intersecting it are included in any s_i.

The quality of this approximation technique is related the quality of the sampling of the uncertain and design space and on the number and size of the pixels. It is here implicitly assumed that the set of points that satisfy the proposition $f(\mathbf{d},\mathbf{u}) < y^*$ is finite and can be covered with a finite set of pixels (a reasonable assumption for the problems of interest). The larger the pixels the lower the accuracy of the coverage and the faster the algorithm. However, it has a main advantage over the convex hull one, as it can represent even very non-convex subdomain s_i. Moreover, as the design domain is discretised, a fixed number of possible design vector is accessible. Therefore, one can consider testing them all to identify the best one(s). If not, an optimiser working with binary variables can be use to solve the OUU.

Finally, since the number of pixels is at most equal to the number of admissible sample points in $\mathcal{S}_{y^*}$, it does not grow exponentially if an efficient sampling procedure is used. The sampling algorithm needs to be run only once per every value of the threshold and, therefore, unlike in the direct approach, is independent of the

total number of focal elements. Then, if f is a single-valued function, the pixels generated for a given level of y^* can be preserved when the threshold is increased. In the following we will use NSGA-II also to identify the set $\mathcal{S}_{y^*}$.

21.5 A Case Study: The BepiColombo Mission

In this section, we will present the results of the three previously described approaches applied to the preliminary design of the BepiColombo mission. The objective is to minimise the wet mass of the spacecraft (for the low-thrust part of the mission) considering uncertainties on a few parameters. The first part of this section presents the mass modelling of the spacecraft, i.e. the system function f.

21.5.1 Spacecraft Mass Model

The mass model presented here is a generic one used for preliminary system mass assessment of a Solar Electric Propulsion (SEP) mission. It enables the mass dependence on thrust profile and specific impulse to be evaluated [17]. The total SEP related mass is given by the following equation:

$$\underset{wet}{m_{SEP}} = m_{tank} + m_{array} + m_{rad} + m_{harness} + m_{PPU} + m_{thrusters} + m_{xenon} \tag{21.29}$$

In this equation, the subsystem considered are the tanks (m_{tank}), the solar arrays (m_{array}), the radiator (m_{rad}), the harness equipment ($m_{harness}$), the power processing subsystem(m_{PPU}), the thrusters ($m_{thrusters}$) and finally the propellant required to perform the low thrust transfer (m_{xenon}). The expressions of all these quantities are given in the following subsections.

21.5.1.1 Propellant Mass

The mass of xenon is estimated from the ΔV budget using the rocket equation.

$$m_{xenon} = m_{TLO}\left(1 - e^{-\frac{\Delta V}{\overline{I_{SP}} * g_0}}\right) \tag{21.30}$$

where m_{TLO} is the trans lunar orbit mass, i.e. the wet mass of the spacecraft just after the Earth-Moon system escape (*specific to this mission*, $m_{TLO} = 2400$ kg), g_0 is the gravitational acceleration ($g_0 = 9.80665$ m/s^2), ΔV is the delta V budget for the SEP transfer from the Earth-Moon system escape to the Mercury capture (in ms^{-1}) and $\overline{I_{SP}}$ is the mean specific impulse of the SEP transfer, given in seconds by Eq. (21.31).

$$\overline{I_{SP}} = 0.989 * \underset{max\ T}{I_{SP}} \tag{21.31}$$

In Eq. (21.31), $\underset{max\ T}{I_{SP}}$ is the specific impulse at maximum thrust (in seconds).

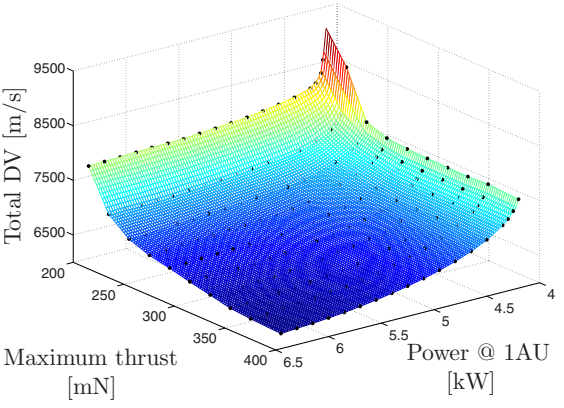


Fig. 21.6 Kriging surrogate of the deep space ΔV for the low thrust mission of BepiColombo

Delta V budget. The delta V budget is composed of the deep space ΔV (cf. below), the ΔV for second Lunar Gravity Assist (40 ms^{-1}), the ΔV for SAA control (100 ms^{-1}), the ΔV for flyby navigation (260 ms^{-1}), the ΔV for other navigation (280 ms^{-1}) and the contingency (+5% of the deep space ΔV).

The deep space ΔV is a quantity essential to any optimisation of spacecraft design. Indeed, it has a direct impact on the propellant mass (cf. Eq.(21.30)) and the tank mass (cf. Eq.(21.32)). In the frame of the BepiColombo test case, this value is computationally expensive to obtain and cannot be done fully automatically. Therefore, it is not feasible to consider it within the model as it is. In order to overcome this issue, a surrogate model has been built based on 180 different transfers priorly computed for various values of P_{1AU} the power to be generated by the solar arrays at 1 Astronomical Unit (AU) and T_{max} the maximum thrust. Moreover, the surrogate reduces significantly the computational time but at the expense of accuracy. For this study, Kriging has been selected via the DACE package [21], with a first order polynomial regression model and an exponential correlation model (cf. Fig. 21.6).

21.5.1.2 Propulsion System Mass

Tank. The mass of the tank is directly proportional to the mass of propellant:

$$m_{tank} = \sigma_{tank} * m_{xenon} \tag{21.32}$$

where σ_{tank} is the specific ratio of the tank subsystem ($\sigma_{tank} = 11\%$).

Solar arrays. The area of the solar arrays required is given by Eq. (21.33).

$$A = \frac{P_{1AU}}{\eta_p * G_s} * \kappa_A \tag{21.33}$$

where η_p is the power conversion efficiency ($\eta_p = 0.22751$), G_s is the solar constant ($G_s = 1367$ W/m^2) and κ_A is the area margin for the solar arrays ($\kappa_A = 1.2$).

Using the area of the solar arrays, their mass is given by Eq. (21.34).

$$m_{array} = \left(A * \rho_{SA} + \underset{array}{m_0} \right) * \kappa_{SA} \tag{21.34}$$

where ρ_{SA} is the specific ratio mass/area of the solar arrays ($\rho_{SA} = 2.89$ kg/m^2), $\underset{array}{m_0}$ is the inevitable structural mass of the solar arrays and κ_{SA} is the mass margin for the solar arrays ($\kappa_{SA} = 1.1$).

Radiator. The radiator (and the associated elements) is sized based on the maximum power P_{max}, thus at the shortest distance to the Sun. In the case of BepiColombo mission, this is at the perihelion of Mercury's orbit, i.e. 0.3 AU. It is calculated using a system of two equations linking the power used by the thrusters, the thrust, the specific impulse and the voltage. The power is a linear function of the thrust and the square root of the voltage. The specific impulse on the other hand is a second order polynomial of the trust with coefficient linear of the square root of the voltage. With $T = T_{max}$ and $I_{SP} = \underset{max\ T}{I_{SP}}$, the voltage is first computed using Eq. (21.35), then P_{max} via Eq. (21.36).

$$I_{SP} = b_2 T^2 + b_1 T + b_0 \tag{21.35}$$

$$P = c * (a_1 T + a_0) \tag{21.36}$$

where a_1, a_0, b_2, b_1 and b_0 are linear function of $\sqrt{V}$. V is the voltage in volt and c a constant.

Once P_{max} is known, the dissipated power while at the perihelion can be evaluated using Eq. (21.37).

$$P_{dis} = \delta_p P_{max} + Q \tag{21.37}$$

where δ_p is the percentage of the maximal power that is wasted ($\delta_p = 0.15$) and Q is the heat to be dissipated at the perihelion of Mercury's orbit.

Two different types of radiator can be envisaged for the BepiColombo mission. The choice depends on the value of the dissipated power (cf. Eq.(21.37)) being above or below a given threshold $\underset{lim}{P_{dis}}$. The mass of the radiator plus its associated structure is calculated using the following equation:

$$m_{rad} = \begin{cases} \left(c_0 + c_1 \frac{P_{dis}}{\underset{lim}{P_{dis}}} \right) * \kappa_{rad} & \text{if } P_{dis} < \underset{lim}{P_{dis}}, \\ \left(c_2 + c_3 \frac{P_{dis}}{\underset{lim}{P_{dis}}} + c_4 \left(\frac{P_{dis}}{\underset{lim}{P_{dis}}} \right)^2 \right) * \kappa_{rad} & \text{otherwise.} \end{cases} \tag{21.38}$$

where c_0, c_1, c_2, c_3 and c_4 are constants and κ_{rad} is the mass margin for the radiator ($\kappa_{rad} = 1.15$).

Table 21.1 Margins used in the low thrust spacecraft model

Margins	Value	Subsystem
ΔV	+5%	ΔV contingency
κ_A	1.20	Area of the solar arrays
κ_{SA}	1.10	Mass of the solar arrays
κ_{rad}	1.15	Mass of the radiator
$\kappa_{harness}$	1.20	Mass of the harness subsystem

Harness. The harness mass is given by the following equation:

$$m_{harness} = m_{0_{harness}} + \rho_{harness} P_{max} \kappa_{harness} \tag{21.39}$$

where $m_{0_{harness}}$ is the inevitable mass of the harness subsystem, $\rho_{harness}$ is the specific ratio mass/power of the harness subsystem ($\rho_{harness} = 1.3763 \cdot 10^{-3}$ kg/W) and $\kappa_{harness}$ is the mass margin for the harness subsystem ($\kappa_{harness} = 1.2$).

Power Processing Unit. The mission of BepiColombo is designed with 4 power processing unit (PPU). The mass of each of them is estimated using an equation linear with the maximum power P_{max} (cf. Eq.(21.36)) and the square of the mean specific impulse (cf. Eq.(21.31)).

Thrusters. Finally, the mass of the thrusters and the associated components varies with the technology used and also the number of thrusters necessary to achieved the required thrust.

$$m_{thrusters} = m_{0_{thrusters}} + n_{thruster} m_{nominal_{thrusters}} \tag{21.40}$$

where $m_{0_{thrusters}}$ is the inevitable mass of the thrusters subsystem, $m_{nominal_{thrusters}}$ is the nominal mass of one thruster and $n_{thruster}$ is the number of thrusters installed aboard the spacecraft ($n_{thruster} = 2$).

Remark. The simple model presented here enables to estimate the mass of the main subsystems of a low thrust spacecraft with only three inputs: (i) the power to be generated by the solar arrays at 1AU P_{1AU}, (ii) the maximum thrust T_{max} and (iii) the specific impulse at maximum thrust $I_{SP_{max\ T}}$. Moreover, margins are conventionally used to take into account uncertainties on this modelling, therefore we report them in table 21.1.

21.5.2 The BPA Structure

In this application, we have selected three parameters as uncertain: η_p, ρ_{SA} and $\rho_{harness}$ that appear in respectively Eq. (21.33), (21.34) and (21.39). Table 21.2 presents their BPA structure. This choice followed recommendations of experts from EADS Astrium. Indeed, various technologies and quality of space solar power

Table 21.2 Uncertainty representation through Evidence Theory

Uncertain parameter	Intervals		Basic probability assignment
	Lower bound	Upper bound	
η_p	0.18959	0.195	0.05
	0.195	0.205	0.15
	0.205	0.215	0.25
	0.215	0.22751	0.55
ρ_{SA}	2.89	3.00	0.10
	3.00	3.10	0.15
	3.10	3.25	0.35
	3.25	3.3105	0.40
$\rho_{harness}$	$1.3763 \cdot 10^{-3}$	$1.4500 \cdot 10^{-3}$	0.05
	$1.4500 \cdot 10^{-3}$	$1.5500 \cdot 10^{-3}$	0.25
	$1.5500 \cdot 10^{-3}$	$1.6000 \cdot 10^{-3}$	0.30
	$1.6000 \cdot 10^{-3}$	$1.6515 \cdot 10^{-3}$	0.40

systems are available to the designer, and their performances varies significantly, impacting directly the value of η_p and ρ_{SA}. Similarly, the specific mass/power ratio of the harness subsystem is dependant on the technology used but also on the internal configuration of the spacecraft, which is unknown at the preliminary stage of the spacecraft design.

The use of system margins is classically to compensate for uncertainties. As we are aiming here at crystallising the uncertainties with Evidence Theory, we selected parameters as uncertain when they were associated to a system margin. In our example, these are κ_A, κ_{SA} and $\kappa_{harness}$. Therefore they are set to 0 for the OUU problem. Note that the BPA structure is such that the effect of the 3 parameters being considered as uncertain is artificially equivalent to applying the default system margins. The consequence is that the optimal design of the OUU is the same as the deterministic one. This is obviously not generally the case but helps here to better comprehend the results.

21.6 Results and Comparisons

The proposed approaches to solve the OUU problem have been tested on the BepiColombo mission described previously. A (nearly) optimal solution was identified after very extensive simulations. This solution was used as a reference to evaluate the quality of the output of each test. The quality of a solution is defined as the area between its CBF curve and the reference one, in the $CBF - y^*$ plane (area shown in Fig. 21.7), normalised dividing by the difference of highest and lowest optimal thresholds. This normalised area is called error area in the following.

The location of the optimal design points is given in Fig. 21.8. It is important to realise that 2 classes of solutions exist for this problem, distinct by the value of

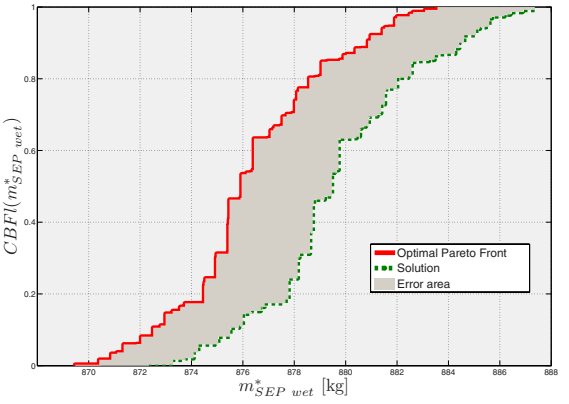


Fig. 21.7 Optimal solution of the OUU problem for the BepiColombo test case. An example of solution found is shown too in dash along with the error area between the two curves

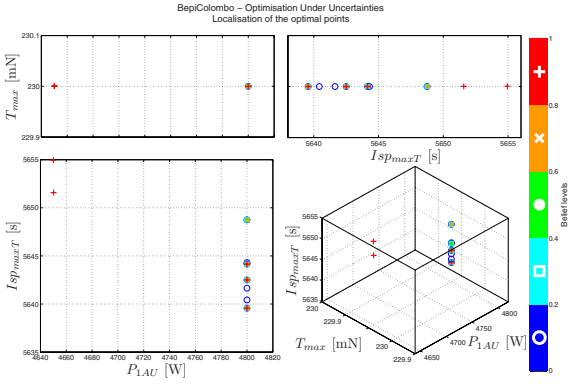


Fig. 21.8 Location of the optimal design points for the OUU - BepiColombo test case

P_1AU: 4,650 or 4,800 W. The optimal maximum thrust is clearly 230 mN and the specific impulse at maximum thrust between 5639 and 5655 seconds.

21.6.1 Direct Solution Simulations

The direct solution approach is tested for three different numbers of total system function evaluations: 100,000, 500,000 and 1,000,000. Each system function evaluation costs 0.00034 s on an Intel Pentium D, 3.6GHz with 1GB of RAM. As the multi-objective optimiser is not deterministic, 100 simulations have been run for both implementations (bi-objective and multi-belief) to draw meaningful conclusions. Moreover, the setting of NSGA-II were: (i) agents: 20 (ii) probability of crossover and mutation: 0.75 and 0.33 (iii) distribution index for crossover and for

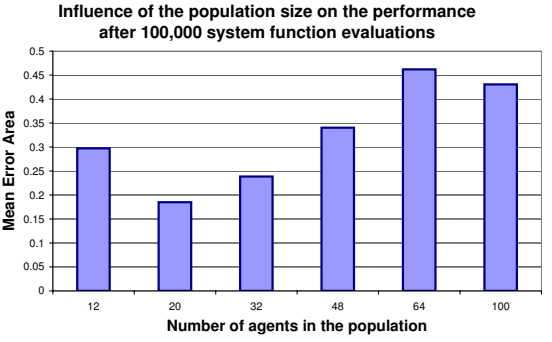


Fig. 21.9 Influence of the number of agents in the performance of NSGA-II for the Bepi-Colombo test case

mutation: 10 and 25. As for any test involving evolutionary algorithm, the settings of the optimiser parameters is tricky and can affect significantly the results. We set the probabilities and distribution indices such that we balance the convergence speed and the global exploration. The most significant parameter however is clearly the size of the population. We set it to 20 agents after running some preliminary tests for up to 100,000 function evaluations. Fig. 21.9 shows that for our selection of probabilities and distribution indices, the optimal population size is around 20.

The BPA structure defined for the BepiColombo test case is composed of 64 adjacent focal elements (cf. table 21.2). As we do not assume convexity here of the system function $\underset{wet}{m_{SEP}}$, a local optimiser[1] is used to identify the maximum of the system function over each of the 64 focal elements.

The average solution found for one hundred function evaluations is given in Fig. 21.10. The mean value of the error area and its variance are given for all simulations in table 21.3. Furthermore, we consider that the OUU problem is successfully solved if the optimiser (NSGA-II in this case) identifies the basins of attraction of all the locally optimal design points. Each basin is a class of solutions. In the case of BepiColombo, there are two classes that are identified by the boxes:

$$\begin{aligned} \text{Class 1} &= [4640\text{ W}, 4740\text{ W}] \times [229\text{ mN}, 231\text{ mN}] \times [5620\text{ s}, 5680\text{ s}] \\ \text{Class 2} &= [4780\text{ W}, 4820\text{ W}] \times [229\text{ mN}, 231\text{ mN}] \times [5620\text{ s}, 5680\text{ s}] \end{aligned} \tag{21.41}$$

Table 21.4 gives the percentage of times, over 100 runs, an approach finds solutions within both classes or, in brackets, within only one class. Once again, both approaches give similar results. It is interesting to note that even though the bi-objective gives worse results than the multi-belief approach, it finds solutions in both classes more often. Indeed, as the bi-objective approach associates a design with a given threshold, it does not guarantee that nearly optimal designs are found for the whole range of thresholds, thus leading to a higher error area.

[1] The Matlab® function `fmincon` is used here.

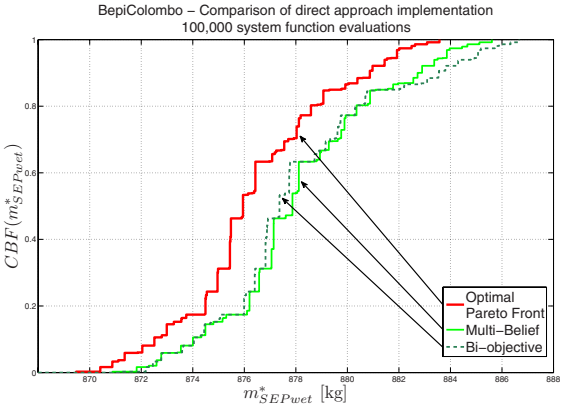


Fig. 21.10 Solution found for the OUU with only 100,000 system function evaluations (BepiColombo test case)

Table 21.3 Mean value and variance of the normalised error area for the OUU BepiColombo test case for 100 runs

n_{val}	Bi-Objective		Multi-Belief	
	mean	variance	mean	variance
100,000	$2.39 \cdot 10^{-1}$	$5.23 \cdot 10^{-2}$	$2.36 \cdot 10^{-1}$	$4.78 \cdot 10^{-2}$
500,000	$9.26 \cdot 10^{-3}$	$2.37 \cdot 10^{-5}$	$9.85 \cdot 10^{-3}$	$1.63 \cdot 10^{-5}$
1,000,000	$5.27 \cdot 10^{-3}$	$2.53 \cdot 10^{-6}$	$3.24 \cdot 10^{-3}$	$3.00 \cdot 10^{-6}$

Table 21.4 Percentage for which solutions have been found over 100 runs in both classes and in at least one class, for the case of BepiColombo

Number of system function evaluations	Bi-Objective		Multi-Belief	
	both classes	one class	both classes	one class
100,000	2%	20%	0%	2%
500,000	94%	99%	58%	100%
1,000,000	100%	100%	79%	100%

21.6.2 Indirect Solution Simulations

For the indirect approach, both the clustering and pixelisation have been implemented and tested. The number of system functions has been limited to 100,000. The resulting approximation of the pareto front is represented in Fig. 21.11. The pixelisation method is such that it may overestimate the real result. This is the reason why it appears to be far better than the clustering, and even a little better than the

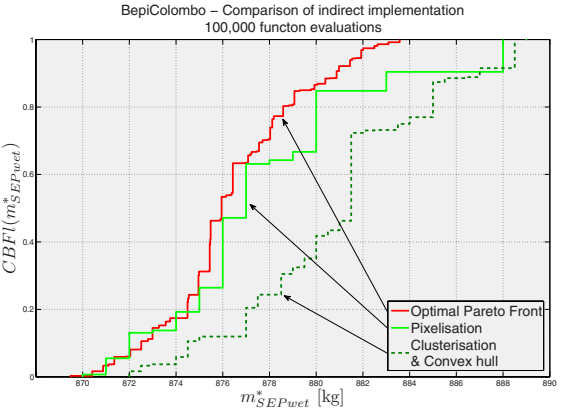


Fig. 21.11 Approximation found with the indirect approaches for the OUU with only 100,000 system function samples (BepiColombo test case)

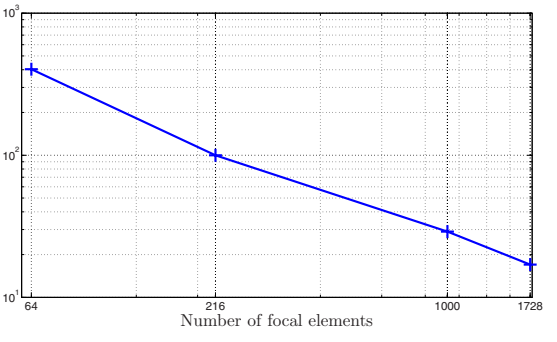


Fig. 21.12 Variation of the number of designs evaluated in the direct approach versus the number of focal elements. The number of system function evaluations has been fixed to 100,000

global solution. However, both the clustering and the pixelisation give a reasonably good approximation of the Pareto front.

Unlike the direct solution, the complexity of the indirect one does not increase with the number of focal elements. Indeed, only the focal elements that lie between the outer and inner axis-aligned boxes need to be checked. Moreover, the number of sample points needed to gather the same information increases polynomially with the number of dimensions. It is not dependant on the number of focal elements in any way. Fig. 21.12 shows the number of design points that the direct approach can test with 100,000 function evaluations. As the number of focal elements increases, the result of the direct approach naturally decreases in quality. On the contrary, an increase in the number of focal elements has no affect on the indirect approach.

21.7 Conclusions

In this chapter, we presented a way to model the uncertainties inherent to preliminary space mission design. The use of Evidence Theory was introduced to represent adequately both aleatory and epistemic uncertainties. The associated robust design problem was formulated as a multi-objective optimisation problem, and two solution approaches were proposed: a direct and an indirect one. The direct approach solves directly the multi-objective optimisation problem (in this chapter we used a population-based multi-objective genetic algorithm). It was tested on two different interpretations of the optimisation under uncertainty problem, however, in both cases, the computational time was increasing exponentially with the number of uncertain parameters. Therefore, an indirect approach was devised to contain the computational cost required to optimise the belief and plausibility functions. The indirect approach, provided good approximations of the belief and plausibility curves with a computational complexity that remained polynomial with the number of uncertain parameters. Therefore, it can be used to produce a first estimation of the optimal solution and to narrow down the design and uncertain domains. The direct approached, instead, could be used on the reduced domains for more accurate results.

Acknowledgements

The authors would like to thank Mr. Stephen Kemble of EADS Astrium UK for providing the reduced system models and plenty of useful suggestions on how to model the design process.

References

1. Agarwal, H., Renaud, J.E., Preston, E.L.: Trust region managed reliability based design optimization using evidence theory. In: Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, vol. 5, pp. 3449–3463 (2003)
2. Agarwal, H., Renaud, J.E., Preston, E.L., Padmanabhan, D.: Uncertainty quantification using evidence theory in multidisciplinary design optimization. Reliability Engineering and System Safety 85(1-3), 281–294 (2004)
3. Bae, H.R., Grandhi, R.V., Canfield, R.A.: Uncertainty quantification of structural response using evidence theory. In: Collection of Technical Papers - AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, vol. 4, pp. 2135–2145 (2002)
4. Bauer, M.: Approximation algorithms and decision making in the dempster-shafer theory of evidence - an empirical study. International Journal of Approximate Reasoning 17(2-3), 217–237 (1997)
5. Bunday, B.D.: Basic Linear programming. Hodder Arnold (1984)
6. Dantzig, G.B.: Linear Programming and Extensions. Princeton University Press, Princeton (1965)
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)

8. Dempster, A.P.: New methods for reasoning towards posterior distributions based on sample data. The Annals of Mathematical Statistics 37(2), 355–374 (1966)
9. Dempster, A.P.: Upper and lower probabilities induced by a multivalued mapping. Ann. Math. Statist. 38, 325–339 (1967)
10. Du, X., Wang, Y., Chen, W.: Methods for robust multidisciplinary design. In: AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit, 41st, Atlanta, GA, no. 1785 in AIAA 2000 (2000)
11. Dubois, D., Prade, H.: Fuzzy sets, probability and measurement. European Journal of Operational Research 40(2), 135–154 (1989)
12. Ferson, S., Nelsen, R.B., Hajagos, J., Berleant, D.J., Zhang, J., Tucker, W.T., Ginzburg, L.R., Oberkampf, W.L.: Dependence in probabilistic modeling, dempster-shafer theory, and probability bounds analysis. Tech. Rep. SAND2004-3072, Sandia National Laboratories (2004)
13. Hayes, B.: A lucid intervals. American Scientist 91(6), 484–488 (2003)
14. Helton, J.C.: Uncertainty and sensitivity analysis in the presence of stochastic and subjective uncertainty. Journal of Statistical Computation and Simulation 57, 3–76 (1997)
15. Helton, J.C., Johnson, J., Oberkampf, W.L., Storlie, C.: A sampling-based computational strategy for the representation of epistemic uncertainty in model predictions with evidence theory. Computer Methods in Applied Mechanics and Engineering 196(37-40 SPEC ISS), 3980–3998 (2007)
16. Hoffman, F.O., Hammonds, J.S.: Propagation of uncertainty in risk assessments: The need to distinguish between uncertainty due to lack of knowledge and uncertainty due to variability. Risk Analysis 14(5), 707–712 (1994)
17. Kemble, S.: Interplanetary Mission Analysis and Design. Springer Praxis Books, Heidelberg (2006)
18. Klir, G.J., Smith, R.M.: On measuring uncertainty and uncertainty-based information: Recent developments. Annals of Mathematics and Artificial Intelligence 32(1-4), 5–33 (2001)
19. Kreinovich, V., Xiang, G., Starks, S.A., Longpré, L., Ceberio, M., Araiza, R., Beck, J., Kandathi, R., Nayak, A., Torres, R., Hajagos, J.G.: Towards combining probabilistic and interval uncertainty in engineering calculations: Algorithms for computing statistics under interval uncertainty, and their computational complexity. Reliable Computing 12, 471–501 (2006)
20. Limbourg, P.: Multi-objective optimization of problems with epistemic uncertainty. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 413–427. Springer, Heidelberg (2005)
21. Lophaven, S.N., Nielsen, H.B., Sondergaard, J.: DACE: a MatLab kriging toolbox. Tech. Rep. IMM-TR-2002-12, Technical University of Denmark (2002)
22. Neumaier, A.: Clouds, fuzzy sets, and probability intervals. Reliable Computing 10(4), 249–272 (2004)
23. Oberkampf, W., Helton, J.C.: Investigation of evidence theory for engineering applications. In: 4th Non-Deterministic Approaches Forum, AIAA, vol. 1569 (2002)
24. Pate-Cornell, M.E.: Uncertainties in risk analysis: Six levels of treatment. Reliability Engineering and System Safety 54, 95–111 (1996)
25. Shafer, G.: A Mathematical Theory of Evidence. Princeton University Press, Princeton (1976)

26. Smarandache, F., Dezert, J.: An introduction to the DSm theory for the combination of paradoxical, uncertain and imprecise sources of information. In: 13th International Congress of Cybernetics and Systems (2005)
27. Tessem, B.: Approximations for efficient computation in the theory of evidence. Artif. Intell. 61(2), 315–329 (1993)
28. Vasile, M.: Robust mission design through evidence theory and multiagent collaborative search. Annals of the New York Academy of Sciences 1065, 152–173 (2005)
29. Zadeh, L.A.: Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems 100(suppl. 1), 9–34 (1999)

Chapter 22
Progressive Design Methodology for Design of Engineering Systems

Praveen Kumar and Pavol Bauer

Abstract. This chapter focuses on a design methodology that aids in design and development of complex engineering systems. This design methodology consists of simulation, optimization and decision making. Within this work a framework is presented in which modelling, multi-objective optimization and multi criteria decision making techniques are used to design an engineering system. Due to the complexity of the designed system a three-step design process is suggested. In the first step multi-objective optimization using genetic algorithm is used. In the second step a multi attribute decision making process based on linguistic variables is suggested in order to facilitate the designer to express the preferences. In the last step the fine tuning of selected few variants is performed. This methodology is named as Progressive Design Methodology (PDM). The method is applied as a case study to design a permanent magnet brushless DC motor drive and the results are compared with experimental values.

22.1 Introduction

The design of complex engineering systems, as such electrical drives and power electronics, requires application of knowledge from several disciplines (multidisciplinary) of engineering (electrical, mechanical, thermal) [1, 2, 3]. The interdisciplinary nature of complex systems presents challenges associated with modelling, simulation, computation time and integration of models from different disciplines. There is a need to develop design methods that can model different degrees of collaboration and help resolve the conflicts between different disciplines. In order to

Praveen Kumar
Indian Institute of Technology, Department of Electronics and Communication Engineering
Guwahati 781039 Assam India
e-mail: praveen_kumar@iitg.ernet.in

Pavol Bauer
Delft University of Technology, Mekelweg 4 2628 CD Delft The Netherlands
e-mail: P.Bauer@ewi.tudelft.nl

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 571–607.
springerlink.com

simplify the design problem assumptions based on the designer's understanding of the system are introduced. The ability and the experience of the designer usually lead to good but not necessarily an optimum design. Hence there is a need to introduce formal mathematical optimization techniques, in design methodologies, to offer an organised and structured way to tackle design problem. A review of different methods for design and optimisation of complex systems is given in [4, 5, 6, 7, 8]. The rise of complexity of systems as well as the number of design parameters needed to be co-ordinated with each other in an optimal way have led to the necessity of using mathematical modelling of system and application of optimisation techniques. In this situation the designer focuses on working out an adequate mathematical model and the analysis of the results obtained while the optimisation algorithms choose the optimal parameters for the system being designed. Marczyk [9] presented stochastic simulation using the Monte-Carlo techniques as an alternative to traditional optimisation. In recent years probabilistic design analysis and optimisation methods have been developed [10, 11, 12] to account for uncertainty and randomness through stochastic simulation and probabilistic analysis. Much work has been performed on developing surrogate-based optimisation (SBO). The SBO methods have been proposed to achieve high-fidelity design optimisation at reduced computational cost. Booker et.al [13] developed a direct search SBO framework that converges to an expensive objective function subject only to bounds on the design variables and that does not require derivative evaluations. Audet et. al. [14] extended that framework to handle general non-linear constraints using a filter for step acceptance [15]. A major barrier to the use of gradient based search methods for engineering design is that complex multidisciplinary design spaces tend to have many apparent local optima. The primary shortcomings of many existing design methodologies is that they tend to be hard coded, discipline or problem specific and have limited capabilities when it comes to incorporation of new technologies. There appears to be a need for a new methodology that can exploit different tools, strategies and techniques which strive to simplify the design cycle associated with large, coupled engineering problems. There are many computational techniques, independently developed computer codes and concepts that are physically separated, yet functionally related. The design methodology presented in this work is a step towards providing the design engineer an environment that allows the combination and/or integration of different techniques. The design methodology presented in this work is named as Progressive Design Methodology (PDM). The above mentioned methods are excellent in design of complex engineering systems but require extensive knowledge of the process itself. The PDM attempts to simplify the design process of complex engineering systems so that a team engineers can use in their day to day work and an extensive knowledge of the design methodology is not a prerequisite. All the components of the PDM can be implemented using commercially available tools and can be easily integrated in the work process of a typical engineering team. Progressive design methodology is a three-step design process. In the first step a simple model of the components of a system is developed and design problem is reformulated as a multi-objective optimisation problem (MOOP). In the second step the results obtained in the MOOP process are analysed and a small set

of feasible solutions is selected. In the final step a detailed model of the variants of the system, as selected from the previous set, are developed and the design variables of the system are fine-tuned. At the end the final design of the system is selected. The aim of this chapter is to:

- Present a framework where optimisation and decision making is employed to accelerate and improve the design of complex systems.
- Support the formulation of the optimisation problem, partly by supporting the selection of optimisation parameters, but also by supporting the formulation of the objective functions. The design problem is often multiobjective in nature, it is therefore natural to formulate the problem as a multiobjective optimisation problem.
- Develop a framework in which system-level simulation models can be composed from sub-system models in different disciplines
- Formalise a multi-domain modelling paradigm that allows to evolve with the design process, increasing in detail as the design process progresses

22.2 Progressive Design Methodology

A design method is a scheme for organising reasoning steps and domain knowledge to construct a solution [16]. Design methodologies are concerned with the question of how to design whereas the design process is concerned with the question of what to design. A good design methodology has following characteristics [17]:

- Takes less time and causes fewer failures
- Produces better design
- Works for a wide range of design requirements
- Integrates different disciplines
- Consumes less resources: time, money, expertise
- Requires less information

An ideal condition in the design of an engineering system will be if all the objectives and constraints can be expressed by a simple model. However, in practical design problems this is seldom the case due to the complexity of the system. Hence, a trade-off has to be made between the complexity of the model and time to compute the model. A complex model will enable us to represent all the objectives and constraints of the system but will be computationally intensive. On the other hand a simple model will be computationally inexpensive but will limit the scope of objectives and constraints that can be expressed. In order to overcome this problem PDM consists of three main phases:

- Synthesis phase of PDM
- Intermediate analysis phase of PDM
- Final design phase of PDM

Since in the first step (synthesis phase) of PDM the detailed knowledge of the system is unavailable, the optimization process is exhaustive. If complex models are

used in this stage then the computational burden will be overwhelming. In order to facilitate the initial optimization process only those objectives and constraints are considered that can be expressed by simple mathematical models of the system. In the synthesis process a set of feasible solutions (Pareto Optimal solutions) is obtained, Fig. 22.1 and Fig. 22.2. The important task in engineering design is to generate various design alternatives and then to make preliminary decision to select a design or a set of designs that fulfils a set of criteria. Hence the engineering design decision problem is a multi criteria decision-making problem. In the conceptual stages of design, the design engineer faces the greatest uncertainty in the product attributes and requirements (e.g., dimensions, features, materials, and performance). Because the evolution of the design is greatly affected by decisions made during the conceptual stage, these decisions have a considerable impact on overall cost. In the intermediate analysis process the multi criteria decision making process is carried

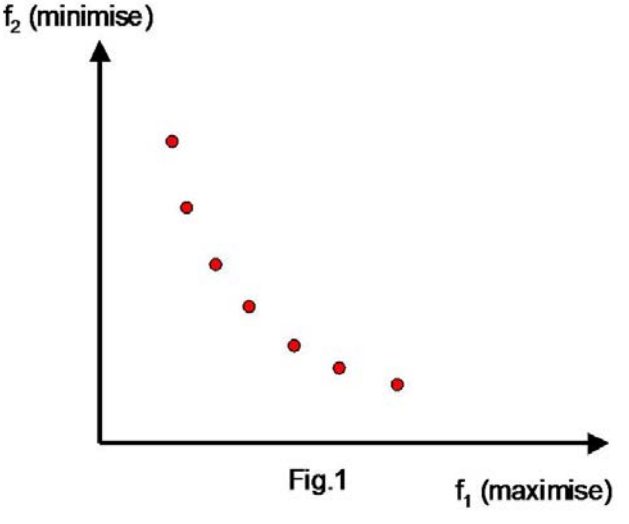


Fig. 22.1 Set of Pareto optimal solutions for a optimisation problem with two objectives

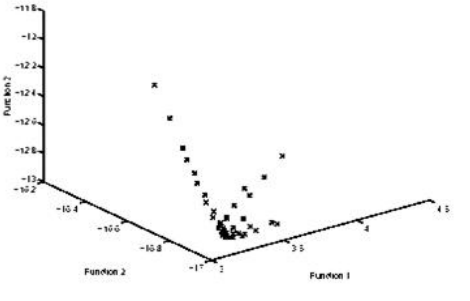


Fig. 22.2 Set of Pareto optimal solutions for a optimisation problem with three objectives

out. This step is a screening process where the large number of solutions obtained from the first step are subjected to process of elimination. In order to achieve the elimination additional constraints are taken into consideration. The constraints considered here are those that cannot be expressed explicitly in mathematical terms, such as manufacturability of an embodiment of the system. In the final design phase detail model of the system is developed. After having executed the synthesis phase a better understanding of the system is obtained and it is possible to develop a detailed model of the system. In this phase all the objectives and constraints that could not be considered in the synthesis phase are taken into consideration. In this phase exhaustive optimisation is not carried out, rather fine tuning of the variables is performed in order to satisfy all the objectives and constraints.

22.3 Synthesis Phase of PDM

In the synthesis phase the requirements of the system to be designed are identified. Based on these requirements system boundaries are defined and performance criterion/criteria are determined. The next step is to determine the independent design variables that will be changed during the optimisation process. The various steps involved in the synthesis phase are:

- System requirements analysis
- Definition of system boundaries
- Determination of performance criterion/criteria
- Selection of variables and sensitivity analysis
- Development of system model
- Deciding on the optimisation strategy

The implementation of the above steps is shown in Fig. 22.3. From Fig. 22.3 it can be seen that the six steps involved in the synthesis phase are not executed in purely sequential manner. After the sensitivity analysis has been done and a set of independent design variables (IDV) has been identified, the designer has to decide if the set of IDV obtained is appropriate to proceed with the modelling process. The decision about the appropriateness of the set of IDV can be made based on previous experience or discussions with other experts. If the set of IDV is not sufficient then it is prudent to go back to system requirement analysis and perform the loop again. This loop can be repeated until a satisfactory set of IDV is identified. Similarly after the model of the system to be designed (target system) is developed, it is important to check if the model includes the system boundaries and the set of IDV. In reality the selection of variables and the development of the model have to be done iteratively since both depend on each other. The choice of variables has influence on modelling and the modelling process itself will influence the variables needed. The details of each of the above steps are given in the following subsections.

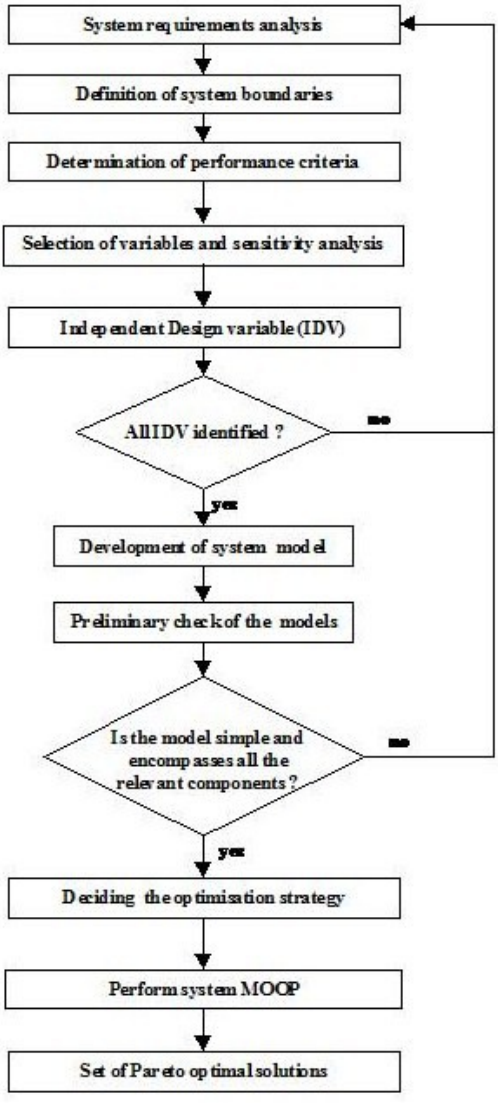


Fig. 22.3 Steps in the synthesis phase of Progressive Design Methodology (PDM)

22.3.1 System Requirements Analysis

The requirements of the system to be designed are analyzed in this phase. The purpose of system requirement analysis is to develop a clear and detailed understanding of the needs that the system has to fulfill. Hence this phase can be a challenging task since the requirements form the basis for all subsequent steps in the design process. The quality of the final product is highly dependent on the effectiveness

of the requirement identification. The primary goal of this phase is to develop a detailed functional specification defining the full set of system capabilities to be implemented.

22.3.2 Definition of System Boundaries

Before attempting to optimise a system, the boundaries of the system to be designed should be identified and clearly defined. The definition of the clear system boundaries helps in the process of approximating the real system [18]. Since, an engineering system consists of many subsystems it may be necessary to expand the system boundaries to include those subsystems that have a strong influence on the operation of the system that is to be designed. As the boundaries of the system increases, i.e. more the number of subsystems to be included, the complexity of the model increases. Hence it is prudent to decompose the complex system into smaller subsystems that can be dealt with individually. However care must be exercised while decomposing the system as too much decomposition may result in misleading simplifications of the reality. For example a brushless direct current (BLDC) motor drive system consists of three major subsystems viz.

- The BLDC motor
- Voltage source inverter (VSI)
- Feedback control

Usually a BLDC motor is designed for a rated load, i.e. the motor is required to deliver a specified amount of torque at specified speed for continuous operation at a specified input voltage. During design process the motor is the primary system under design. However, optimized design of the motor based only on the magnetic circuit may result in misleading results. It is possible that this optimized motor has a high electrical time constant and the VSI is not able to provide sufficient current resulting in lower torque at rated speed and given input voltage. Hence, for the successful design of the BLDC motor it is important to include the VSI in the system, i.e. the boundary of the system is expanded. Of course, it is a different matter that the model of the system that includes the BLDC motor and the VSI is more complicated but nevertheless is closer to the reality.

22.3.3 Determination of Performance Criterion/Criteria

Once the proper boundaries of the system have been defined, performance criterion/criteria are determined. The criterion/criteria form the basis on which the performance of the system is evaluated so that the best design can be identified. In engineering design problems different types of criteria can be classified as depicted in Fig. 22.4 [19]:

- Economic criterion/criteria: In engineering system design problems the economic criterion involves total capital cost, annual cost, annual net profit, return on investment, cost-benefit ration or net present worth.
- Technological criterion/criteria: The technological criterion involves production time, production rate, and manufacturability.
- Performance criterion/criteria: Performance criterion is directly related to the performance of the engineering system such as torque, losses, speed, mass, etc.

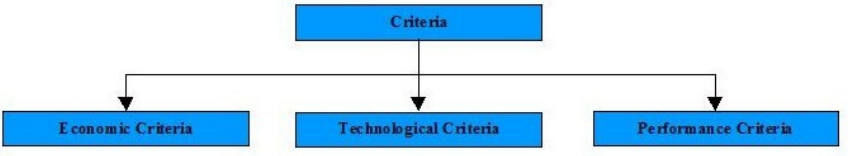


Fig. 22.4 Classification of criterion

In the synthesis phase of PDM the Performance criterion/criteria are taken into consideration because they can be expressed explicitly in the mathematical model of the system. The economic and technological criteria are suitable for Intermediate analysis and Final design phases of PDM because by then detailed knowledge about the engineering systems performance and dimensions are available.

22.3.4 Selection of Variables and Sensitivity Analysis

The next step is selection of variables that are adequate to characterize the possible candidate design. The design variables can be broadly classified as, Fig. 22.5:

- Engineering variables: The engineering variables are specific to the system being designed. These are variables with which the designer deals.
- Manufacturing variables: These variables are specific to the manufacturing domain.
- Price variables: This variable is the price of the product or the system being designed.

In the synthesis phase of PDM engineering variables are considered. There are two factors to be taken into account while selecting the engineering variables. First, it is important to include all the important variables that influence the operation of the system or affect the design. Second, it is important to consider the level of detail at which the model of the system is developed. While it is important to treat all the key engineering variables, it is equally important not to obscure the problem by the inclusion of a large number of finer details of secondary importance [19]. In order to select the proper set of variables, sensitivity analysis is performed. For sensitivity analysis all the engineering variables are considered and its influence on the objective parameters is considered. The sensitivity analysis enables to discard the engineering variables that have least influence on the objectives.

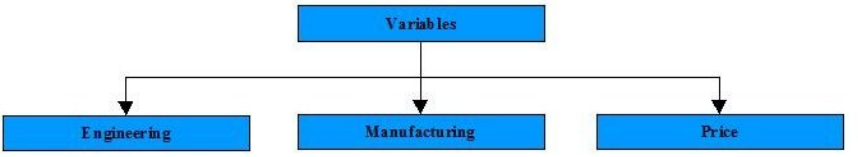


Fig. 22.5 Classification of variables

22.3.5 Development of System Model

Developing a model is to answer a question or a set of questions. If the questions that the model has to answer, about the system under investigation, are specific then it is easier to develop a suitable and useful model. The models that have to answer a wide range of questions or generic questions are most difficult to develop. The most effective process for developing a model is to begin by defining the questions that the model should be able to answer. Broadly models can be classified into following categories [20], Fig. 22.6:

- Physical models: These models are full-scale mock-up, sub-scale mock-up or electronic mock up.
- Quantitative models: These models give numerical answers. These models can be either analytical, simulation or judgmental. These models can be dynamic or static. An analytical model is based on system of equations that can be solved to produce a set of closed form solutions. However finding exact solutions of analytical equations is not always feasible. Simulation models are used in situations where analytical models are difficult to develop or are not realistic. The main advantage of analytical models is that they are faster than numerical mod-

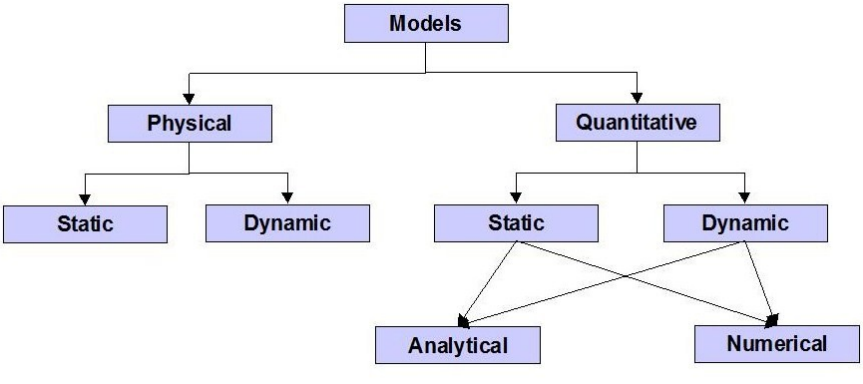


Fig. 22.6 Classification of models

els and hence are suited for MOOP. The major aspect of analytical model is that certain approximations are required to develop analytical models. However in certain cases where approximations cannot be made and a very deep insight of the system are required then numerical simulation methods such as Finite Element Method (FEM), Computational Fluid Dynamics (CFD), etc. have to be adopted. The main drawback of numerical models is that they are computationally intensive and are not suitable for exhaustive optimisation process.

A detailed discussion about the suitability of the models is given in another section of this chapter.

22.3.6 Deciding on the Optimization Strategy

Multi-objective optimization results in a set of Pareto optimal solutions specifying the design variables and their objective tradeoffs. These solutions can be analyzed to determine if there exist some common principles between the design variables and the objectives [21]. If a relation between the design variables and objectives exits they will be of great value to the system designers. This information will provide knowledge of how to design the system for a new application without resorting to solving a completely new optimization problem again. The principles of multi-objective optimisation are different from that of a single objective optimisation. When faced with only a single objective an optimal solution is one that minimises the objective subjected to the constraints. However, in a multi-objective optimisation problem (MOOP) there are more than one objective functions and each of them may have a different individual optimal solution. Hence, many solutions exist for such problems. The MOOP can be solved in three different ways depending on when the Decision Maker (DM) articulates his preference concerning the different objectives [21]. The classification of the strategies is as follows, Fig. 22.7:

- Priori articulation of preference information: In this method the DM gives his preference to the objectives before the actual optimisation is conducted. The objectives are aggregated into one single objective function. Some of the optimisation techniques that fall under this category are weighted-sum approach [22, 23], Non-Linear Approaches [24], Utility Theory [24, 25].
- Progressive articulation of preference information: In this method the DM indicates the preferences for the objectives as the search moves and the decision-maker learns more about the problem. In these methods the decision maker either changes the weights in a weighted-sum approach [26], or by progressively reducing the search space as in the STEM method of reference [27]. The advantages of this method are that it is a learning process where the decision-maker gets a better understanding of the problem. Since the DM is actively involved in the search it is likely that the DM accepts the final solution. The main disadvantage of this method is that a great degree of effort is required from the DM during the entire search process. Moreover the solution depends on the preference of one DM and if the DM changes his/her preferences or if a new DM comes then the process has to restart.

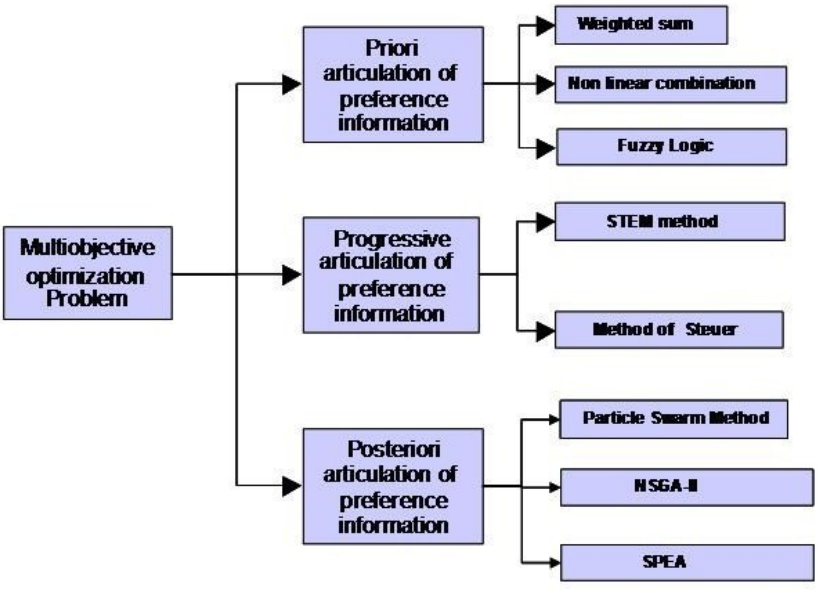


Fig. 22.7 Classification of optimisation methods based on aggregation of information

- Posteriori articulation of preference information: In this method the search space is scanned first and Pareto optimal solutions are identified. This set of Pareto optimal solution is finally presented to DM. The main advantage of this method is that the solutions are independent of DM's preferences. The process of optimisation is performed only once and Pareto optimal set does not change as long as the problem description remains unchanged. The disadvantage of this method is that they need large number of computations to be performed and the DM is presented with too many solutions to choose from.

The principle goal of multi-objective optimisation algorithms is to find well spread set of Pareto optimal solutions. Each of the solutions in the Pareto optimal set corresponds to the optimum solution of a composite problem trading-off different objective among the objectives. Hence each solution is important with respect to some trade-off relation between the objectives. However in real situations only one solution is to be implemented. Therefore, the question arises about how to choose among the multiple solutions. The choice may not be difficult to answer in the presence of many trade-off solutions, but is difficult to answer in the absence of any trade-off information. If a designer knows the exact trade-off among objective functions there is no need to find multiple solutions (Pareto optimal solutions) and a priori articulation methods will be well suited. However, a designer is seldom certain about the exact trade-off relation among the objectives. In such circumstance it is better to find a set of Pareto optimal solutions first and then choose one solution from the set by using additional higher level information about the system being designed.

With this in view in PDM posteriori based optimisation method is used. In principle any posteriori based multiobjective optimisation algorithm such as NSGA-II [28], SPEA 2 [29], etc. can be used in PDM. In this work the NBGA [30] was used. Choosing a suitable solution from the Pareto optimal set forms the second phase of PDM and is described in the next section 22.4.

22.4 Intermediate Analysis Phase of PDM

Once the synthesis process is done and a set of Pareto optimal solutions is determined the next step involves analysis of the solutions. In the conceptual stages of design, the design engineer faces the greatest uncertainty in the product attributes and requirements (e.g., dimensions, features, materials, and performance). Because the evolution of the design is greatly affected by decisions made during the conceptual stage, these decisions have a considerable impact on overall cost. In the intermediate analysis phase the various alternatives obtained from the previous step (synthesis phase) are analysed and a small set of solutions are selected for deeper analysis. The most important tasks in engineering design, besides modelling and simulation, are to generate various design alternatives and then to make preliminary decision to select a design or a set of designs that fulfils a set of criteria. Hence the engineering design decision problem is a multi criteria decision-making problem. It is a general assumption that evaluation of a design on the basis of any individual criterion is a simple and straightforward process. However in practice, the determination of the individual criterion may require considerable engineering judgement [31]. An extensive literature survey on multi criteria decision making is given in the work of Bana de Costa [32]. Carlsson and Fuller [33] gave a survey of fuzzy multi criteria decision making methods with emphasis on fuzzy relations between interdependent criteria. A new elicitation method for assigning criteria importance based on linguistic variables is presented in [34]. Roubens [35] introduced a new pair wise preferred approach that permitted a homogeneous treatment of different kinds of criteria evaluations. A fuzzy model for design evaluation based on multiple criteria analysis in engineering systems is presented by Martinez and Liu et.al. [36]. In the initial phase of development of an engineering system the details of a design are unknown and design description is still imprecise that the most important decisions are made [37]. In this initial engineering design phase, the final values of the design variables are uncertain. Hence at this stage decision making using fuzzy linguistic variables is appropriate. After a decision is made and an alternative or set of alternatives is selected, detailed modelling of the system using standard tools (such as Finite Element Analysis, etc) serve to calculate the performance of the system and also help in reducing the uncertainty in the design variables. In the initial stage of decision making the designers represent their preferences for different values of design variables using a set of fuzzy linguistic variables. Each value of design variable is assigned a preference between absolutely unacceptable and absolutely acceptable. The values of design variables have linguistic preference values. Hence the designer's judgement and experience are formally included in the preliminary design problem.

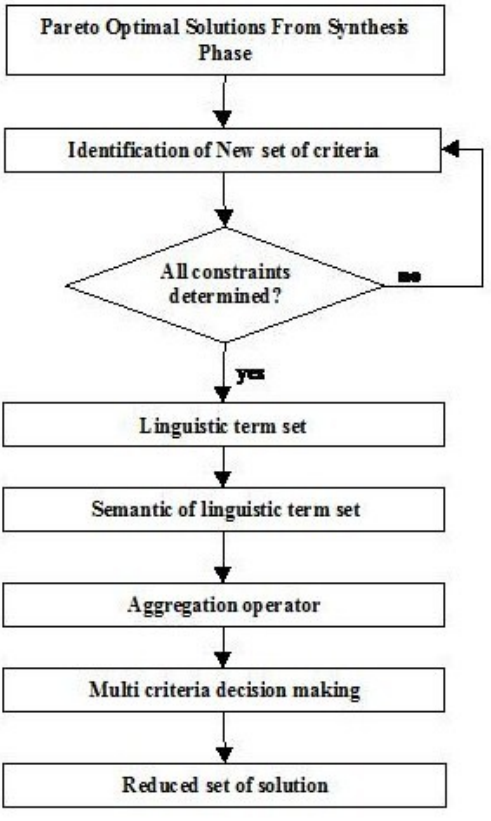


Fig. 22.8 Steps in the intermediate analysis phase of PDM

The general problem is thus a Multi Criteria Decision-Making problem, where the designer is to choose the highest performing design configuration from the available set of design alternatives and each design is judged by several, even competing, performance criteria or variables. A Multi Criteria Decision-Making problem (MCDM) is expressed as:

$$D = \begin{array}{c} \\ A_1 \\ A_2 \\ \vdots \\ A_m \end{array} \begin{array}{c} \begin{array}{cccc} C_1, & C_2, & \cdots, & C_n \end{array} \\ \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{2n} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix} \end{array} \tag{22.1}$$

$$w = \{w_1, w_2, ..., w_n\} \tag{22.2}$$

where A_i, i=1,...,m are the possible alternatives; c_j, j=1,...,n are the criteria with which alternative performances are measured and x_{ij} is the performance score of the alternative A_i with respect to attribute C_j and w_j are the relative importance of attributes. The alternative performance rating x_{ij} can be crisp, fuzzy, and/or linguistic. The linguistic approach is an approximation technique in which the performance ratings are represented as linguistic variable [38, 39, 40] . The classical MCDM problem consists of two phases:

- an aggregation phase of the performance values with respect to all the criteria for obtaining a collective performance value for alternatives
- an exploitation phase of the collective performance value for obtaining a rank ordering, sorting or choice among the alternatives.

The various parts of intermediate analysis phase of PDM are:

- Identification of new set of criteria
- Linguistic term set
- Semantic of linguistic term set
- Aggregation operator for linguistic weighted information

The flow chart of the above steps is shown in Fig. 22.8.

22.4.1 Identification of New Set of Criteria

In the synthesis stage the constraints imposed on the system are engineering constraints. The engineering constraints are specific to the system being designed and can be considered as criteria based on which decision making is done. Besides engineering constraints there are other non-engineering constraints such as manufacturing limitations. It may be possible that certain Pareto optimal solutions obtained in the synthesis stage may not be feasible from the manufacturing point of view or may be too expensive to manufacture. Hence, in order to determine these constraints a high level of information is to be collected from various experts.

22.4.2 Linguistic Term Set

After determining all the constraints, the next step is to determine the linguistic term set. This phase consists of establishing the linguistic expression domain used to provide the linguistic performance values for an alternative according to different criteria. The first step in the solution of a MCDM problem is selection of linguistic variable set. There are two ways to choose the appropriate linguistic description of term set and their semantic [41]. In the first case by means of a context-free grammar, and the semantic of linguistic terms is represented by fuzzy numbers described by membership functions based on parameters and a semantic rule [41, 42]. In the second case the linguistic term set by means of an ordered structure of linguistic terms, and the semantic of linguistic terms is derived from their own ordered structure which may be either symmetrically/asymmetrically distributed on the [0,1]

scale. An example of a set of seven terms of ordered structured linguistic terms is as follows:

$$S = \{S_0 = none, S_1 = very\ low, S_2 = low, S_3 = medium, S_4 = high, S_5 = very\ high, S_6 = perfect\} \tag{22.3}$$

22.4.3 Semantic of Linguistic Term Set

The semantics of the linguistic term set can be broadly classified into three categories (Fig. 22.9), (a) Semantic based on membership functions and semantic rule [43, 44, 45], (b) Semantic based on the ordered structure of the linguistic term set [41, 46, 47, 48] and (c) Mixed semantic [46, 48].

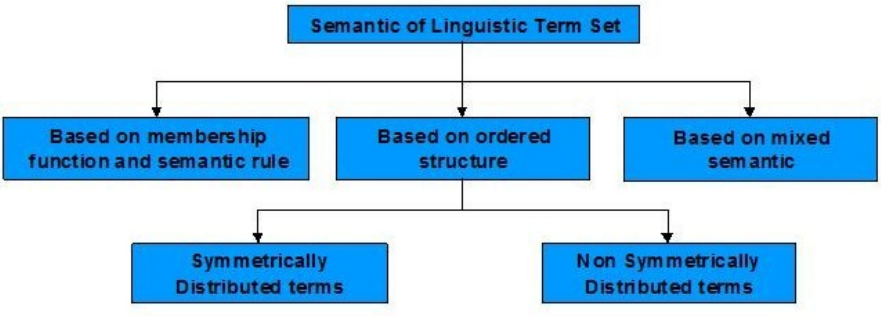


Fig. 22.9 Classification of semantic of linguistic term set

22.4.4 Aggregation Operator for Linguistic Weighted Information

Aggregation of information is an important aspect for all kinds of knowledge based systems, from image processing to decision making. The purpose of aggregation process is to use different pieces of information to arrive at a conclusion or a decision. Conventional aggregation operators such as the weighted average are special cases of more general aggregation operators such as Choquet integrals [49]. The conventional aggregation operators have been articulated with logical connectives arising from many-valued logic and interpreted as fuzzy set unions or intersections [50]. The latter have been generalised in the theory of triangular norms [51]. Other aggregation operators that have been proposed are symmetric sums [52], null-norms [53], uninorm [54], apart from other. The aggregation operators can be grouped into the following broad classes [50]:

- **Operators generalising the notion of conjunction** are basically the minimum and all those functions f bounded from above by the minimum operators.
- **Operators generalising the notion of disjunction** are basically the maximum and all those functions f bounded from below by the maximum operations

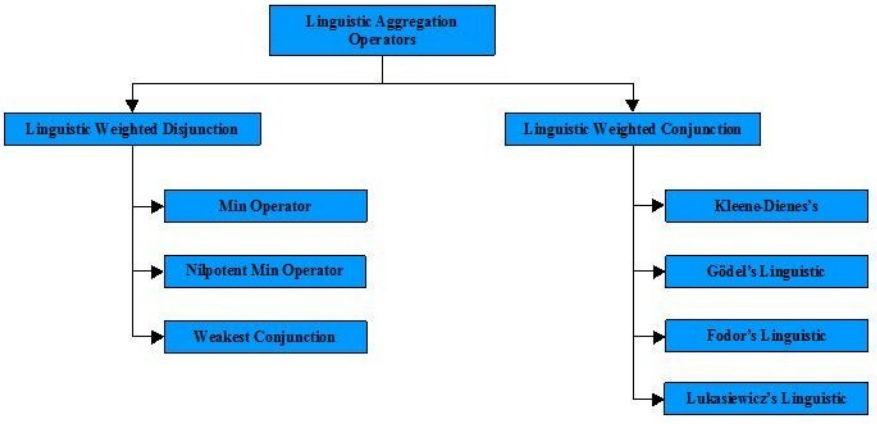


Fig. 22.10 Classification of aggregation operator for linguistic variables

- **Averaging operators** are all those functions lying between the maximum and minimum.

For linguistic weighted information the aggregation operators mentioned above have to be modified for linguistic variables and can be placed under two categories [55] Linguistic Weighted Disjunction (LWD) and Linguistic Weighted Conjunction (LWC). In Fig. 22.10 the detailed classification of the linguistic aggregation operators is shown. In the following example the mathematical formulation of LWD and LWC is given. In order to illustrate each of the above mentioned linguistic aggregation operators the following example is considered [56]: **Example:** For each alternative an expert is required to provide his/her opinion in terms of elements from the following scale

$$S = \{OU(S_7), VH(S_6), H(S_5), M(S_4), L(S_3), VL(S_2), N(S_1)\} \tag{22.4}$$

where OU stands for Outstanding, VH for Very High, H for High, M for Medium, L for Low, VL for Very Low, N for None. The expert provides the opinion on a set of five criteria . An example of criteria as for electrical drive can be:

C_1=*Mass of the motor (Minimum mass is 100 gram and maximum mass is 800 gram)*

C_2=*Cost of the electrical drive (Minimum cost is 10 Euros and maximum cost is 80 Euros)*

C_3=*Losses in the electrical drive (Minimum loss is 10 watts and maximum loss is 80 watts)*

C_4=*Electrical time constant (Minimum loss is 0 .1 milliseconds and maximum time constant is 0.8 milliseconds)*

C_5=*Moment of inertia of the motor (Minimum moment of inertia is 1 and maximum moment of inertia is 8)*

Then perofrmance of each alternative is also defined in terms of the scale. The performance of each alternative is also defined in terms of the scale shown above. The scale is evenly distributed and the scale for each alternative is given in Table 22.1 [55]. The problem is to select a drive that has lowest losses, lowest cost, lowest mass, low electrical time constant and low moment of inertia. The motor is to be used in a hand held drill. For this application the mass of the motor and its cost are very important because a lighter motor with a low cost will be most preferred. Hence these two criteria are given Very High (VH) importance. For this application the efficiency of the motor is of moderate importance and is given a Medium (M) importance. The electrical time constant and moment of inertia of the rotor are important from the dynamic behaviour of the motor and are not very important for the application in hand held drill and are given low (L) and Very Low (VL) importance. The importance to each criterion is shown in Table 22.2. The performance of an alternative on all the criteria is also shown in Table 22.2; in brackets the numerical value is given. The aggregation of the weighted information using Linguistic Weighted Conjunction (LWC) is defined as follows

$$f = LWC\left[\left(w_1, a_1\right), \cdots, \left(w_m, a_m\right)\right] \tag{22.5}$$

where $LWC = MIN_{i=1,\cdots,m} Max\left(Neg\left(w_i\right), a_i\right)$ and m is the number of alternatives. An example of LWC is Kleene-Diene's Linguistic Implication Function $LI_1^{\rightarrow}$ [58]:

$$LI_1^{\rightarrow}\left(w, a\right) = \left(Neg\left(w\right), a\right) \tag{22.6}$$

Table 22.1 The relation between numerical values and linguistic variables

	N	VL	L	M	H	VH	OU
C_1	100-200	200-300	300-400	400-500	500-600	600-700	700-800
C_2	10-20	20-30	30-40	40-50	50-60	60-70	70-80
C_3	10-20	20-30	30-40	40-50	50-60	60-70	70-80
C_4	0.1-0.2	0.2-0.3	0.3-0.4	0.4-0.5	0.5-0.6	0.6-0.7	0.7-0.8
C_5	1-2	2-3	3-4	4-5	5-6	6-7	7-8

Table 22.2 Importance and score of alternative

Criteria	C_1	C_2	C_3	C_4	C_5
Importance Weight (w)	VH	VH	M	L	VL
Score of Alternative 1	M(425)	L(34)	OU(77)	VH(0.65)	OU(7.6)
Score of Alternative 2	M(460)	OU(75)	VH(64)	VH(0.67)	H(5.6)
Score of Alternative 3	H(572)	M(47)	VH(64)	H(0.53)	OU(7.8)
Score of Alternative 4	OU(72)	M(45)	H(53)	VH(0.66)	H(5.8)
Score of Alternative 5	H(550)	M(46)	H(55)	OU(0.74)	VH(6.5)

Based on the example given in Table 22.1 the net performance of the first alternative based on $LI_1^{\rightarrow}$ is:

$$f_1 = MIN\left[LI_1^{\rightarrow}(VH,M), LI_1^{\rightarrow}(VH,L), LI_1^{\rightarrow}(M,OU), LI_1^{\rightarrow}(L,VH), LI_1^{\rightarrow}(VL,OU)\right]$$
$$= MIN\left[M,L,OU,VH,OU\right] = L \tag{22.7}$$

The final score of the second alternative is

$$f_2 = MIN\left[LI_1^{\rightarrow}(VH,M), LI_1^{\rightarrow}(VH,OU), LI_1^{\rightarrow}(M,VH), LI_1^{\rightarrow}(L,VH), LI_1^{\rightarrow}(VL,H)\right]$$
$$= MIN\left[M,OU,VH,VH,VH\right] = M \tag{22.8}$$

The final score of the third alternative is

$$f_3 = MIN\left[LI_1^{\rightarrow}(VH,H), LI_1^{\rightarrow}(VH,M), LI_1^{\rightarrow}(M,VH), LI_1^{\rightarrow}(L,H), LI_1^{\rightarrow}(VL,OU)\right]$$
$$= MIN\left[H,M,VH,H,OU\right] = M \tag{22.9}$$

The final score of the fourth alternative is

$$f_4 = MIN\left[LI_1^{\rightarrow}(VH,OU), LI_1^{\rightarrow}(VH,M), LI_1^{\rightarrow}(M,H), LI_1^{\rightarrow}(L,VH), LI_1^{\rightarrow}(VL,H)\right]$$
$$= MIN\left[OU,M,H,VH,VH\right] = M \tag{22.10}$$

The final score of the fifth alternative is

$$f_5 = MIN\left[LI_1^{\rightarrow}(VH,M), LI_1^{\rightarrow}(VH,M), LI_1^{\rightarrow}(M,H), LI_1^{\rightarrow}(L,OU), LI_1^{\rightarrow}(VL,H)\right]$$
$$= MIN\left[M,M,H,OU,VH\right] = M \tag{22.11}$$

Hence on the basis of $LI_1^{\rightarrow}$ the final score of all the alternatives is $[L,M,M,M,M]$. The results of total score of all the five alternatives based on different aggregation operators is summarised below in Table 22.3. From the above the following conclusions can be drawn:

- The choice of linguistic aggregation operator can influence the results of the intermediate analysis process.

Table 22.3 Result of total score of all the alternatives using different aggregation operators

Alternative →	1	2	3	4	5
Min $LD_1^{\rightarrow}$	M	VH	H	VH	H
Nilpotent $LD_2^{\rightarrow}$	M	VH	H	VH	H
Weakest $LD_3^{\rightarrow}$	M	VH	VL	VH	L
Kleene-Diene's $LI_1^{\rightarrow}$	L	M	M	M	M
Gdel's $LI_2^{\rightarrow}$	L	M	M	M	M
Fodor's $LI_3^{\rightarrow}$	L	M	M	M	M
Lukasiewicz's $LI_4^{\rightarrow}$	M	H	M	H	H

- The linguistic weighted disjunction aggregation operators in general give an optimistic average value to alternatives. The Weakest linguistic disjunction gives the least optimistic value to the alternatives.
- The linguistic weighted conjunction aggregation operators in general give a pessimistic average value to the alternatives.
- Out of all the conjunction operators the Lukasiewicz's implication operator gives the least pessimistic final score to all the alternatives.
- The disjunction aggregation operators are suitable if it is required to select a set of as many alternatives as possible. This situation can arise in the initial design phase when the designer wants to include as many alternatives as possible for further investigation.
- In the initial design process if the number of alternatives is large and there is limited capability, in terms of manpower and computing power, to investigate each alternative then linguistic weighted conjunction operators are preferred.

22.5 Final Analysis Phase of PDM

In the final analysis detailed simulation model of the target system is developed. After intermediate analysis the set of plausible solutions is greatly reduced and hence a detailed simulation for each solution is feasible. After setting up of the simulation model a new set of Independent design variables and objectives is identified. The steps involved in this stage are:

- Detailed simulation model of the target system is developed.
- Independent design variables and objectives are identified.
- Each solution in the reduced solution set is optimised for the new objectives and a set of solutions is obtained.
- Final decision is made.

22.6 Model Suitable for PDM

The engineering systems can be modelled at many levels of approximation. The right model will depend, in general, on the problem to be solved. The type of model needed to synthesise a new design may be different from the type of models required to accurately predict the performance of a single proposed design or to diagnose problems with an existing design. When an engineering system is to be designed and optimised the choice of proper models will have a profound influence on the results. One of the problems with modelling is that, according to the definition, models simplify the reality. This means that some information will be lost somewhere along the line that can cause problems. Hence, it is important to know how the model relates to real system. The Progressive Design Methodology (PDM) involves three essential features namely ***Design***, ***Selection***, and ***Tuning***. The main goal of design is to create a set of feasible new artefact based on requirements. This process is carried out in the ***Synthesis Phase*** of PDM. In this phase the ***design models*** of the system under consideration are used. Using the ***design models*** together with

multiobjective optimization algorithms an initial set of feasible solution is generated. Since the multiobjective optimization is employed and the detailed knowledge of the system is not available, it is prudent to use simple low fidelity models of the system. The advantage of low fidelity models is that they are computationally less intensive and hence are suitable for multiobjective optimization. The suitable ***low fidelity models*** are the analytical models. For many situations it is possible to develop an ***analytical model*** of the system by making suitable assumptions. However if analytical models are not possible then simple numerical models of the system should be used in the synthesis phase of PDM. In the ***Intermediate Analysis*** phase of PDM the ***selection*** is performed. The central challenge of this phase is to select from the set of solutions, obtained in the ***Synthesis Phase***, a subset of suitable solutions. The ***selection*** process involves evaluating the alternatives available. In PDM the alternatives are evaluated based on criteria that cannot be expressed mathematically such as manufacturability of the system. In order to achieve this the ***judgmental models*** are used. The ***judgmental models*** are formed by the deductions and assessments contained in the mind of an expert. In ***Intermediate Analysis*** the expert evaluates each alternative based on judgmental models and assigns preference based on linguistic variables and the entire multi attribute decision making is carried out (chapter 2). After the ***selection*** process a small set of suitable solutions is generated. The ***Final Analysis*** phase of PDM involves the ***tuning*** process. In the ***tuning*** process the system performance criteria are improved by varying system parameters. In order to achieve this, ***high fidelity*** model of the system that is to be designed is developed. Each alternative obtained after Intermediate Analysis phase is evaluated using the high fidelity model and tuning of the system is performed. The ***high fidelity models*** can be developed using finite element methods (FEM), computational fluid dynamic (CFD), etc. These models are computationally intensive but are closer to the actual system and are suitable for ***Final Analysis*** phase of PDM. In the next section the PDM is applied for design of a BLDC motor. The various aspects of PDM are used in the design of BLDC motor.

22.7 Synthesis Phase of PDM for Design of a BLDC Motor Drive

In this section the PDM is applied for the design of a BLDC motor for a specific application. All the steps of PDM are applied and the motor is designed that optimal with respect to the system in which it has to work. In the next subsection the customer requirements are elicited and validated.

22.7.1 Requirement Analysis

The specified parameters of the motor are:

Rated speed 800 rpm (mechanical)
Torque at rated speed 0.2 Nm
Number of phases 3

The aim of the problem is to design a motor with a cogging torque of less than 20 milliNm, maximum efficiency, minimum mass and trapezoidal back emf.

Inverter Full bridge Voltage source inverter
Motor topology Inner rotor with surface mount magnets
Phase connection The phases are connected in star
The additional constraints of the motor are:
Outer stator diameter 40 mm
Max. Length 50 mm
Air gap length 0.2 mm
Maximum input voltage 50 Volts

22.7.2 Definition of System Boundaries

The BLDC motor to be designed is driven by a voltage source inverter (VSI). The VSI topology used here is a full bridge inverter. Hence while designing the motor it is important to include the VSI in the system boundaries so that the motor parameters are not mutually exclusive. This will ensure that the designed motor will produce the required torque when it is integrated with the VSI. The model of the system that includes the BLDC motor and the VSI is more complicated but will ensure a well designed motor. Hence the system boundary under consideration in the synthesis phase consists of:

- ” The BLDC motor (Primary system)
- ” Three phase VSI (Secondary system)

22.7.3 Determining of Performance Criteria

From the requirement analysis the primary objectives that have to be satisfied are:

- Minimum Cogging Torque
- Maximum Efficiency
- Minimum Mass
- Sinusoidal shape of back EMF

In the synthesis phase of PDM only simple model of the BLDC drive is developed. However determining parameters like cogging torque and shape of the back emf requires detailed analytical models or FEM models. The mass and efficiency of the motor can be calculated with relative ease compared to the cogging torque and back emf shape. Hence in the synthesis phase the objectives that will be considered are:

- Minimise the mass
- Maximise the efficiency

A generic topology of BLDC motor with surface mount magnets as shown in Fig. 22.11 is considered. This topology is optimised for minimum mass and maximum efficiency. In the final design the parameters of this optimised generic topology are fine-tuned to reduce the cogging torque and obtain sinusoidal back emf shape. The independent design variables that are used in this case are shown in Table 22.4.

Table 22.4 List of independent variables used in the synthesis phase

Serial No.	Variable name	Symbol	Min. value	Max. Value	Units
1	Number of poles	N_p	2	10	-
2	Number of slots	N_s	3	15	-
3	Length of the motor	L_{mot}	1	15	mm
4	Ratio of inner diameter of motor to outer diameter	α_{dido}	0.1	0.7	-
5	Ratio of magnet angle to pole pitch	α_m	0.1	1	-
6	Height of the magnet	h_m	1	3	mm
7	Reminance field of the permanent magnet	B_r	0.5	1.2	T
8	Maximum allowable field density in the lamination material for linear operation	B_{fe}	0.5	2	T
9	Number of turns in the coils of the motor	N_{turns}	1	100	-
10	Input Voltage	V_{dc}	10	50	V

22.7.4 Development of System Model

22.7.4.1 Motor Model

In this section a simple design methodology for the surface mounted BLDC motor is given [57]. To develop this model certain assumptions have been made. The assumptions made are:

- No saturation in iron parts.
- Magnets are symmetrically placed.
- Slots are symmetrically placed.
- Back emf is trapezoidal in shape.
- Motor has balanced windings.
- Permeability of iron is infinite.

The general configuration of the motor is shown in Fig. 22.11. The motor design equations are developed in detail in [57].

22.7.4.2 Dynamic Model of BLDC Motor

The schematic of the typical voltage source inverter is shown in Fig. 22.12. The coiupled circuit equations of the stator windings in terms of the motor electrical parameters are

$$[V] = [R][i] + [L]\frac{d[i]}{dt} + [e] \tag{22.12}$$

where

$$[V] = [V_a, V_b, V_c]' \tag{22.13}$$

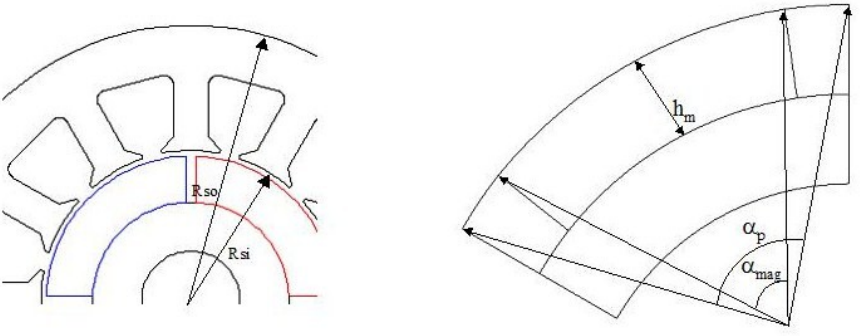


Fig. 22.11 Typical lamination and variables of BLDC motor

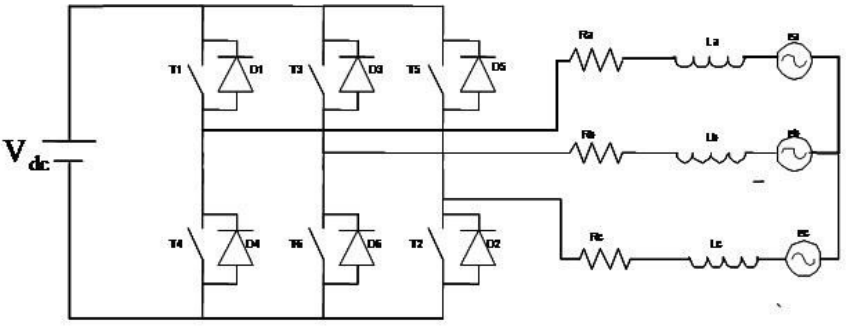


Fig. 22.12 Schematic diagram of a three phase voltage source inverter

$$[R] = \begin{bmatrix} R_{ph} & 0 & 0 \\ 0 & R_{ph} & 0 \\ 0 & 0 & R_{ph} \end{bmatrix} \tag{22.14}$$

$$[i] = [i_a, i_b, i_c]' \tag{22.15}$$

$$[L] = \begin{bmatrix} L_{ph} & 0 & 0 \\ 0 & L_{ph} & 0 \\ 0 & 0 & L_{ph} \end{bmatrix} \tag{22.16}$$

$$[e] = [e_a, e_b, e_c]' \tag{22.17}$$

where R_{ph} and L_{ph} are the phase resistance and phase inductance values respectively defined earlier and V_a, V_b and V_c the input voltages to each phase a, b and c respectively. The induced emf e_a, e_b and e_c and the phase resistance R_{ph} and phase inductance L_{ph} are determined from the motor model described above. The electromagnetic torque is given by

$$T_e = [e_a i_a, e_b i_b, e_c i_c] \frac{1}{\omega_m} \tag{22.18}$$

where ω_m is the mechanical speed of the motor. The analytical solution of the eq. (7) is done following the lines of Nucera et.al. work [58].

22.7.5 Optimisation Strategy

In the present case study optimisation strategy based on Posteriori articulation of preference information is used. To achieve the multiobjective optimisation the Non-dominated sorting Biologically Motivated Genetic Algorithm (NBGA) [30] is used. The parameters of NBGA are as follows

Number of generations =50
Number of individuals =100
Crossover probability = 80
Single point crossover was used.
The mutation rate was fixed between 0 and 10

Hence the multiobjective optimisation problem to be solved is expressed mathematically as

$$min\left\{f_1(x^{\rightarrow}) = P_{cu} + P_{hys} + P_{eddy};\ f_2(x^{\rightarrow}) = M_{iron} + M_{magnet}\right\} \tag{22.19}$$

where P_{cu}, P_{hys}, P_{eddy} are the copper loss, hysteresis loss in the stator yoke, the eddy current loss in the stator yoke and losses in the MOSFET (both switching and conduction losses) respectively and M_{iron} and M_{magnet} are the mass of yoke (stator and rotor) and mass of permanent magnets respectively. subject to $h(x^{\rightarrow}) = T_{motor} \geq$ 0.2Nm where $x^{\Longrightarrow} = (B_r, B_{Fe}, h_m, \alpha_m, \alpha_d, N_m, N_s, N_{turns}, F_{sw}, V_{dc})$ are the independent variables and the limits of the variables are given in Table **??**. In the optimization process several constraints are also included. For example if the r.m.s value of the phase current is higher than the rated current then a violation of constraint takes place. The combinations of variables where constraints are violated are inferior to those in which constraints are not violated.

22.7.6 Results of Multiobjective Optimisation

The results of optimisation given in Fig. 22.13 to Fig. 22.18. From the results it can be seen that for each pole slot combination a number of Pareto optimal solutions are present and as the mass of the motor increases the losses decreases. Since the number of feasible solutions is large the results have to be screened so that a reduced set is obtained. Detailed analysis can be then performed on the reduced set. In the next section the screening process is performed.

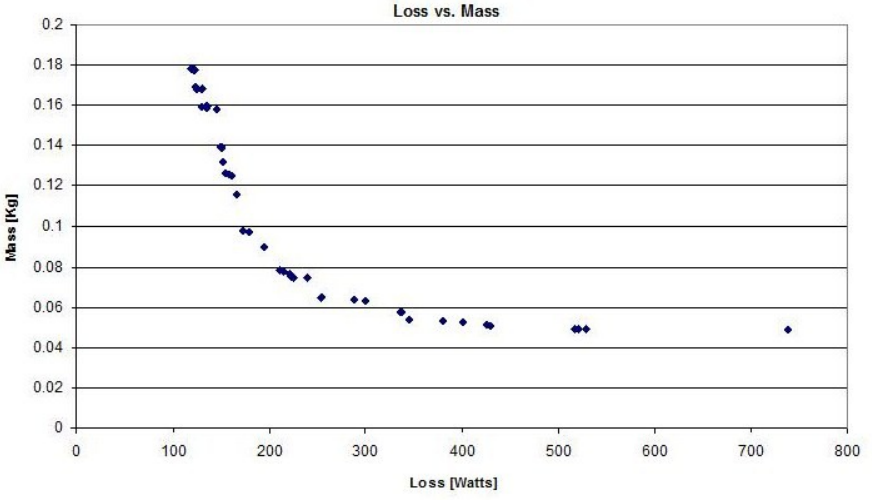


Fig. 22.13 Pareto optimal solutions with N_s=6 and N_p=4

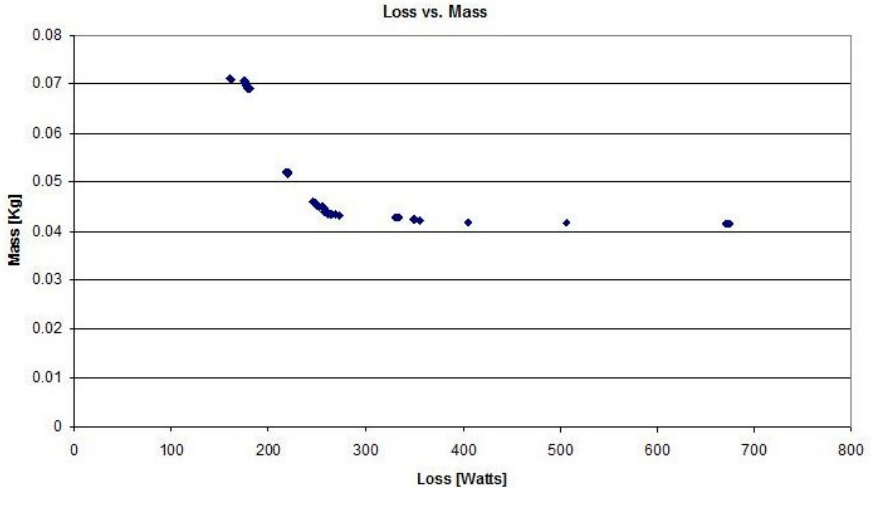


Fig. 22.14 Pareto optimal solutions with N_s=9 and N_p=6

22.8 Intermediate Analysis Phase of PDM for Design of a BLDC Motor Drive

In this section the results of the multiobjective optimization obtained in the previous section are screened to reduce the number of feasible solution set. In order to

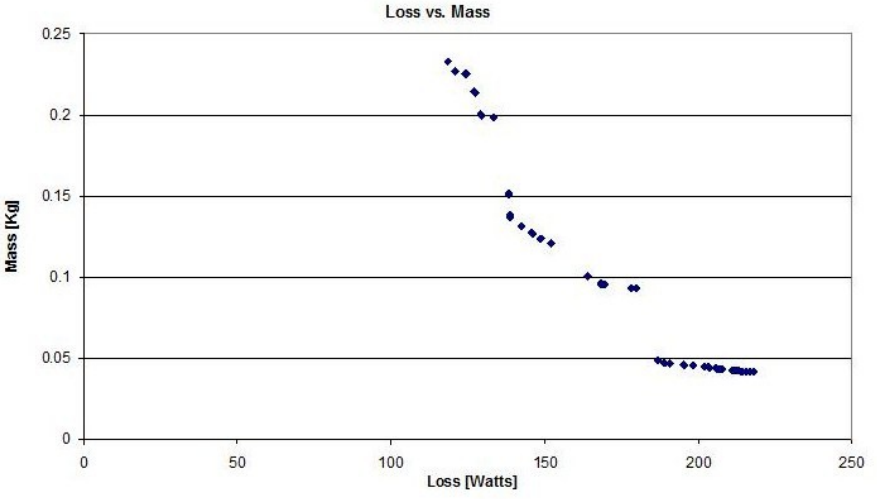


Fig. 22.15 Pareto optimal solutions with N_s=6 and N_p=8

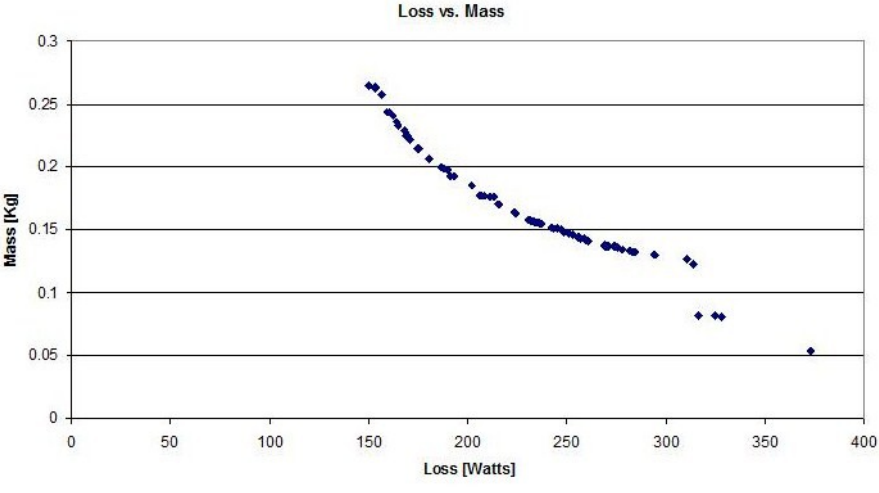


Fig. 22.16 Pareto optimal solutions with N_s=12 and N_p=8

perform the screening process certain parameters are required. Each solution obtained in the previous section is evaluated based on the values these parameters. The application of various steps of intermediate analysis is explained in the following subsection.

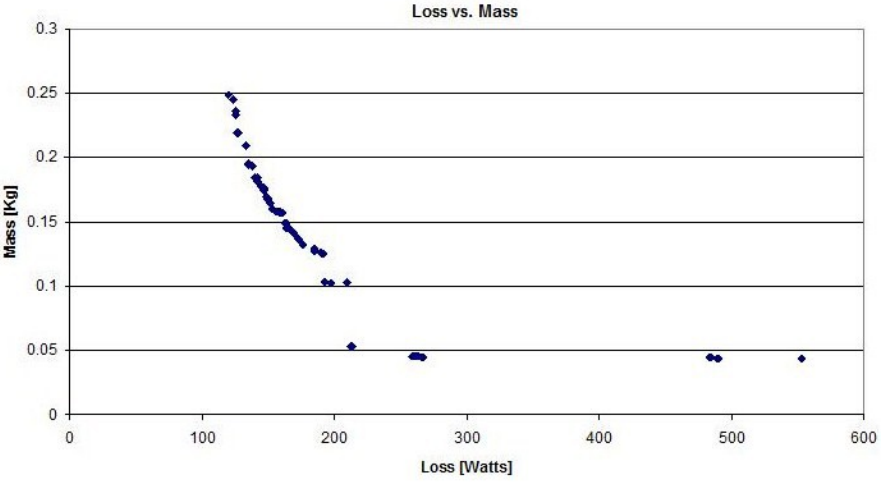


Fig. 22.17 Pareto optimal solutions with N_s=9 and N_p=8

22.8.1 Identification of New Set of Objectives

For decision making the following parameters of the motor are taken into consideration

- Stack length
- Losses
- Mass
- Electrical time constant
- Inertia of the rotor
- Ratio of inner diameter of stator to outer diameter
- Number of turns
- Switching frequency
- Width of the tooth
- Thickness of the stator yoke
- Input Voltage
- Area of slots

The losses and mass of the motor are the primary parameters. A motor with smallest losses and smallest mass is preferable. However as can be seen from the results of the previous section as the mass increases the losses decrease. Hence in the intermediate analysis both are considered for the screening purpose. Electrical time constant of the motor has a direct influence on the dynamic performance of the motor. A motor with lower time constant has a better dynamic response compared to the motor with higher electrical time constant. Similarly the inertia of the rotor is important parameter because it influences the dynamic performance of the motor. A motor with high inertia will accelerate slowly compared to the motor with lower

inertia. The ratio of inner diameter of stator to outer diameter of stator is considered because it has an influence on the end turn of the winding. Switching frequency has an impact on the performance of the motor. Higher switching frequency results is lower torque ripple but higher switching losses and a lower switching frequency results in higher torque ripple but lower switching frequency. The magnetic loading and the mechanical aspects determine the width of the tooth. If the tooth is too thin then it may not be able to withstand the mechanical forces acting on it. Hence in this analysis tooth with higher thickness is preferred. The thickness required for the stator yoke depends on the magnetic loading of the machine as well as on the mechanical properties. If the number of the pole pairs is small, often the allowable magnetic loading and the mechanical loading determines the thickness of the stator yoke. However, if the number of pole pairs is high enough the stator yoke may be thin if it is sized according to the allowed magnetic loading. The mechanical constraints may thus determine the minimum thickness of the stator yoke. In the decision making process it smaller the thickness of stator yoke the better it is. A smaller yoke thickness is preferred because it reduces the mass of the steel lamination required. The area of the slot is considered as an objective because it influences the winding. A slot with smaller area is difficult to wind. Hence in this analysis a larger slot area is preferred.

22.8.2 Linguistic Term Set

For the screening purpose the Linguistic term set Based on the Ordered Structure is used. A set of seven terms of ordered structured linguistic terms is used here: following scale

$$S = \{S_0 = none, S_1 = very\ low, S_2 = low, S_3 = medium, S_4 = high, \\ S_5 = very\ high, S_6 = perfect\} \tag{22.20}$$

where $S_a < S_b if a < b$. The linguistic term set in addition satisfy the following conmditions:

$$Negation\ operator Neg\,(S_i) = S_j, j = T - i\,(T + 1 is\ cardinality)) \tag{22.21}$$

$$Maximisation\ operator Max\,(S_i, S_j) = S_i, if S_i \geq S_j \tag{22.22}$$

$$Minimisation\ operator Min\,(S_i, S_j) = S_i, if S_i \leq S_j \tag{22.23}$$

22.8.3 The Semantic of Linguistic Term Set

In this case the Semantic Based on the Ordered Structure is used. The terms are symmetrically distributed, i.e. it is assumed that linguistic term sets are distributed on a scale with an odd cardinal and the mid term representing an assessment of ”approximately 0.5” and the rest of the terms are placed symmetrically around it.

22.8.4 Aggregation Operator for Linguistic Weighted Information

In this case the Linguistic Weighted conjunction aggregation operator is used.

Table 22.5 Importance of different parameters used in screening proces

Parameter	Importance	Direction
Length of the stack	M	L
Losses	H	L
Mass	VH	L
Electrical time constant	H	L
Inertia of the rotor	L	L
Ratio of inner stator to outer stator diameters	H	H
Number of turns	M	L
Reminance field of permanent magnet	N	L
Max. field density in stator lamination material	N	L
Width of the tooth	VL	L
Width of the yoke	L	L
Input Voltage	H	L
Area of slot	H	H

22.8.5 The Screening Process

The importance of different parameters discussed in the previous section is shown in Table 22.5. The length of the motor stack is given medium importance and the smaller the length of the motor the better it is, i.e. a smaller stack length is preferred over the larger length. For the losses a high importance and the lower the losses the better. Similarly for the mass a very high importance is given and smaller the mass the more preferred is the motor. The electrical time constant of the motor is given a high importance and the lower value is better. Ratio of inner to outer stator diameter is given a higher value and higher the value the better it is. A medium importance is given to number of turns and lower the number of turns is preferred. The reminance field of permanent magnet and maximum allowable field density of stator lamination is given no importance. The width of the tooth and width of the yoke are given very low and low importance respectively and lower the values of both the parameters

Table 22.6 Parameters of the set of solutions after final screening

N_{turns}	L_{motor}	α_{dido}	α_m	B_r	B_{fe}	h_m	V_{dc}	F_{sw}	wt	wy	N_s	N_m
59	10.35	0.60	0.94	1.19	1.88	1.61	36.24	206.17	5.80	4.35	6	4
60	10.65	0.60	0.94	1.19	1.55	1.59	21.10	207.34	3.39	2.54	9	6
60	10.36	0.60	0.98	1.20	1.85	1.57	25.05	149.98	3.92	2.94	12	8
60	18.21	0.50	0.88	0.82	1.99	1.53	22.51	192.57	2.90	1.09	6	8
60	19.93	0.60	0.78	0.82	1.94	1.51	19.56	115.40	2.14	1.20	9	8
60	19.49	0.54	0.96	1.01	1.97	1.55	25.36	120.08	2.61	1.18	9	10

the better it is. The area of the slot is given a high importance and the higher value of the slot area is preferred. The results of the multicriteria decision for motors are given in Table 22.6.

22.9 Final Analysis Phase of PDM for Design of a BLDC Motor Drive

In this section detailed analysis of the motors obtained in the previous section is done. At this point it is important to obtain accurate value of induced voltage, cogging torque, torque profile etc. Hence, in this case formal (mathematical) deterministic and a high fidelity model of the BLDC motor drive is required. These deterministic and high fidelity models are developed using Finitie Element Models of the motor using FEMAG and smartFEM.

22.9.1 Detailed Simulation Model

The Final Analysis phase of PDM involves the tuning process. In the tuning process the system performance criteria are improved by varying system parameters. In order to achieve this, high fidelity model of the system that is to be designed is developed. Each alternative obtained after Intermediate Analysis phase is evaluated using the high fidelity model and tuning of the system is performed. The high fidelity models can be developed using finite element methods (FEM), computational fluid dynamic (CFD), etc. These models are computationally intensive but are closer to the actual system and are suitable for Final Analysis phase of PDM. The detailed model of the BLDC motors are developed using FEM packages FEMAG and smartFEM. The FEM model is able to calculate the cogging torque and shape of the back EMF accurately.

22.9.2 Independent Design Variables and Objectives

The independent design variable are length of the stack (L_{stack}), ratio of stator inner to outer diameter (α_{DiDo}), ratio of magnet angle to pole pitch (α_m) and B_{mag}. The new objectives are cogging torque and back emf values. The shape of the back emf and the magnitude of the cogging torque strongly depend on these variables [57], hence they are chosen as independent design variables for the final analysis.. It is required that the motor has a trapezoidal back emf and cogging torque less than 20mNm. The FEM model of the BLDC motor is run for different combinations of the independent design variables and the results are studied and the solution meeting the criteria (trapezoidal shape of back emf and cogging torque less than 20mNm) are selected.

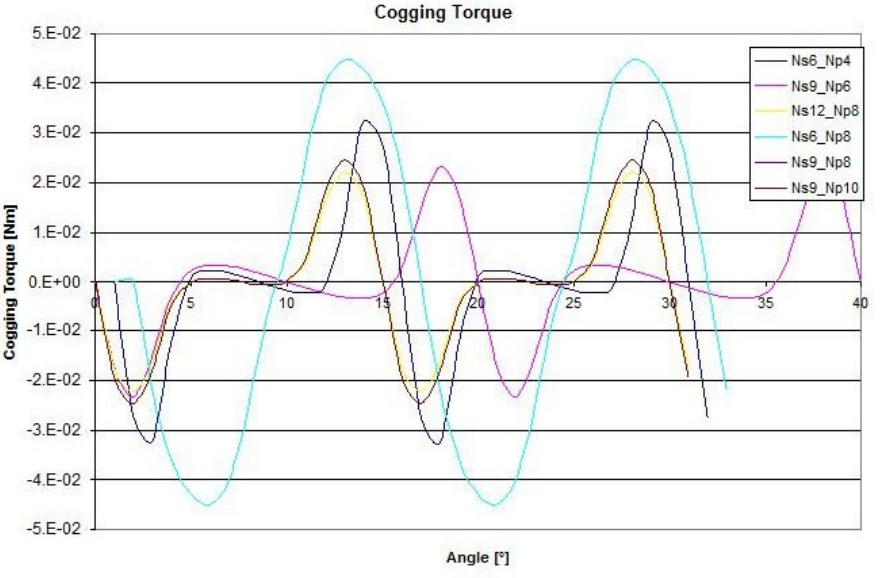


Fig. 22.18 Cogging torque waveform of motors

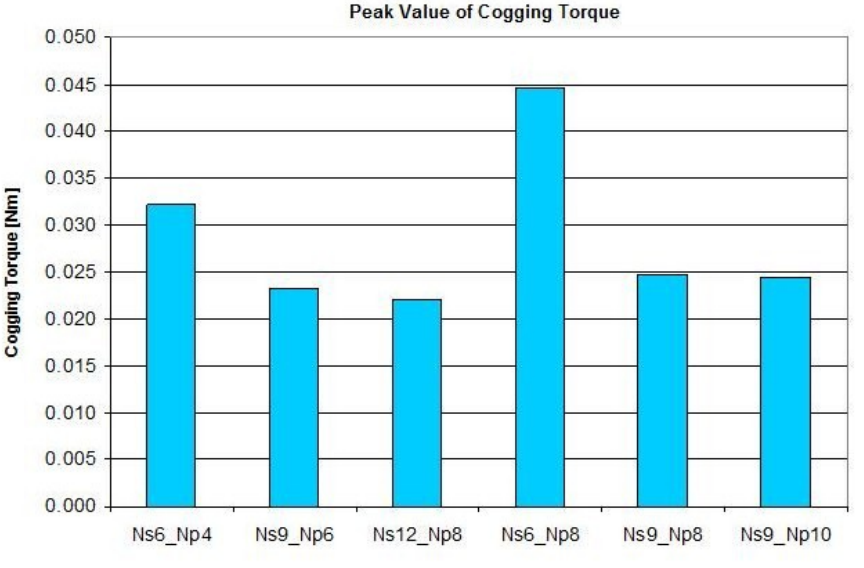


Fig. 22.19 Peak values of the cogging torque

22.9.3 Set of Solutions

The results of cogging torque for all the 6 alternatives in Table 22.6 are shown in Fig. 22.18 and the peak values of cogging torque are shown in Fig. 22.19 respectively. From the Fig. 22.18 and Fig. 22.19 it is seen that motor with 12 slots and 8 poles has the minimum cogging torque, hence this motor was considered for detailed analysis and its parameters were determined so as to meet all the required criteria. The geometric parameters of the motor were fine-tuned so as to obtain cogging torque less than 0.02Nm and a trapezoidal back emf. The final configuration of the motor is given in Table 22.7 below. Finally a prototype based on configuration given in Table 22.7 was made. The characteristics curves of the prototype are given in Fig. 22.20 to Fig. 22.23. From these figures it can be seen that the performance of the motor is close to the simulated values.

Table 22.7 Parameters of the motor after fine tuning

N_{turns}	L_{motor}	α_{dido}	α_m	B_r	B_{fe}	h_m	V_{dc}	F_{sw}	wt	wy	N_s	N_m
60	10	0.60	1	0.65	1.57	1.505	24	206.17	2.015	1.511	12	8

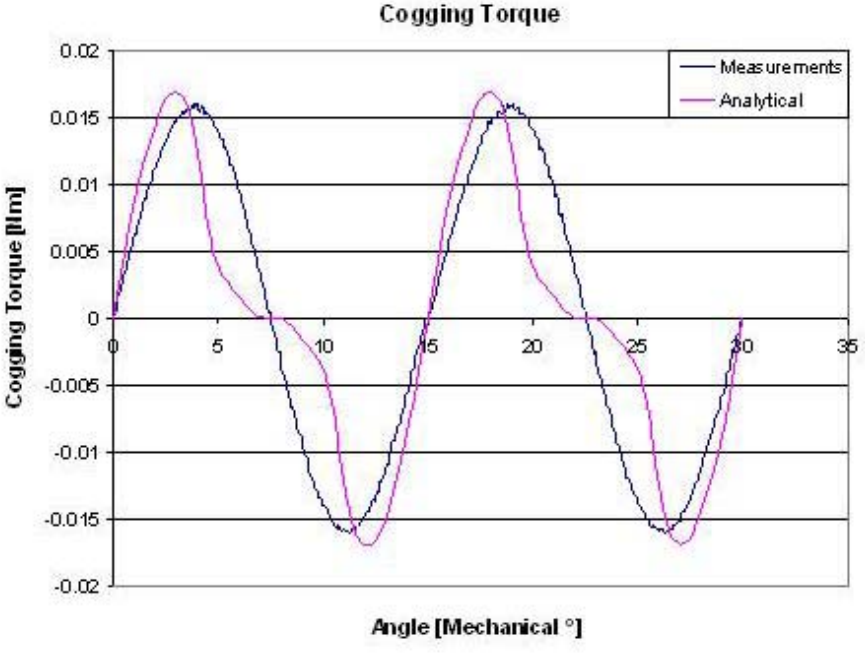


Fig. 22.20 Power vs. Speed characteristics: Comparison between simulation and experimental values

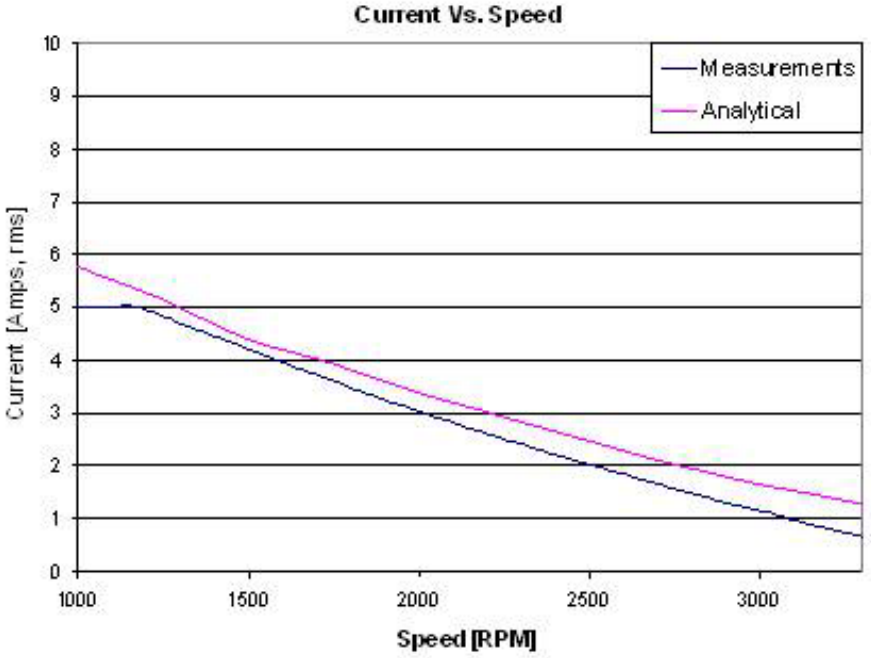


Fig. 22.21 Current vs. Speed Characteristic comparison between simulation and experimental values

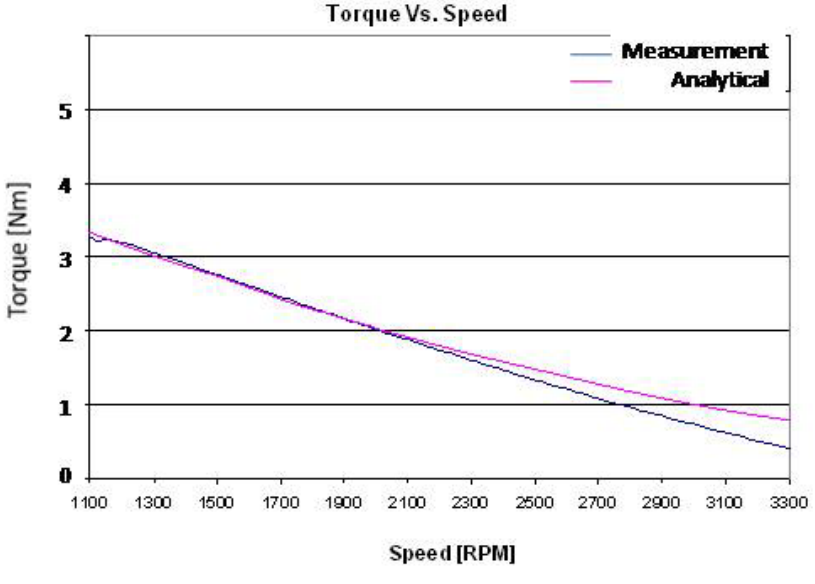


Fig. 22.22 Torque vs. Speed characteristics: Comparison between simulation and experimental values

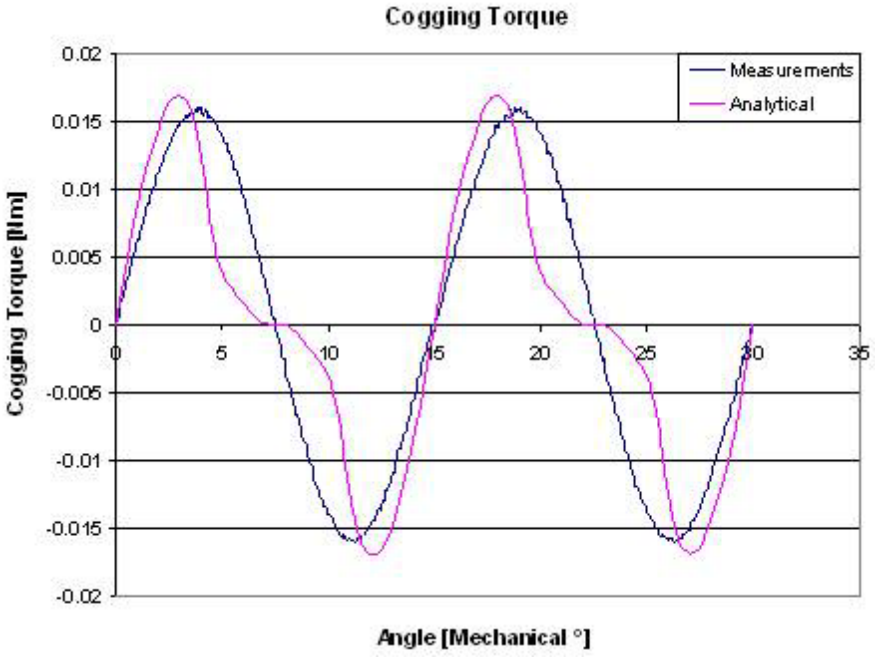


Fig. 22.23 Cogging torque comparison between simulations and experimental values

22.10 Conclusions

In this chapter the progressive design methodology (PDM) is proposed. This methodology is suitable for designing complex systems, such as electrical drive and power electronics, from conceptual stage to final design. The main aspects of PDM discussed are as follows:

- PDM allows effective and efficient practices and techniques to be used from the start of the project.
- PDM ensures that each component of the system is compatible with each other.
- The computation time required for optimisation is reduced as the bulk of optimisation is done in the synthesis phase and the models of the components of the target system are simple in the synthesis phase.
- The experience of design engineers and production engineers are included in the intermediate analysis thus ensuring that the target system is feasible to manufacture.

In PDM the decision making factor is critical as proper decisions about dimensions, features, materials, and performance in the conceptual stage will ensure a robust and optimal design of the system. The different stages of PDM are explained using the example of the design of a BLDC motor and the results are validated by

experiments. It is shown that using PDM an optimal design of the motor can be obtained that meets the performance requirements.

References

1. Balling, R.J., Sobieszczanski, J.S.: Optimization of coupled Systems: A Critical Overview of Approaches. AIAA 34, 6–17 (1996)
2. Lewis, K., Mistree, F.: Collaboration, Sequential and Isolated Decision Design. ASME Journal of Mechanical Design 120, 643–652 (1998)
3. Sobieszczanski, J.S., Haftka, R.T.: Multidisciplinary Design Optimization. Structural Optimization 14, 1–23 (1997)
4. Alexandrov, N.M., Lewis, R.M.: Analytical and Computational Aspects with Multidisciplinary Design. AIAA 40, 301–309 (2002)
5. Sobieszczanski, J.S.: Optimization by Decomposition: A Step from Hierarchic to Non-Hierarchic Systems. In: Second NASA/USAF Symposium on Recent Advances in Multidisciplinary Analysis and Optimization (1988)
6. Sobieszczanski, J.S., Agte, J.S., Sandusky, R.R.J.: Bilevel Integrated System Synthesis. AIAA Journal 38, 164–172 (2000)
7. Sobieszczanski, J.S., Altus, T.D., Philips, M., Sandusky, R.: Bilevel Integrated System Synthesis (BLISS) for Concurrent and Distributed Processing. In: 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization (2002)
8. Tappeta, R., Nagendra, S., Renand, J.E., Badhrinath, K.: Concurrent Sub-Space Optimization (CSSO) Code using iSIGHT. Technical Report 97CRDD188, GE (1998)
9. Marczyk, J.: Stochastic Multidisciplinary Improvement: Beyond Optimization. Presented at Proceedings of 8th AIAA/ USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long beach, USA (2000)
10. Egorov, N., Kretinin, G.V., Leshchenko, I.A.: Stochastic Optimization of Parameters and Control Laws of Aircraft Gas-Turbine Engines- a Step to a Robust Design. In: Inverse Problem in Engineering Mechanics III, pp. 345–353 (2002)
11. Koch, P.N., Wujek, B., Golovidov, O.: A Multi-Stage, Parallel Implementation of Probabilistic Design optimization in an MDO Framework. Presented at Proceeding of 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, USA (2000)
12. Tong, M.T., Wujek, B., Golovidov, O.: A Probabilstic Approach to Aeropropulsion System Assessment. Presented at Proceedings of ASME TURBOEXPO, Munich, Germany (2000)
13. Booker, A., Dennis, J., Frank, P., Serafini, D., Torczon, V., Trosset, M.: A Rigorous Framework for Optimization of Expensive Functions by Surrogates. Structural Optimization 17, 1–13 (1999)
14. Audet, C., Dennis, J., Moore, D.W., Booker, A., Frank, P.D.: A surrogate model based method for constrained optimization. Presented at Proceedings of the 8th AIAA/USAF/NASA/ASSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, USA (2000)
15. Audet, C., Dennis, J.: A pattern search filter method for nonlinear programming without derivatives. SIAM Journal of Optimization 14, 980–1010 (2004)
16. Dasgupta, S.: The Structure of Design Processes. Advances in Computers 28, 1–67 (1989)

17. Shakeri, C.: Discovery of Design Methodologies for the Integration of Multi-disciplinary Design Problems. Mechanical Engineering: Worcester Polytechnic Institute (1998)
18. Systems Engineering Manual Version 3.1, The Federal Aviation Administration
19. Chong, E.P.K., Zak, S.H.: An Introduction to Optimization, 2nd edn. John Wiley & Sons, Chichester (2001)
20. Buede, D.M.: The Engineering Design of Systems: Models and Methods. Wiley Interscience, Hoboken (1999)
21. Hwang, S.P.C., Yoon, K.: Mathematical Programming With Multiple Objectives: A Tutorial. Computers & Operations Research 7, 5–31 (1980)
22. Steuer, R.: Multiple Criteria Optimization: Theory, Computation and Application. John Wiley & Sons, Chichester (1986)
23. Das, I., Dennis, J.E.: A close look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multi-criteria optimization problems. Structural Optimization 14, 63–69 (1997)
24. Krus, P., Palmberg, J.O., Löhr, F., Backlund, G.: The Impact of Computational Performance on Optimisation in Aircraft Design. Presented at I MECH E, AEROTECH 1995, Birmingham, UK (1995)
25. Thurston, D.: A formal method for subjective design evaluation with multiple attributes. Research in Engineering Design 3, 105–122 (1991)
26. Steuer, R., Choo, E.-U.: An Interactive Weighted Tchebycheff Procedure for Multiple Objective Programming. Mathematical Programming 26, 326–344 (1983)
27. Benayon, R., Montgolfier, J.D., Tergny, J., Laritchev, O.: Linear Programming with multiple objective functions: Step method (STEM). Mathematical Programming, 366–375 (1971)
28. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist nondominated sorting genetic algorithm for multi-objective optimization. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
29. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary alorithm (2001)
30. Kumar, P., Gospodaric, D., Bauer, P.: Improved Genetic Algorithm Inspired by Biological Evolution. Soft Computing- A Fusion of Foundations, Methodologies and Applications 11, 923–941 (2006)
31. Scott, M.J., Antonsson, E.K.: Arrow's Theorem and Engineering Design Decision Making. Research in Engineering Design 11, 218–228 (1999)
32. Costa, B., Vincke, P.: Multiple criteria decision aid: An overview. Readings in Multiple Criteria Decision Aid, 3–14 (1990)
33. Carlsson, C., Fuller, R.: Fuzzy Multiple Criteria Decision Making: Recent Developments. 78, 139–153 (1996)
34. Riberio, R.A.: Fuzzy Multiple Attribute Decision Making: A Review and New Preference elicitation Techniques. Fuzzy Sets and Systems 78, 155–181 (1996)
35. Roubens, M.: Fuzzy Sets and Decision Analysis. Fuzzy Sets and Systems 90, 199–206 (1997)
36. Matinez, L., Liu, J., et al.: A Fuzzy Model for Design Evaluation Based on Multiple-Criteria Analysis in Engineering Systems. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 14, 317–336 (2006)
37. Whitney, D.E.: Manufacturing by Design. Harvard Business Review 66, 83–91 (1988)
38. Zadeh, L.A.: The concept of a linguistic variable and its applications to approximate reasoning. Part III. Information Sciences 9, 43–80 (1975)

39. Zadeh, L.A.: The concept of a linguistic variable and its applications to approximate reasoning. Part I. Information Sciences 8, 301–357 (1975)
40. Zadeh, L.A.: The concept of a linguistic variable and its applications to approximate reasoning. Part II. Information Sciences 8, 357–377 (1975)
41. Bordogna, G., Passi, G.: A Fuzzy Linguistic Approach Generalizing Boolean Information Retrival: A Model and its Evaluation. Journal of the American Society for Information Science 44, 70–82 (1993)
42. Bonissone, P.P.: A Fuzzy Sets Based Linguistic Approach. Approximate Reasoning in Decision Analysis, 329–339 (1986)
43. Bonissone, P.P., Decker, K.S.: Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-off Precision and Complexity. In: Uncertainty in Artificial Intelligence, pp. 217–247 (1986)
44. Bordogna, G., Fedrizzi, M., Passi, G.: A Linguistic Modelling of Consensus in Group Decision Making Based on OWA Operators. IEEE Transactions on System, Man, and Cybernatics - Part A: Systems and Humans 27, 126–132 (1997)
45. Delago, M., Verdegay, J.L., Vila, M.A.: Linguistic Decision Making Models. International Journal of Intelligent Systems 7, 479–492 (1992)
46. Herrera, F., Herrera-Viedma, E.: Linguistic Decision Analysis: Steps for Solving Decision Problems under Linguistic Information. Fuzzy Sets and Systems 115, 67–82 (2000)
47. Torra, V.: Negation Functions Based Semantics for Ordered Linguistic Labls. International Journal of Intelligent Systems 11, 975–988 (1996)
48. Herrera, F., Verdegay, J.L.: A Linguistic Decision Process in Group Decision Making. Group Decision and Negotiation 5, 165–176 (1996)
49. Gabrish, M., Murofushi, T., Sugeno, M.: Fuzzy Measures and Integrals. Physica-Verlag, Heidelberg (1999)
50. Dubious, D., Prade, H.: On the use of aggregation operations in information fusion process. Fuzzy Sets and Systems 142, 143–161 (2004)
51. Klement, E.P., Mesiar, R., Pap, E.: Triangular Norms. Kluwer Academic Publishers, Dordrecht (2000)
52. Sivert, W.: A Class of Operations Fuzzy Sets. IEEE Transactions on System, Man, and Cybernatics - Part A: Systems and Humans 9 (1979)
53. Calvo, T., Baets, B.D., Fodor, J.: The functional equations of Alsina and Frank for uniforms and null-norms. Fuzzy Sets and Systems 120, 15–24 (2001)
54. Yager, R., Fodor, J.: Structure of Uninorms. Journal of Uncertainity, Fuzziness and Knowledge based Systems 5, 411–427 (1997)
55. Herrera, F., Herrera-Viedma, E.: Aggregation Operators for Linguistic Weighted Information. IEEE Transactions on Systems, Man and Cybernatics 27, 646–656 (1997)
56. Carlsson, C., Fuller, R.: On fuzzy screening systems. Presented at EUFIT 1995, Aachen, Germany (1995)
57. Hanselmann, D.C.: Brushless Permanent Magnet Motor Design, 2nd edn. The Writer's Collective (2003)
58. Nucera, R.R., Sundhoff, S.D., Krause, P.C.: Computation of Steady State Performance of an Electronically Commutated Motor. IEEE Transactions on Industry Application 25, 110–111 (1989)

Chapter 23
Reliable Network Design Using Hybrid Genetic Algorithm Based on Multi-Ring Encoding

Jin-Myung Won, Alice Malisia, and Fakhreddine Karray

Abstract. Designing an inexpensive but reliable network is an important problem in many engineering applications such as transportation, telecommunication, and computer networks. This problem belongs to the class of NP-hard problems and lots of research works have been performed to develop a practical and efficient heuristic algorithm. This chapter surveys up-to-date research efforts to address the reliable network design problem and proposes a new hybrid heuristic of the genetic algorithm and local search ant colony system. The proposed hybrid heuristic represents a two-edge-connected candidate network as multiple rings. The genetic algorithm evolves a population of multi-ring-encoded individuals with special genetic operators while the local search ant colony system fine-tunes each ring. The results of computer experiments show the effectiveness and efficiency of the proposed hybrid heuristic.

23.1 Reliable Network Design

It is an important problem to design the topology of reliable networks that remains connected under the failure of network components. This problem so called a Reliable Network Design Problem (RNDP[1]) has many applications in the area of transportation, utility, telecommunication, and computer networks [1]. The general objective of the RNDP is to find the minimum-cost sub-network of a given undirected network that satisfies a prescribed level of reliability and other problem-specific constraints. Difficulties in handling the RNDP arise from its huge search space. In

Jin-Myung Won
Vestec Inc., 145 Columbia Street West Suite 1, Waterloo, Ontario, Canada N2L 3L2
e-mail: jinmyung.won@gmail.com

Alice Malisia
Department of Electrical and Computer Engineering, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1
e-mail: armalisi@engmail.uwaterloo.ca

Fakhreddine Karray
Department of Electrical and Computer Engineering, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada N2L 3G1
e-mail: karray@uwaterloo.ca

[1] The singular and plural of abbreviations are spelled the same in this chapter.

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 609–635.
springerlink.com

fact, the exact evaluation of network reliability belongs to the class of NP-hard problems [2]; that is, these exists no polynomial-time algorithm to calculate the reliability of a given network exactly. Moreover, even if we know the reliability of every candidate network, the RNDP itself cannot be solved exactly in a polynomial time.

To address realistic RNDP, many algorithms have been studied and proposed in the literature. They can be classified into three categories: (i) enumeration-based approaches, (ii) heuristic approaches, and (iii) computational intelligence. The enumeration-based approaches attempt to evaluate all the possible candidate solutions to find the best one. To avoid exhaustive enumeration, reduction techniques such as branch and bound methods [1] should be involved. The reduction technique restructures the search space so that scanning the limited portion of the search space could yield the optimal solution. It has been shown that even with a well-designed reduction technique, the enumeration-based approaches are applicable for small networks only.

The heuristic approaches find a sub-optimal solution of the RNDP by exploring the search space using problem specific trial and error mechanism. The heuristic approaches never guarantee the discovering of the optimal solution, but they are practical choices to find a satisfactory solution in an acceptable time. Typical examples of the classical heuristic-based approaches are greedy heuristics [3], cross-entropy methods [4], simulated annealing [5, 6, 7], and tabu search [8, 9].

Algorithms based on computational intelligence can be regarded as a branch of heuristic approaches. However, computational intelligence simulates distinctive search process based on learning, adaptation, and evolution mechanism. A Genetic Algorithm (GA) is the most widely used optimization technique based on computational intelligence [10, 11]. Mimicking natural evolution process, the GA maintains a population of candidate solutions by applying selection, crossover, and mutation operators iteratively. The goal of the GA process is adapting the population to the fitness landscape of the RNDP to find a good sub-optimal solution. More recently, it has been often attempted to hybridize the GA with problem-specific local search algorithms to achieve better solution quality [12]. Such hybrid GA is called genetic local search or memetic algorithms [13, 14].

This chapter surveys up-to-date research efforts for the RNDP and proposes a new GA hybridized with an Ant Colony System (ACS). The ACS is a heuristic inspired by the behavior of real ants, which establish the shortest path between the nest and food source [15, 16]. To combine the GA and ACS, the proposed heuristic algorithm incorporates a Multi-Ring Encoding (MRE), which encodes a candidate network as a union of rings. The MRE has three distinctive advantages for the RNDP. First, it can represent every possible two-edge-connected network. Second, it is free from expensive algorithms required to repair disconnected or unreliable candidate networks generated by the GA. Third, the MRE allows incorporating a local search heuristic dedicated to ring optimization. In the proposed hybrid heuristic, the GA works as a high-level heuristic evolving a population of multi-ring-encoded individuals with special genetic operators. On the other side, the ACS fine-tunes each ring by trying to connect the nodes in other possible orders.

This chapter is organized as follows. Section 23.2 describes the mathematical formulation of the RNDP and suggests the ways handling two objectives,

cost and reliability. Section 23.3 classifies reliability metrics and introduces the issues regarding reliability evaluation and estimation. Section 23.4 outlines previous works related to this study. Section 23.5 reviews existing encoding methods developed for the network design problem and discusses the advantages of the proposed MRE. Section 23.6 explains each procedure of the proposed hybrid heuristic in detail. Section 23.7 discusses numerical results comparing the proposed hybrid heuristic to existing exact algorithm and genetic local search.

23.2 Problem Formulation

The RNDP is a combinatorial optimization problem to find a sub-network of a given undirected network that satisfies given cost, reliability, and performance criteria. Let $G = (N, E)$ be the given complete undirected network, where N is a set of n nodes representing communication stations and E is a set of $e := n(n-1)/2$ edges representing communication links. The network G has neither self-loops nor redundant edges. Two distinct nodes in G are connected if there exist at least one path between them. A network is connected if every pair of two nodes is connected.

To represent a sub-network of G, denote $x_k \in \{0,1\}$ for $k \in \{1,\ldots,e\}$ by a binary decision variable indicating if the k-th edge is purchased. Then, a binary vector representing one of 2^e possible candidate solutions can be defined as

$$\mathbf{x} := (x_1,\ldots,x_e)' \in \{0,1\}^e. \tag{23.1}$$

Let $C(\mathbf{x})$ and $R(\mathbf{x})$ be the cost and reliability of the candidate solution $\mathbf{x}$. Then, the objective of the basic RNDP is to find $\mathbf{x}^*$ that forms a connected sub-network of G such that $C(\mathbf{x})$ is minimized while $R(\mathbf{x})$ is maximized. If other performance criteria like capacity or transmission delay are specified, they act as additional constraints of the RNDP.

It is clear that the cost and reliability are conflicting objectives. Adding edges to a certain network will make it more reliable but more expensive and vice versa. This implies that the RNDP can be regarded as a bi-objective optimization problem having multiple solutions, where the improvement in cost sacrifices the reliability. Such solutions are called *Pareto optimal solutions* [17]. Many research works have been performed to establish the theory and applications of the multi-objective optimization problem. In particular, the GA has enjoyed great success in addressing multi-objective optimization problems. The population-based search paradigm of the GA provides a simple but efficient way to approximate the Pareto optimal solutions from the single GA run [18].

A simple way to make the RNDP single-objective is to consider either objective as a constraint. This study assumes that the minimum reliability requirement $R_{\min}$ is predetermined and $C(\mathbf{x})$ should be minimized accordingly. Hence, the RNDP is formulated as:

Given:	G and $R_{\min} \in (0,1)$
Over:	$\mathbf{x}$
Minimize:	$C(\mathbf{x})$
Subject to:	$R(\mathbf{x}) \geq R_{\min}$.

Table 23.1 Relationship between network size and search space size of RNDP

Network size (n)	Search space size ($2^{n(n-1)/2}$)	Number of spanning trees (n^{n-2})
5	1024	125
10	3.52×10^{13}	1.00×10^{8}
20	1.57×10^{57}	2.62×10^{23}

With a slight modification, the proposed hybrid heuristic can be applied to the opposite case, where the available budget is prescribed and $R(\mathbf{x})$ should be maximized.

The cost of the edge k is given as $c_k \in \Re_+$ and $C(\mathbf{x})$ is the summation of the total edge cost:

$$C(\mathbf{x}) = \sum_{k=1}^{e} c_k x_k.$$

This study assumes that nodes are invulnerable (perfectly reliable) but the edge k may fail with a probability of $q_k \in [0,1]$. The operating probability p_k, is hence $1 - q_k$. The probabilistic metric for all-terminal reliability is used as $R(\mathbf{x})$. Refer to the next section to see how $R(\mathbf{x})$ is evaluated or estimated from $\mathbf{x}$ and p_k.

The RNDP belongs to the class of NP-hard problems. As mentioned above, the exhaustive search space of the RNDP has a cardinality of 2^e. This grows faster than exponentially as the network size n increases. Table 23.1 shows the relationship between the number of nodes and the associated search space size of the RNDP. Even for a small network with $n = 10$, it is impractical to check the cost and reliability of every candidate solution to pick up the best one. Instead, we should rely on heuristic methods, which yield good sub-optimal solutions in an acceptable time.

23.3 Network Reliability

The RNDP heuristics require either exact evaluation or rough estimation of network reliability. Since the exact evaluation of network reliability is NP-hard, relying on the estimation technique is unavoidable. The proper choice of the reliability estimation method is a crucial factor that decides the performance of RNDP heuristics. This section mathematically defines network reliability metrics and introduces several estimation methods popularly used.

23.3.1 Reliability Metrics

Network reliability metrics can be categorized into *deterministic metrics* and *probabilistic metrics* [19]. The deterministic metrics represent the number of network components whose failure disconnects the network. The deterministic metrics sometimes give inadequate reliability measure since the operating probability of network components is not considered. On the other hand, the probabilistic metrics indicate how probably the network will remain connected for given operating probabilities of network components. The reliability metrics can be further classified into two

categories based on the number of interested nodes: *All-Terminal Reliability* (ATR) and *Two-Terminal Reliability* (TTR). Under the all-terminal case, all the node pairs should be connected. The two-terminal case predefines the source node and sink node that should be connected.

To give formal definition of the reliability metrics, we need to introduce some definitions regarding network connectivity. Let $G(\mathbf{x})$ be a candidate network of the RNDP represented by a decision vector $\mathbf{x}$. A set of edges in $G(\mathbf{x})$ whose failure disconnects the network is defined as an *edge cutset*. The minimum edge cutset is called a *prime edge cutset* and the minimum cardinality of the edge cutset is entitled *edge connectivity*, which can be used as a deterministic metric for ATR. Analogous definitions can be made for TTR. For example, *two-terminal edge connectivity* is the minimum number of edges whose failure disconnects the source node from the sink node.

The RNDP considered in this chapter adopts the probabilistic metric for ATR as the reliability measure $R(\mathbf{x})$, which is termed ATR in the sequel for simplicity's sake. To compute $R(\mathbf{x})$ for a general $G(\mathbf{x})$, all the edge operating scenarios of $G(\mathbf{x})$ should be enumerated. Denote $e(\mathbf{x})$ by the set of edges in $G(\mathbf{x})$. Then, $G(\mathbf{x})$ has $2^{e(\mathbf{x})}$ edge operating scenarios, which can be grouped into connected or disconnected scenarios. If the failing edges in an edge operating scenario form an edge cutset, $G(\mathbf{x})$ becomes disconnected; otherwise, $G(\mathbf{x})$ remains connected. Aggregating the probability of every connected edge operating scenario of $G(\mathbf{x})$ yields $R(\mathbf{x})$.

If $G(\mathbf{x})$ has a special layout composed of a limited number of edges, its ATR is easily computable. The exact ATR computation is NP-hard though [20]. For a general $G(\mathbf{x})$, there exists no polynomial-time algorithm to group $2^{e(\mathbf{x})}$ edge operating scenarios into connected and disconnected cases.

23.3.2 Reliability Estimation

Given the NP-hard nature of ATR computation, we should rely on approximation techniques such as bound-based approaches and Monte Carlo Simulation (MCS) methods. The bound-based approach uses the upper or lower bound of the ATR as a reliability estimate. Good examples of such bounds are Jan's upper bound [1] and Lomonosov's lower bound [21]. This section describes how Jan's upper bound can be derived.

Let d_i for $i \in N$ be the degree of node i, which is the number of edges incident to i. Denote μ_i by the set of edges connected to node i. Then, μ_i is an edge cutset of $G(\mathbf{x})$. Let F_i be the event that all the edges in μ_i fail and F_i^c the complement of F_i. Since the network failure probability, $1 - R(\mathbf{x})$, is no less than $\Pr[F_1 \cup \cdots \cup F_n]$, we have the following inequality describing an upper bound of $R(\mathbf{x})$:

$$R(\mathbf{x}) \leq 1 - \Pr[F_1] - \Pr[F_2 \cap F_1^c] - \cdots - \Pr[F_n \cap F_1^c \cap F_2^c \cap \cdots \cap F_{n-1}^c]. \quad (23.2)$$

For the sake of simplicity, suppose again all the edges have the same failure probability of q. Then, we have $\Pr[F_1] = q^{d_1}$. For the other terms on the right-hand side of (23.2), we have

$$\Pr[F_i \cap F_1^c \cap F_2^c \cap \cdots \cap F_{i-1}^c] \geq q^{d_i} \prod_{j=1}^{i-1} (1 - q^{d_j - 1})$$

for $i \in \{2,\ldots,n\}$ as $G(\mathbf{x})$ may have an edge between node i and $j \in \{1,\ldots,i-1\}$. Given d_i, we may have a tighter lower bound as follows:

$$\Pr[F_i \cap F_1^c \cap F_2^c \cap \cdots \cap F_{i-1}^c] \geq q^{d_i} \prod_{j=1}^{\min(d_i,i-1)} (1-q^{d_j-1}) \prod_{j=\min(d_i+1,i)}^{i-1} (1-q^{d_j}) \quad (23.3)$$

From (23.2) and (23.3), we have Jan's upper bound:

$$R(\mathbf{x}) \leq 1 - \sum_{i=1}^{n} \left\{ q^{d_i} \prod_{j=1}^{\min(d_i,i-1)} (1-q^{d_j-1}) \prod_{j=\min(d_i+1,i)}^{i-1} (1-q^{d_j}) \right\}. \quad (23.4)$$

This upper bound can be calculated in a polynomial time of n, but sometimes has a considerable error from the actual ATR.

Another simple but intuitive estimation method is the MCS. Given $G(\mathbf{x}) = (N, E(\mathbf{x}))$ and p_k for $k \in E(\mathbf{x})$, the MCS randomly generates a number of edge operating scenarios and checks if the nodes remain connected under each scenario. The edge operating scenario is generated by sampling a uniform random number $u \in [0,1]$ for each edge k; if this number is greater than p_k, the edge is removed. For every scenario, a connectivity test such as breadth-first search is performed. The ratio of the operating scenarios that maintain the node connectivity becomes the reliability measure of $G(\mathbf{x})$.

As the iteration number of MCS grows, more precise reliability estimation can be expected. The computational load of the MCS for precise estimation grows slightly faster than linearly with network size, but this is much heavier than the bound-based method. For this reason, the bound-based method is generally used for the screening purpose of unreliable candidate networks while the MCS is applied to obtain more precise reliability estimates of the low-cost candidate solutions having high ATR bounds.

Under certain conditions making reliability evaluation straightforward, the RNDP can be easily solved even for a big n. For example, if all the e edges in E have identical operating probability p and $R_{\min} \leq p^{n-1}$, the RNDP is reduced to the minimum spanning tree problem whose objective is to find the shortest-length spanning tree to connect all the nodes in N. Greedy heuristics such as Prim's algorithm [22] can solve the minimum spanning tree problem in a polynomial time. If $R_{\min} > p^{n-1}$ and $R_{\min} \leq p^n + np^{n-1}(1-p)$, the RNDP becomes the Travelling Salesman Problem (TSP) to find out the shortest-length Hamiltonian cycle visiting all the nodes in N [23]. The TSP is NP-hard and thus exact algorithms work only for a small network. However, many approximation algorithms have been proposed so far to find a good sub-optimal solution in an acceptable time. Once the entire edges in E have the identical cost, theoretical clues obtained from previous works can help us to find a good solution of the RNDP even when $R_{\min}$ is large. For example, [19] showed that the network with a largest number of trees maximizes the ATR when p is higher. However, these kind of exact methods cannot cope with realistic RNDP whose $R_{\min}$ is big and c_k and p_k are different for a different $k \in E$. The heuristic methods surveyed in the next section are practical approaches for the realistic RNDP.

23.4 Previous Works

Modern heuristic methods such as the GA and ACS have enjoyed great success in solving various network designing and routing problems. This section briefly describes the GA and ACS used for network optimization problems and outlines previous works that tried to hybridize the two algorithms.

23.4.1 Genetic Algorithm

The GA is a heuristic based on the computer-simulated natural genetic system. Since proposed by Holland in the early 1970's [24], the GA has been popularly used to solve expensive combinatorial optimization problems for which no satisfactory problem-specific heuristic exists. The GA maintains a population of individuals, which act as abstract representation of candidate solutions. The general format of the abstract representation is a binary string, but other formats can be used. The GA initializes the individuals as random strings and enters a generational loop. At every generation, the GA evaluates the fitness of every individual in the current population, selects multiple individuals based on their fitness, and evolves them with crossover and mutation operators to form a new population for the next generation. The generational loop continues until the termination condition is satisfied. The termination condition can be either the number of generations or the level of fitness that should be reached by the best individual.

The GA has successfully tackled various network design problems. For example, Kumar et al. [25] developed a GA to design a distributed system topology considering graph diameter and exactly calculated network reliability. Pierre and Legault [26] used a GA to generate the minimum-cost distributed packet switch networks subject to delay and reliability constraints. Chou et al. [27] designed a GA and examined the relationship between the GA parameters and performance. Gen et al. [11] surveyed research works on the GA for the network design problems such as a fixed charge transportation problem. Marseguerra et al. [28] assumed that network components have uncertainty in reliability and proposed a multi-objective genetic algorithm to find the Pareto optimal solutions maximizing the expected network reliability while minimizing its variance.

23.4.2 Ant Colony Optimization

The ACS is a kind of Ant Colony Optimization (ACO) technique that was firstly proposed by Dorigo in the early 1990's [16]. Inspired by the behavior of real ants, the ant algorithm sends out multiple artificial ants, which try to find the shortest path from the source node to the destination node. Each artificial ant deposits pheromone on its path to the destination node. The next ant can smell the pheromone and prefers the path with high pheromone concentration. In this manner, the search history obtained from the predecessor ants is propagated to the successors. It was experimentally shown that this group behavior of ants reveals a good path between the source node and destination node.

Although the ACO has broadened its application area to various engineering problems such as routing, quadratic assignment, and scheduling, it especially outperforms other heuristic algorithms in dealing with the TSP. It has been rarely attempted to apply the ACO technique to general network design problems [15]. This is because an ant itself is incapable of building a generic network, where a node may have three or more adjacent edges. An intuitive way to utilize the ACO for the network design problems is to make it generate an initial Hamiltonian cycle and then augment the cycle by adding edges until the given network constraints are satisfied. This approach is not recommended since it cannot synthesize every possible connected network. As a part of network design heuristics, however, the ACO may play an important role of building the minimum-cost path or cycle. From this motivation, we propose the hybrid heuristic in Section 23.6.

23.4.3 Hybrid Heuristics

The hybridization of a GA and ACO has been an active research area. Despite the GA and ACO being successfully applied to various real-world optimization problems, researchers have been combining the two techniques to achieve better solution quality. For instance, one particular variant of the ACO, named a best-worst ant system, incorporated pheromone mutation based on the concepts from evolutionary computation [29]. Similarly, Poorzahedy combined a GA and an ant system to improve the performance of the ant system for the network design problem [12]. The GA is used to mutate the pheromone matrix, but the algorithm remains mostly an ant algorithm. Gong and Ruan attempted to address a TSP by integrating a GA and ACO [30]. In their implementation, an ant mapped to each chromosome undergoes special genetic operators based on linkage-based operation. Pilat and White proposed two other hybrid approaches that focus on parameter optimization of the ant algorithm [31].

In other hybrid implementations, the algorithms work together by sharing successful candidate solutions. Tseng proposed an algorithm, where a GA and ACO work in parallel on the quadratic assignment problem [32]. If the GA produces a better solution than the one found by the ACO, the pheromone is updated with the GA solution. The pheromone matrix is used as the link between the GA and ACO. In another study, Acan developed an algorithm to combine the benefits of a GA and ACO [33]. The two algorithms work in parallel, but as soon as one of the algorithms finds an improved solution, the solutions are migrated to the other algorithm. A common GA–ACO hybridization approach is to use the ant algorithm to generate the population for the GA [34, 35, 36]. This is an intuitive form of hybridization, but requires the assumption that the ACO can generate every possible candidate solution, which is not true for the RNDP.

Another popular hybridization strategy is using the algorithms to optimize different parts of the problems. To solve the concrete delivery problem, Silva et al. introduced a method, where hybridization is done sequentially [37]. The GA is the main algorithm and the ACO is used to solve a sub-problem. Clearly, this is a flexible approach as one can use the best algorithm to optimize a particular

sub-problem. Likewise, Li et al. hybridized the ACO and GA to handle the task mapping in multi-core based system, where task scheduling and task assignment problems are combined [38].

23.5 Solution Representation

This section describes the MRE, which represents a two-edge-connected network as a union of rings. The MRE plays a key role in the proposed hybrid heuristic as it enables to hybridize the ACS as the local search heuristic of the GA. This section also explains how the MRE could help to improve the performance of the MCS for reliability estimate.

23.5.1 Multi-Ring Encoding

The encoding of a candidate solution is an important factor that may determine the performance of a GA. A good encoding method should be able to represent all the possible candidate solutions. One-to-one mapping between the encoded string and candidate solution is also desirable. Moreover, it should have high locality, which makes a small change in the encoded string result in a small change in the candidate solution [27]. This guarantees the standard crossover operator and mutation operator of the GA work properly.

Popular encoding techniques developed for the network design problem are edge representation [26], Prüfer encoding [39], predecessor encoding [40], determinant encoding [41], random keys [42], and weighted encoding [43]. The edge encoding belongs to *direct encoding*, which represents a candidate network as a string whose element indicates the presence of a certain edge in the candidate network. The direct encoding is intuitive and exhibits high locality, but mostly generates disconnected networks via standard genetic operators. This means the GA working with the direct encoding requires extra algorithms to repair disconnected networks as in [10]. *Indirect encoding* such as Prüfer encoding employs a special string structure that needs to be decoded to produce specific types of network topology such as minimum spanning trees or ring-star topologies [44, 45]. The indirect encoding largely suffers from low locality and does not work well with genetic operators.

To address the RNDP efficiently, this study uses the MRE [46, 47], which represents a network as a union of rings (i.e., cycles) traversing three or more nodes. The distinct advantage of the MRE is that it only represents the networks with the edge connectivity of two, which is a necessary condition for a reliable network. In the same context, we need not incorporate an additional algorithm to repair the connectivity or reliability of infeasible candidate solutions, which may be generated during search process. Another important merit of the MRE is that it enables to hybridize the GA with the ACS, which acts as a local search heuristic specialized for ring optimization. The MRE has high locality [48] and works fine with evolutionary operators specially designed for the RNDP. Existing search operators developed for permutation-based encoding are directly applicable to the MRE. Examples of

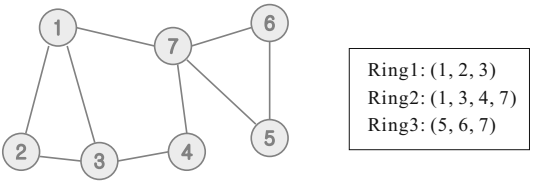


Fig. 23.1 Example of multi-ring encoding

such operators are the nearest neighbor algorithm, Clarke-Wright algorithm [49], Christofides algorithm [50], 2-Opt [51], 3-Opt [52], double-bridge move [53], and Lin-Kernighan algorithm [54].

Under the MRE, a two-edge-connected candidate network can be represented by a set of rings $\mathbf{Y} = \{\mathbf{r}_1, \ldots, \mathbf{r}_L\}$, where $\mathbf{r}_l$ for $l \in \{1, \ldots, L\}$ is a simple ring visiting m_l distinct nodes in N. The candidate network is formed by taking the union of all the edges composing $\mathbf{r}_1, \ldots, \mathbf{r}_l$. For example, Fig. 23.1 illustrates a seven-node network represented by three rings. Define a node whose degree is three or more as a *bridge node* of a candidate network. Under the MRE, the bridge node acts as a junction point of two or more rings. The network in Fig. 23.1 has three bridge nodes 1, 3, and 7.

23.5.2 Contraction Model

The contraction model of a candidate network is obtained by modelling an edge or a chain of edges connecting two bridge nodes as a *contracted edge*. With the contraction model, we can evaluate the ATR of the candidate network more precisely with a less computation time. Since the reliability evaluation is the main computational bottleneck of population-based heuristics, using the contraction model can significantly reduce the computation time of the GA.

Given a candidate network $G(\mathbf{x})$, we can establish its contraction model $\bar{G}(\mathbf{x}) = (\bar{N}, \bar{E})$ by going through the following steps:

1. Count the degree of every node in $G(\mathbf{x})$ to decide bridge node set $\bar{N}$.
2. Add a single edge connecting two bridge nodes to $\bar{E}$.
3. Add a chain of edges connecting two bridge nodes to $\bar{E}$.

Note that $\bar{G}(\mathbf{x})$ may have self-loops or redundant contracted edges. For example, the network depicted in Fig. 23.1 has a contraction model illustrated in Fig. 23.2. Node 1, 3, and 7 become the member of $\bar{N}$ and five contracted edges are generated accordingly.

A contracted edge may have three states:

1. Connected: All the edges in the contracted edge are operating.
2. Edge-Disconnected: One edge fails but others are operating.
3. Node-Disconnected: Two or more edges fail.

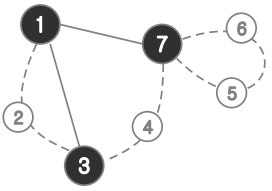


Fig. 23.2 Contraction model of the network depicted in Fig. 23.1 (black circles: bridge nodes; white circles: non-bridge nodes; solid lines: contracted edges composed of a single edge; dashed lines: contracted edges composed of two or more edges)

Table 23.2 List of contracted edges in Fig. 23.2

k	Nodes connected	$\bar{p}_k$	$\bar{q}_k$	$\bar{r}_k$
1	1, 3	0.95	0.05	0
2	1, 7	0.95	0.05	0
3	1, 2, 3	0.9025	0.095	0.0025
4	3, 4, 7	0.9025	0.095	0.0025
5	5, 6, 7	0.857375	0.135375	0.00725

If a contracted edge k connecting two bridge nodes i and j is in the edge-disconnected state, k is not operating, but non-bridge nodes traversed by k is connected to either i or j. If a contracted edge is in the node-disconnected state, at least one non-bridge node is disconnected from both i and j. Suppose that the probabilities of the three states for a contracted edge k are denoted by $\bar{p}_k$, $\bar{q}_k$, and $\bar{r}_k$, respectively, and k comprises m edges whose operating probability are p. Then, we can easily show that

$$\begin{aligned}\bar{p}_k &= p^m;\\ \bar{q}_k &= mp^{m-1}(1-p);\\ \bar{r}_k &= 1-p^m-mp^{m-1}(1-p).\end{aligned}$$

Based on the operating probability of contracted edges, we can evaluate the ATR of the candidate network more quickly due to the less number of nodes involved during computation. For example, recall the network depicted in Fig. 23.1 and assume that the edge operating probability p is 0.95 identically. It is difficult to compute the ATR of this network directly. For the contraction model illustrated in Fig 23.2, the indices and operating probabilities of the contracted edges are given as in Table 23.2. From simple computation based on Table 23.2, we can easily derive the ATR as 0.982450.

If the cardinality of $\bar{E}$ is large, we should rely on the MCS. Even for this case, the contraction model helps to obtain a more accurate reliability estimate with a less iteration number thanks to the less cardinality of $\bar{E}$. To verify this, we performed computer experiments to compare the performances of the MCS conducted for the original network in Fig. 23.1 and its contraction model in Fig. 23.2. The average ATR estimation errors obtained from the 20 independent runs are depicted in

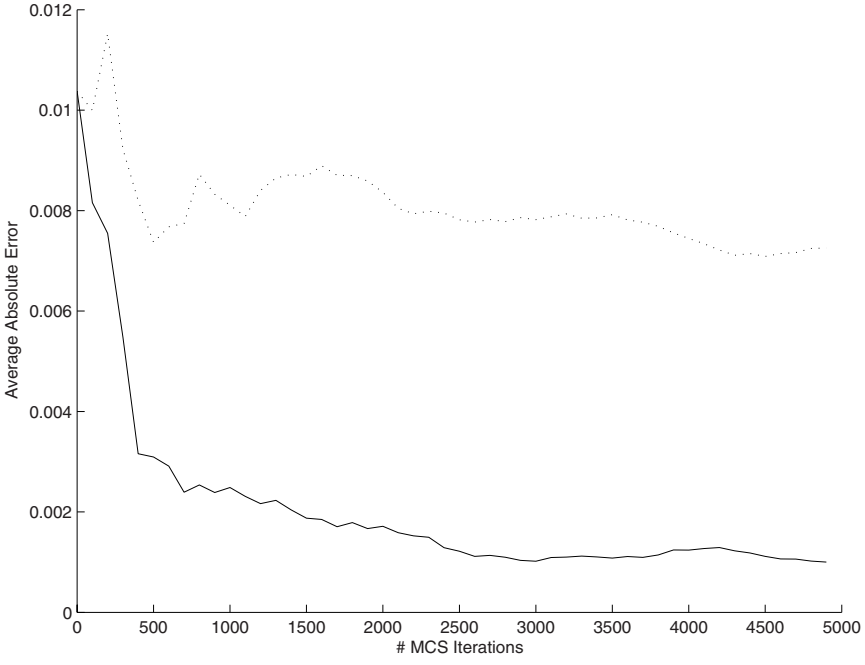


Fig. 23.3 ATR estimation errors averaged over 20 independent MCS runs performed for original network in Fig. 23.1 and its contraction model in Fig. 23.2 (dotted line: original network; solid line: contraction model)

Fig. 23.3. It is seen that the reliability estimate obtained from the contraction model converged to the actual ATR much faster. The difference in convergence speed will be even bigger if the given network has lots of nodes and only a limited portion of them are bridge nodes. Note that the contraction model is also useful in evaluating other network constraints such as throughput or transmission delay though it is beyond the scope of this study.

23.6 Hybrid Genetic Algorithm

Based on the MRE, this section proposes a new GA hybridized with Local Search ACS (LSACS). This GA is called Hybrid GA (HGA) henceforth. An individual of the HGA is given as a pair of a ring set and a pheromone matrix. With special operators dedicated to the MRE, the HGA evolves a population of individuals to seek for the best combination of rings to form the optimal solution of the given RNDP. The node order of each ring is fine-tuned by the LSACS that works with the pheromone matrix part of the individual. With this hybridization strategy, the poor local search capability of the GA is efficiently overcome.

Table 23.3 Pseudo code of the proposed hybrid GA

Line #	`Function` HGA
1	$g \leftarrow 0$;
2	Initialize($\mathbf{P}_g$);
3	`FOR EACH` $\mathbf{I} \in \mathbf{P}_g$ `DO`
4	Repair($\mathbf{I}$);
5	Evaluate($\mathbf{I}$);
6	`END_FOR`
7	`REPEAT`
8	$\mathbf{I}_1, \mathbf{I}_2 \leftarrow$ Select_Parents($\mathbf{P}_g$);
9	$\mathbf{I}_3, \mathbf{I}_4 \leftarrow$ Generate_Offspring($\mathbf{I}_1, \mathbf{I}_2$);
10	Mutate($\mathbf{I}_3$, $\mathbf{I}_4$);
11	LSACS($\mathbf{I}_3$, $\mathbf{I}_4$);
12	Evaluate($\mathbf{I}_3$, $\mathbf{I}_4$);
13	$\mathbf{P}_{g+1} \leftarrow$ Select($\mathbf{P}_g$, $\mathbf{I}_3$, $\mathbf{I}_4$);
14	$g \leftarrow g+1$;
15	`UNTIL` $g < g_T$

The pseudo code of the HGA is outlined in Table 23.3, where g denotes the generation index of the HGA, g_T the termination generation, $\mathbf{P}_g$ the population of individuals at g, $\mathbf{I}_1$ and $\mathbf{I}_2$ two parent individuals, and $\mathbf{I}_3$ and $\mathbf{I}_4$ two offspring individuals. At the initial generation, the HGA creates P individuals and evaluates them. At every subsequent generation, $\mathbf{I}_1$ and $\mathbf{I}_2$ are selected from the current population $\mathbf{P}_g$ to generate $\mathbf{I}_3$ and $\mathbf{I}_4$. The offspring $\mathbf{I}_3$ and $\mathbf{I}_4$ undergo mutation and LSACS operations. After evaluating the offspring, the HGA decides which individual will survive to the next generation. In the following subsections, each step of the HGA is discussed in detail.

23.6.1 Representation and Initialization

An individual $\mathbf{I}$ is a pair of a ring set $\mathbf{Y}$ and a pheromone matrix $\mathbf{H} \in \Re_+^{n \times n}$. The (i, j)-th element of $\mathbf{H}$, $\tau_{i,j}$, represents the desirability of choosing j as the next node of a ring from node i.

The initialization procedure creates P individuals to build the initial population $\mathbf{P}_0$. All the elements of $\mathbf{H}$ are initialized as

$$\tau_0 = \frac{1}{n \sum_{k=1}^{e} c_k}.$$

Denote V by the set of nodes visited by the rings in $\mathbf{Y}$, W a temporary node set used to create a new ring, and $N \backslash V$ the relative complement of V in N. The initialization procedure for $\mathbf{Y}$ is as follows:

1. Make $\mathbf{Y}$ and V empty sets.
2. Choose the size of a new ring r as a random integer $u \in \{3, \ldots, n\}$.
3. Make W an empty set.
4. Pick up a random node i in $N \backslash V$ and add it to W.
5. Choose $u-1$ random nodes other than i from N and add them to W.
6. Apply the nearest neighbor algorithm to the nodes in W to build r.
7. Add r to $\mathbf{Y}$ and update V. If $V = N$, stop; otherwise, go to Step 2.

The nearest neighbor algorithm is a simple heuristic to find a short ring r. Its procedure is as follows:

1. Select a random starting node in W as the current node and mark it as visited.
2. Add the edge connecting the current node to the nearest unvisited node to r.
3. Mark the nearest node as visited and assign the nearest node to the current node.
4. If all the nodes in W are visited, added the edge connecting the current node to the starting node and stop; otherwise go to Step 2.

Usually, the output of the nearest neighbor algorithm is not the shortest ring and can be improved by other heuristics.

23.6.2 Repair

The ring set $\mathbf{Y}$ of a newly created individual $\mathbf{I}$ undergoes a repair procedure, where its connectivity is tested and repaired. Even when every node in N is visited by one or more rings in $\mathbf{Y}$, the network represented by $\mathbf{Y}$ may be disconnected due to the presence of disconnected rings. The breadth-first search is used to check the connectivity of a given individual. If the individual is disconnected, it is repaired by taking new rings. To build a new ring, the repair procedure picks up a disconnected node i from N and builds a ring traversing i using the ring creation method used for the initialization procedure. The repair procedure is summarized as follows:

1. Check if the network represented by $\mathbf{Y}$ is connected. If so, stop; otherwise, go to next step.
2. Choose the size of a new ring r as a random integer $u \in \{3, \ldots, n\}$.
3. Make W an empty set.
4. Pick up a disconnected node $i \in N$ and add it to W.
5. Choose $u-1$ random nodes other than i from N and add them to W.
6. Apply the nearest neighbor algorithm to the nodes in W to obtain r.
7. Add r to $\mathbf{Y}$ and go to Step 1.

Note that a network represented by the MRE is two-edge-connected once it is connected. So, the connectivity repair algorithm also repairs two-edge-connectivity.

23.6.3 Fitness Evaluation

To handle the reliability constraint of the RNDP, the constraint violation rate is incorporated as the penalty term of the fitness function. Let $\mathbf{x}$ be a binary decision vector represented by an individual $\mathbf{I}$. Then, the fitness function f is formulated as:

$$f(\mathbf{x}) = \begin{cases} \frac{1}{C(\mathbf{x})} & \text{if } R(\mathbf{x}) \geq R_{\min}, \\ \frac{1}{C(\mathbf{x}) + n(R_{\min} - R(\mathbf{x}))\sum_{k=1}^{e} c_k} & \text{otherwise.} \end{cases} \tag{23.5}$$

Using this fitness function allows an infeasible candidate solution to be included in the population. Maintaining infeasible individuals in the population is known to be helpful to find the optimal solution that may lie on the constraint satisfaction boundary in the search space.

For a newly created individual $\mathbf{x}$, $R(\mathbf{x})$ is initially estimated as Jan's upper bound. At every generation of the HGA, the best individual $\mathbf{x}^*$ is transformed to the contraction model and undergoes the MCS of 10000 iterations to have a better estimate of $R(\mathbf{x})$. This step is skipped if $\mathbf{x}^*$ was the best individual in the past generation and already took the MCS. Moreover, if the fitness of offspring individual $\mathbf{I}_3$ (or $\mathbf{I}_4$) is higher than that of $\mathbf{x}^*$, it undergoes the MCS and the fitness is recalculated based on the MCS result.

23.6.4 Parent Selection and Offspring Generation

The HGA uses the *steady state selection*, which generates two offspring at every generation and makes them compete with the individuals in the current population [56]. The HGA carries out tournament selection of size κ to select two parents $\mathbf{I}_1$ and $\mathbf{I}_2$ from $\mathbf{P}_g$. That is, κ individuals are randomly chosen from $\mathbf{P}_g$ and the one with the highest fitness value is selected as $\mathbf{I}_1$. The same procedure is repeated to select $\mathbf{I}_2$.

The offspring $\mathbf{I}_3$ and $\mathbf{I}_4$ are generated either from parents themselves or from crossover operation. Denote the crossover rate by $p_C \in [0,1]$. If a uniform random number $u \in [0,1]$ is greater than p_C, $\mathbf{I}_3$ and $\mathbf{I}_4$ are copied from $\mathbf{I}_1$ and $\mathbf{I}_2$, respectively; otherwise, $\mathbf{I}_3$ and $\mathbf{I}_4$ are generated from the crossover operation. In the literature, special crossover operators have been proposed to handle permutation-based representation. Such operators include order crossover [57] and partially mapped crossover [58]. These crossover operators, however, cannot be used for the HGA since they cannot manipulate individuals represented by multiple rings.

We use a new crossover operator considering a ring as the minimum swapping unit. Let $\mathbf{Y}_1$, $\mathbf{Y}_2$, $\mathbf{Y}_3$, and $\mathbf{Y}_4$ be the ring sets of $\mathbf{I}_1$, $\mathbf{I}_2$, $\mathbf{I}_3$, and $\mathbf{I}_4$, respectively. The crossover procedure is outlined as follows:

1. Make $\mathbf{Y}_3$ and $\mathbf{Y}_4$ empty sets.
2. Build a temporary pool of rings as the union of $\mathbf{Y}_1$ and $\mathbf{Y}_2$.
3. Move each ring in the temporary pool either to $\mathbf{Y}_3$ or to $\mathbf{Y}_4$ with the equal probability.
4. Apply the repair algorithm to $\mathbf{Y}_3$ (or $\mathbf{Y}_4$) if $\mathbf{Y}_3$ (or $\mathbf{Y}_4$) represents a disconnected network.

Since computer experiments showed that swapping pheromone matrix elements is not helpful, no crossover operation is performed for the pheromone matrix part of the individuals; that is, the pheromone matrices of $\mathbf{I}_1$ and $\mathbf{I}_2$ are directly copied to

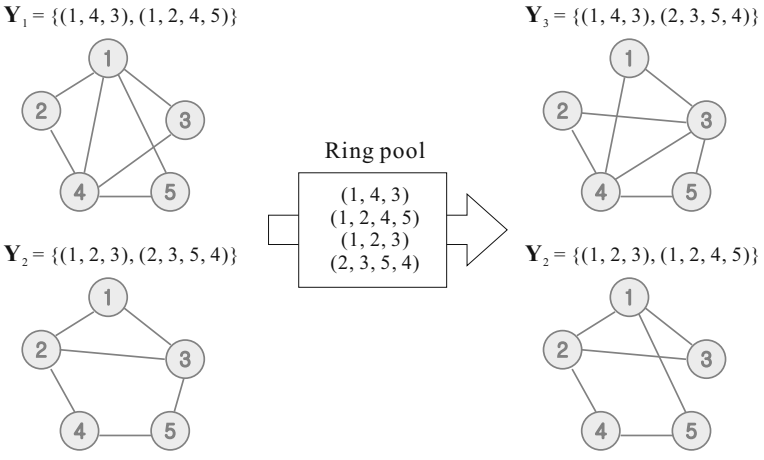


Fig. 23.4 Example of crossover operation

$\mathbf{I}_3$ and $\mathbf{I}_4$, respectively. Fig. 23.4 illustrates the crossover operation of multi-ring-encoded networks.

23.6.5 Mutation

The mutation operator comprises three sub-operators dedicated to the MRE: *ring-merging*, *ring-splitting*, and *ring-resizing* operators. The first two operators changes the size of $\mathbf{Y}$ while the last one changes the number of nodes visited by the rings. Since the node order of each ring will be fine-tuned by the LSACS, no permutation-based mutation operators such as 2-Opt are used here. Moreover, no mutation operation is performed for the pheromone matrix.

The three sub-operators are applied one by one. The ring-merging operation is performed with a probability $p_{\mathrm{RM}} \in [0,1]$. It randomly selects two rings r_a and r_b in $\mathbf{Y}$ and applies the nearest neighbor algorithm to the nodes visited by r_a and r_b to create a new ring, which replaces both r_a and r_b in $\mathbf{Y}$. The network represented by the mutated individual is generally cheaper but less reliable than the original one.

The ring-splitting operator is carried out with a probability $p_{\mathrm{RS}} \in [0,1]$. It randomly selects one ring $r_c \in \mathbf{Y}$ whose size is greater than three and replicates it as r_d. Let the number of nodes visited by r_d is n_d. Given a uniform random integer $u \in \{1,\ldots,n_d-3\}$, the ring-splitting operator removes $u+1$ consecutive edges from r_d and places an edge between two disconnected nodes to make r_d connected. The ring-splitting operator ends by adding r_d to $\mathbf{Y}$. As a consequence, a shortcut is placed between two nodes visited by r_c to make the network more expensive but more reliable.

To every ring $r \in \mathbf{Y}$, the ring-resizing operator is applied with a probability $p_{\mathrm{RR}} \in [0,1]$. Let N_r be the set of nodes visited by the ring r. The ring-resizing operator either augments or diminishes r with an equal probability 0.5. When augmenting r, the ring-resizing operator randomly chooses a node i in $N \backslash N_r$ and takes the nearest

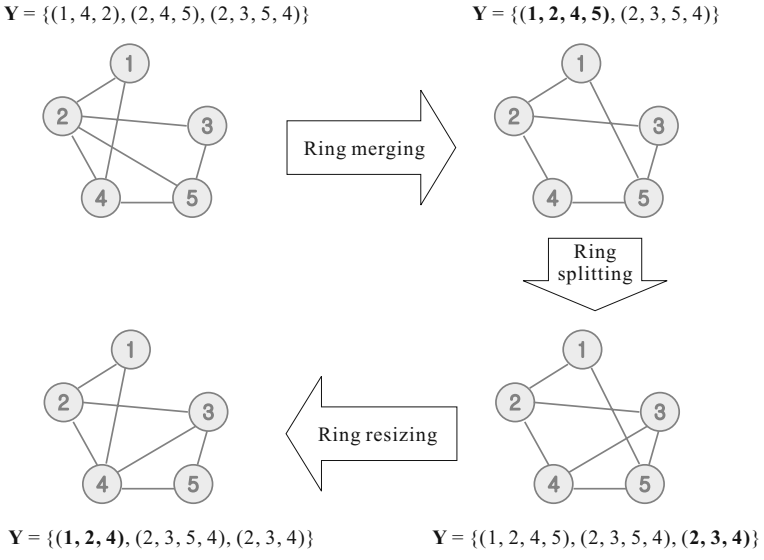


Fig. 23.5 Example of mutation operation for the multi-ring-encoded network

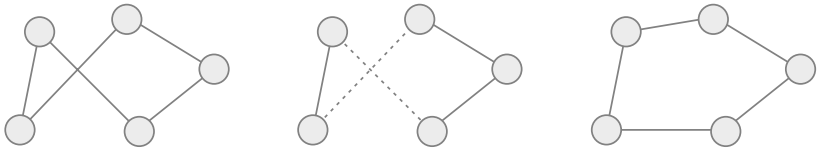
node $j \in N_r$ from i. Between two neighbor nodes of j in r, the one h nearer to i is chosen. The existing edge between j and h is removed while two edges between i and j and i and h are added to r. When diminishing r, the ring-resizing operator randomly chooses a node i in N_r. While removing two edges adjacent to i, the edge connecting two neighborhood nodes of i is added. The ring-diminishing operation may destroy the connectivity of the network represented by $\mathbf{Y}$. If this is the case, the repair algorithm is applied. Fig. 23.5 illustrates the mutation operation.

23.6.6 Local Search Ant Colony System

The LSACS is a key procedure fine-tuning the rings representing the offspring. The same LSACS operation is applied to $\mathbf{I}_3$ and $\mathbf{I}_4$. Let $\mathbf{I} = (\mathbf{Y}, \mathbf{H})$ be the offspring at hand. Then, the LSACS procedure comprises the following seven steps:

1. Given $\mathbf{I} = (\mathbf{Y}, \mathbf{H})$, generate a ring sets with ants.
2. Calculate f's of the networks represented by the a ring sets and select the best one as $\mathbf{Y}^*$.
3. If f of $\mathbf{Y}^*$ is higher than that of $\mathbf{Y}$, replace $\mathbf{Y}$ with $\mathbf{Y}^*$.
4. Apply 2-Opt operation to $\mathbf{Y}$ to build b ring sets.
5. Calculate f's of the networks represented by the b ring sets and select the best one as $\mathbf{Y}^{**}$.
6. If f of $\mathbf{Y}^{**}$ is higher than that of $\mathbf{Y}$, replace $\mathbf{Y}$ with $\mathbf{Y}^{**}$.
7. Perform global update of $\mathbf{H}$ with $\mathbf{Y}$.

In Step 1, the LSACS sends out artificial ants for all the rings in $\mathbf{Y}$ to generate each of a ring sets. The artificial ants attempt to connect the nodes visited by original



(a) A ring is given.

(b) Two edges are randomly chosen and removed.

(c) Disconnected paths are reconnected in other possible way.

Fig. 23.6 Example 2-Opt operation

rings in other possible ways. If the new ring set generated by the ants achieve a better fitness, it replaces $\mathbf{Y}$ as described in Step 2 and 3. In Step 4, a ring randomly chosen from $\mathbf{Y}$ passes through further fine-tuning stage named 2-Opt operation to generate each of b ring sets [51]. The 2-Opt operator randomly picks up two edges in r and removes them to have two separate paths. The paths are reconnected in other possible way by reversing the node sequence of one path. The example of 2-Opt operation is illustrated in Fig. 23.6. If this modification improves the fitness, the modified ring replaces r as shown in Step 5 and 6. Step 7 updates the pheromone matrix with the updated ring set. More details on each step is described in the later part of this subsection.

The proposed LSACS is different from the standard ACS in two aspects. First, the LSACS optimizes multiple rings at the same time. Given $\mathbf{Y} = \{r_1, \ldots, r_L\}$, the LSACS creates L ants, which share the same pheromone matrix $\mathbf{H}$. Second, the LSACS performs a single iteration of ant algorithm while the standard ACS performs multiple iterations, each of which creates multiple ant tours and updates pheromone matrix with the best tour. For the proposed HGA, a single iteration is enough since the pheromone matrix evolves as a part of the HGA individual. The multi-generation operation of the HGA effectively simulates the multi-iteration operation of the standard ACS.

To generate each of a ring sets, the LSACS assigns L ants to $N_1, \ldots, N_L$, where N_l for $l \in \{1, \ldots, L\}$ represents the set of nodes visited by r_l. The daemon of each ant generates a tour (ring) from N_l using the algorithm that is identical to the nearest neighbor algorithm described in Section 23.6.1 except for the way to choose the next node from the current node. Instead of taking the nearest node as the next node, the ant daemon uses the pseudo-random-proportional rule to choose the next node. Given the current node i, unvisited node set U, $\mathbf{H}$, and edge cost $c_{i,j}$ for $j \in U$, the daemon builds an ant decision table whose element corresponding to $j \in U$ is formulated as:

$$a_{i,j} = \frac{\tau_{i,j} \cdot c_{i,j}^{\beta}}{\sum_{m \in U} \tau_{i,m} \cdot c_{i,m}^{\beta}},$$

where $\beta < 0$ is a tunable parameter representing the relative importance of $c_{i,j}$ over $\tau_{i,j}$. With a probability $p_{\mathrm{A}} \in [0,1]$, the daemon chooses the next node j such that

$j = \text{argmax}_{m \in U} a_{i,m}$. With a probability $1 - p_A$, a roulette wheel selection method is used, where the probability of choosing j is given as $a_{i,j} / \sum_{m \in U} a_{i,m}$.

The LSACS performs a local pheromone update, which gives penalty to the pheromone content corresponding to the edges in the ant tours just generated. If the ant tours include an edge between node i and j, the associated pheromone is updated as

$$\tau_{i,j} = (1 - \varphi)\tau_{i,j} + \varphi\tau_0, \tag{23.6}$$

where $\varphi \in (0,1)$ is a tunable parameter. The local pheromone update increases solution diversity by encouraging the next ants to generate a new tour that has not emerged so far.

A global pheromone update is performed for the updated ring set $\mathbf{Y}$. If $\mathbf{Y}$ contains an edge between node i and j, the associated pheromone is updated as

$$\tau_{i,j} = (1 - \rho)\tau_{i,j} + \rho f,$$

where $\rho \in (0,1)$ is a tunable parameter and f is the fitness of the updated $\mathbf{Y}$. The higher the fitness of $\mathbf{Y}$ is, the more pheromone is deposited over the edges in $\mathbf{Y}$. Note that the LSACS never destroys the connectivity of the network represented by the offspring.

23.6.7 Selection for New Generation

The best P individuals are selected from $\mathbf{P}_g \cup \{\mathbf{I}_3, \mathbf{I}_4\}$ to form the next population $\mathbf{P}_{g+1}$. The best individual in $\mathbf{P}_{g+1}$ undergoes the MCS unless it has already taken the MCS. Because the MCS is the most time-consuming procedure of the HGA, how frequently the best individual changes over generations affects the execution time of the HGA.

23.7 Numerical Results

To verify the efficiency of the HGA, we compared its solution quality and CPU second to those of Branch and Bound Algorithm (BBA) and Local Search GA (LSGA) proposed in [1] and [10], respectively. The BBA is an exact algorithm that arranges all the candidate solutions of the RNDP in a special order. This arrangement inspires an efficient strategy to find the optimal solution by scanning a portion of the search space. The LSGA is an edge-encoding GA that works with special genetic operators implementing greedy local search heuristics dedicated to the RNDP. The details on the LSGA and BBA are described first and the numerical results are discussed later in this section.

The main differences between the LSGA and HGA are summarized in Table 23.4. For fair comparison between the LSGA and HGA, we used the following experimental setup. First, both algorithms started from the same initial population and evaluated a candidate solution using the same fitness formula (23.5). Second,

Table 23.4 Main differences between the LSGA and HGA

	LSGA [10]	HGA
Network encoding	Edge encoding	Multi-ring encoding
MCS	Based on original network	Based on contraction model
Repair algorithm	Greedy edge augmentation	Random ring augmentation
Crossover	Uniform edge crossover	Ring swapping
Local search heuristic	Randomized greedy mutation	Local search ant colony system

the steady state version of the LSGA was implemented so that it would breed two offspring at every generation. This choice was based on the observation that the number of offspring mainly affects the computation time of both algorithms. Third, the two algorithms applied the MCS to the best individual in the current population and the offspring individuals whose initial fitness evaluated with Jan's upper bound are higher than the best individual.

On the other hand, we retained the core features of the LSGA such as the encoding method, repair algorithm, and genetic operators. The edge encoding of the LSGA represented a candidate network with an e-dimensional binary vector (23.1). To repair the candidate network violating two-edge-connectivity constraint, the LSGA used a greedy edge augmentation procedure, which added the least-cost edges to connect the nodes of degree one. The uniform crossover operator of the LSGA ensured that each offspring is two-edge-connected and contains a least-cost spanning tree in its parent. The mutation operator implemented a randomized greedy local search algorithm. If every node in N have a degree of two, an edge was randomly chosen and added to the network. If the node degrees are greater than two for all the nodes, expensive edge was removed from the network once the two-edge-connectivity was maintained. The LSGA carried out the MCS for the original network while the HGA used the contraction model for the MCS runs.

Both algorithms used $P = 50$, $g_T = 5000$, $\kappa = 3$, $p_C = 0.7$, and MCS iterations of 10000, which gave good results during the experiment. The HGA used $p_{RM} = 0.3$, $p_{RS} = 0.3$, $p_{RR} = 0.1$, $a = b = 2$, $\beta = 3$, $p_A = 0.9$, and $\varphi = \rho = 0.1$. The mutation rate and drop rate of the LSGA are chosen as 0.3 and 0.6, respectively, as suggested in [10].

The BBA is an exact algorithm attempting to find the optimal solution of the RNDP by going through the limited portion of the search space using special arrangement of candidate solutions. The BBA works only for the case, where all the edges in E have the same operating probability p. All the candidate solutions of the RNDP are grouped into $e-n+2$ sets $S_{n-1}, S_n, \ldots, S_e$, where $e = n(n-1)/2$ and S_l represents the set of candidate solutions composed of l edges. Using (23.4), the BBA approximates the maximum ATR achievable by S_l denoted by R_l. Since $R_{n-1} < R_n < \ldots < R_e$, the BBA first determines l^* such that $R_{l^*-1} < R_{\min}$ and $R_{l^*} \geq R_{\min}$ to reduce the search space to $S_{l^*} \cup \cdots \cup S_e$.

Table 23.5 RNDP instances used for the experiments

Network size (n)	Edge operating probability (p)	Minimum reliability requirement ($R_{\min}$)
7	0.9	0.95
10	0.9	0.95
20	0.95	0.95
30	0.97	0.95
40	0.975	0.95
50	0.98	0.95
70	0.985	0.95
100	0.99	0.95

Candidate solutions in each set S_l for $l \in \{l^*, \ldots, e\}$ are sorted according to the ascending order of costs. Starting from S_{l^*}, the candidate networks in each set are evaluated one by one. If the ATR upper bound (23.4) of a candidate network is higher than $R_{\min}$, the MCS is applied to the candidate network to obtain a better ATR estimate. If this estimate is no less than $R_{\min}$, the BBA skips the rest solutions in the current set S_l since they are more expensive than the one just evaluated. Before moving on the next set S_{l+1}, the BBA checks if the first solution candidate in S_{l+1} is more expensive than the best solution found so far. If this is the case, the BBA terminates outputting the best solution.

The three algorithms were implemented with C++ Standard Template Library and tested over Intel Core2 Duo T5550 1.83GHz CPU. Eight RNDP instances with $n = 7$, 10, 20, 30, 40, 50, 70, and 100 were created for the experiments as listed in Table 23.5. When creating a RNDP instance, a node position was chosen randomly and uniformly in a unit square region $[0, 1] \times [0, 1]$. The edge cost was set to the two-dimensional Euclidean distance between the two nodes connected by the edge. The minimum reliability requirement was set to 0.95 for all the RNDP instances. The operating probability of edges was chosen low enough to avoid having a Hamiltonian cycle as the solution of the RNDP.

For each RNDP instance, a single run of the BBA and 30 independent runs of the LSGA and HGA were performed since the BBA is an exact algorithm while the other two are stochastic ones. When starting each run of the LSGA and HGA, we generated 50 individuals using the method described in Section 23.6.1 and assigned them to the initial population of the two algorithms.

Table 23.6 shows the solution quality of the three algorithms. The BBA found the solution for the RNDP instance with $n = 7$ only. For the other RNDP instances, BBA did not terminate within the first six hours. This implies that the exact algorithms cannot handle the realistic RNDP in an acceptable time. On the other hand, both LSGA and HGA yielded fairly good solutions for all the RNDP instances. Even for the RNDP with $n = 7$, the solution quality of the HGA was comparable to the BBA.

It was also seen that the average solution quality of the HGA was better than the LSGA. We ran two-tailed paired t-test for the best fitness values of the 30 runs

Table 23.6 Average solution quality obtained from the single BBA run and 30 LSGA and HGA runs (Better results are highlighted in boldface)

RNDP (n)	BBA	LSGA	HGA
7	**2.623**	2.913	2.625
10	N/A	4.763	**4.318**
20	N/A	5.712	**5.190**
30	N/A	7.206	**6.749**
40	N/A	8.425	**7.582**
50	N/A	9.220	**8.315**
70	N/A	10.974	**10.133**
100	N/A	13.638	**11.490**

and obtained P-value less than 0.001 for every RNDP instance. This verifies that the HGA significantly outperforms the LSGA in solution quality. Further investigation on the experiment results revealed that the LSGA tended to converge prematurely at early generations due to its greedy repair and mutation operators biased to find nearby local optima. The diversity of the LSGA crossover operator was limited because the edge of an offspring always comes from one of its parents. Moreover, the edge-wise greedy heuristic was not so helpful to fine-tune the paths comprising a candidate network. For example, the edge-wise greedy heuristic cannot mimic 2-Opt operation. On the other hand, the HGA exhibited strong solution diversity. Actually, the crossover operator of the HGA is able to generate any two-edge-connected network regardless how parents look like. This is because of the ring swapping and augmentation mechanism. It was also observed that the mutation and LSACS operators of the HGA worked properly to fine-tune the ring topologies comprising a candidate network.

Table 23.7 lists the statistics of the CPU seconds obtained from the three algorithms. Between the LSGA and HGA, we could not judge which algorithm generally ran faster. We only observed that the CPU seconds of the LSGA varied widely while the CPU seconds of the HGA were roughly proportional to n. As mentioned before, the CPU second is mainly decided by the number of MCS runs performed for the offspring individuals. Once all the individuals in the current population represent the same network topology, the LSGA keeps generating similar offspring. Therefore, the LSGA will experience no MCS run if the offspring never outperforms the best individual in future generations. On the other hand, if the reliability estimate of the best individual is slightly lower than $R_{\min}$, the offspring will outperform the best individual and undergo the MCS procedure, which may adjust the fitness of the offspring lower than the best individual. This could be repeated for all the remaining generations to waste CPU seconds. The consequence of this irregular behavior of the LSGA is also illustrated in Table 23.8, which shows the ratio of computation time spent to perform MCS during the runs of the two algorithms. The non-MCS computational load was roughly proportional to n for the HGA, but no such a tendency was observed for the LSGA. It is also seen that the HGA required more computational

Table 23.7 CPU seconds obtained from the single BBA run and 30 LSGA and HGA runs (Better results are highlighted in boldface)

RNDP (n)	BBA	LSGA		HGA	
		mean	std. dev.	mean	std. dev.
7	311	121	130	**28**	12
10	N/A[2]	74	156	**40**	23
20	N/A	174	394	**77**	27
30	N/A	**12**	13	58	20
40	N/A	**64**	193	111	49
50	N/A	**57**	138	114	39
70	N/A	312	1110	**258**	100
100	N/A	**227**	794	389	49

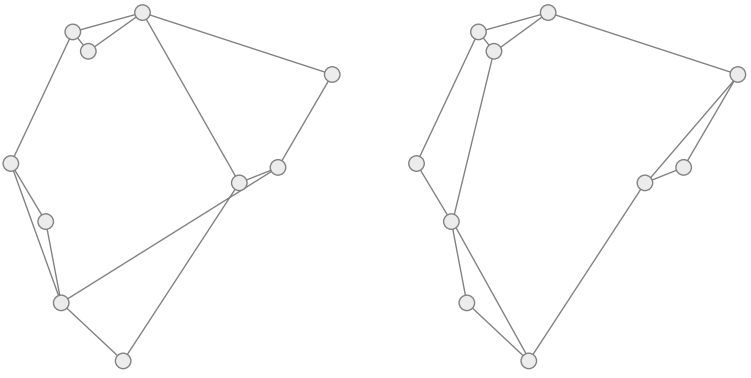
Table 23.8 Average ratios of CPU seconds spent to perform MCS during the LSGA and HGA runs

RNDP (n)	LSGA	HGA
7	0.98	0.86
10	0.98	0.88
20	0.97	0.82
30	0.64	0.58
40	0.89	0.61
50	0.81	0.41
70	0.93	0.42
100	0.71	0.16

resources than the LSGA to perform non-MCS routines. This implies that the special genetic operators and LSACS dedicated to the MRE are more expensive than conventional counterparts.

Fig. 23.7 depicts the best solutions obtained from the 30 runs of the LSGA and HGA for the RNDP instance with $n = 10$. Even with naked eyes, we could see that the HGA yielded a better solution. This is even more impressing if we consider the number of evaluated candidate solutions. With $P = 50$, $g_T = 5000$, and $a = b = 2$, the 30 HGA runs evaluated 1.5×10^6 distinctive candidate solutions at most. This is very tiny compared to the exhaustive search space size 3.52×10^{13}.

The results of computer experiments verified that the RNDP is a computationally expensive problem. As an enumeration-based approach, the BBA worked for the RNDP only when n is very small. Even for a moderate network size with $n = 10$, the BBA did not terminate within six hours of CPU time. On the other hand, the counterpart methods based on computational intelligence handled realistic RNDP effectively. The numerical results also proved that the search capability of the traditional edge-represented GA is limited compared to the HGA, which could handle



(a) LSGA ($C = 4.389$, $R = 0.951$) (b) HGA ($C = 4.048$, $R = 0.950$)

Fig. 23.7 Best solutions obtained from the LSGA and HGA for the RNDP instance with $n = 10$

the RNDP more efficiently. This suggests that a proper hybridization may achieve a synergetic alliance of two or more computational-intelligence-based heuristics. Moreover, we showed that the problem-specific representation method such as the MRE plays an important role in improving solution quality and CPU seconds of the population-based heuristics.

References

1. Jan, R.-H., Hwuang, F.-J., Chen, S.-T.: Topological optimization of a communication network subject to a reliability constraint. IEEE Transactions on Reliability 42(1), 63–70 (1993)
2. Johnson, D.S., Lenstra, J.K., Kan, A.H.G.R.: The complexity of the network design problem. Networks 8, 279–285 (1978)
3. Aggarwal, K.K., Chopra, Y.C., Bajwa, J.S.: Topological layout of links for optimising the overall reliability in a computer communication system. Microelectronics and Reliability 22(3), 347–351 (1982)
4. De Boer, P.-T., Kroese, D.P., Mannor, S., Rubinstein, R.Y.: A tutorial on the cross-entropy method. Annals of Operations Research 134(1), 19–67 (2005)
5. Pierre, S., Hyppolite, M.-A., Bourjolly, J.-M., Dioume, O.: Topological design of computer communication networks using simulated annealing. Engineering Applications of Artificial Intelligence 8(1), 61–69 (1995)
6. Randall, M., Mcmahon, G., Sugden, S.: A simulated annealing approach to communication network design. Journal of Combinatorial Optimization 6(1), 55–65 (2002)
7. Jayaraman, V., Ross, A.: A simulated annealing methodology to distribution network design and management. European Journal of Operational Research 144(3), 629–645 (2003)
8. Glover, F., Lee, M., Ryan, J.: Least-cost network topology design for a new service: An application of a tabu search. Annals of Operations Research 33(5), 351–362 (1991)

9. Pedersen, M.B., Crainic, T.G., Madsen, O.B.G.: Models and tabu search metaheuristics for service network design with asset-balance requirements. Transportation Science (2008), doi:10.1287/trsc.1080.0234
10. Dengiz, B., Altiparmak, F., Smith, A.E.: Local search genetic algorithm for optimal design of reliable networks. IEEE Transactions on Evolutionary Computation 1(3), 179–188 (1997)
11. Gen, M., Kumar, A., Kim, J.R.: Recent network design techniques using evolutionary algorithms. Int. J. Production Economics 98(2), 251–261 (2005)
12. Poorzahedy, H., Rouhania, O.M.: Hybrid meta-heuristic algorithms for solving network design problem. European Journal of Operational Research 175(2), 707–721 (2006)
13. Gang, P., Iimura, I., Nakayama, S.: An evolutionary multiple heuristic with genetic local search for solving TSP. International Journal of Information Technology 14(2), 1–11 (2008)
14. Krasnogor, N., Smith, J.: A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Transactions on Evolutionary Computation 9(5), 474–488 (2005)
15. Randall, M., Tonkes, E.: Solving network synthesis problems using ant colony optimisation. In: Monostori, L., Váncza, J., Ali, M. (eds.) IEA/AIE 2001. LNCS (LNAI), vol. 2070, pp. 1–10. Springer, Heidelberg (2001)
16. Dorigo, M., Caro, G.D., Gambardella, L.M.: Ant algorithms for discrete optimization. Artificial Life 5(2), 137–172 (1999)
17. Deb, K.: Multi-objective optimization using evolutionary algorithms. John Wiley and Sons, Chichester (2001)
18. Konak, A., Coit, D.W., Smith, A.E.: Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering & System Safety 91(9), 992–1007 (2006)
19. Weichenberg, G.E., Chan, V.W.S., Medard, M.: High-reliability architectures for networks under stress. In: Proc. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (2004)
20. Ball, M., Van Slyke, R.M.: Backtracking algorithms for network reliability analysis. Ann. Discrete Math. 1, 49–64 (1977)
21. Lomonosov, M.V., Polesskii, V.P.: Lower bound of network reliability. Problems of Information Transmission 8, 118–123 (1972)
22. Prim, R.C.: Shortest connection networks and some generalizations. Bell System Technical Journal 36, 1389–1401 (1957)
23. Lawler, E.L., Lenstra, J.K., Kan, A.H.G.R., Shmoys, D.B.: The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. Wiley, Chichester (1985)
24. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)
25. Kumar, A., Pathak, R.M., Gupta, Y.P.: Genetic algorithm based reliability optimization for computer network expansion. IEEE Transactions on Reliability 44(1), 63–72 (1995)
26. Pierre, S., Legault, G.: A genetic algorithm for designing distributed computer network topologies. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 28(2), 249–258 (1998)
27. Chou, H., Premkumar, G., Chu, C.-H.: Genetic algorithms for communications network design-an empirical study of the factors that influence performances. IEEE Transactions on Evolutionary Computation 5(3), 236–249 (2001)
28. Marseguerra, M., Zio, E., Podofillini, L., Coit, D.W.: Optimal design of reliable network systems in presence of uncertainty. IEEE Transactions on Reliability 54(2), 243–253 (2005)

29. Cordón, O., Viana, I.F., Herrera, F., Moreno, L.: A new ACO model integrating evolutionary computation concepts: the best-worst ant system. In: Proc. Second International Workshop on Ant Algorithms, Brussels, Belgium, pp. 22–29 (2000)
30. Gong, D., Ruan, X.: A hybrid approach of GA and ACO for TSP. In: Proc. 5th World Congress on Intelligent Control and Automation, pp. 2068–2072 (2004)
31. Pilat, M.L., White, T.: Using genetic algorithms to optimize ACS-TSP. In: Proc. 3rd Int. Workshop on Ant Algorithms, Brussels, Belgium, pp. 282–287 (2002)
32. Tseng, L.-Y., Liang, S.-C.: A hybrid metaheuristic for the quadratic assignment problem. Computational Optimization and Applications 34(1), 85–113 (2006)
33. Acan, A.: GAACO: A GA + ACO Hybrid for Faster and Better Search Capability. In: Proc. 3rd Int. Workshop on Ant Algorithms, Brussels, Belgium, pp. 15–26 (2002)
34. Lee, Z.-J.: A hybrid algorithm applied to travelling salesman problem. In: Proc. IEEE International Conference on Networking, Sensing and Control, pp. 237–242 (2004)
35. Tseng, L.-Y., Chen, S.-C.: A hybrid metaheuristic for the resource-constrained project scheduling problem. European Journal of Operational Research 175(2), 707–721 (2006)
36. Lei, C.: A MCM interconnect test generation optimization scheme based on ant algorithm and genetic algorithm. In: Proc. 6th Intern. Conf. Electronic Packaging Technology, pp. 710–713 (2005)
37. Silva, C.A., Faria, J.M., Abrantes, P., Sousa, J.M.C., Surico, M., Naso, D.: Concrete delivery using a combination of GA and ACO. In: Proc. 44th IEEE Conf. Decision and Control and European Control Conference, pp. 7633–7638 (2005)
38. Li, M., Wang, H., Li, P.: Tasks mapping in multi-core based system: hybrid ACO&GA approach. In: Proc. 5th Intern. Conf. on ASIC, pp. 335–340 (2003)
39. Prüfer, H.: Neuer beweis eines satzes uber permutation. Arch. Math. Phys. 27, 742–744 (1918)
40. Krishnamoorthy, M., Ernst, A.T., Sharaiha, Y.M.: Comparison of algorithms for the degree constrained minimum spanning tree (Tech. Rep.). CSIRO Mathematical and Information Sciences, Clayton, Australia
41. Abuali, F.N., Wainwright, R.L., Schoenefeld, D.A.: Determinant factorization: a new encoding scheme for spanning trees applied to the probablilistic minimum spanning tree problem. In: Proc. 6th Intern. Conf. Genetic Algorithms, pp. 470–477. Morgan Kaufmann, San Mateo (1995)
42. Rothlauf, F., Goldberg, D.E., Heinzl, A.: Network random keys – a tree network representation scheme for genetic and evolutionary algorithms (Tech. Rep. No. 8/2000). University of Bayreuth, Germany
43. Raidl, G.R., Julstrom, B.A.: A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem. In: Proc. 2000 ACM Symp. Applied Computing, pp. 440–445 (2000)
44. Cheriton, D., Tarjan, R.E.: Finding minimum spanning trees. SIAM J. Comput. 5(4), 724–742 (1976)
45. Lee, Y., Chiu, S.Y., Sanchez, J.: A branch and cut algorithm for the Steiner ring star problem. International Journal of Management Science 4, 21–34 (1998)
46. Song, Y., Wool, A., Yener, B.: Combinatorial design of multi-ring networks with combined routing and flow control. Computer Networks 41(2), 247–267 (2003)
47. Resendo, L.C., Pedro, J.M., Ribeiro, M.R.N., Pires, J.J.O.: ILP approaches to study interconnection strategies for multi-ring networks in the presence of traffic grooming
48. Won, J.-M., Karray, F.: A genetic algorithm with cycle representation and contraction digraph model for guideway network design of personal rapid transit. In: Proc. 2007 IEEE Cong. Evolutionary Computation, pp. 2405–2412 (2007)

49. Clarke, G., Wright, J.W.: Scheduling of vehicles from a central depot to a number of delivery points. Operations Research 12, 568–581 (1964)
50. Christofides, N.: Worst-case analysis of a new heuristic for the travelling salesman problem, Report No. 388, GSIA, Carnegie-Mellon University, Pittsburgh, PA
51. Croes, G.A.: A method for solving traveling salesman problems. Operations Research 6, 791–812 (1958)
52. Bock, F.: An algorithm for solving traveling-salesman and related network optimization problems. unpublished manuscript associated with talk presented at the 14th ORSA National Meeting (1958)
53. Jung, S., Moon, B.-R.: Toward minimal restriction of genetic encoding and crossovers for the two-dimensional Euclidean TSP. IEEE Transactions on Evolutionary Computation 6(6), 557–565 (2002)
54. Lin, S., Kernighan, B.: An effective heuristic algorithm for the traveling salesman problem. Operations Research 21, 498–516 (1973)
55. Johnson, D.S., McGeoch, L.A.: The traveling salesman problem: a case study in local optimization. In: Aarts, E.H.L., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 215–310. Wiley, New York (1997)
56. Sayoud, H., Takahashi, K., Vaillant, B.: Designing communication networks topologies using steady-state genetic algorithms. IEEE Communication Letters 5(3), 113–115 (2001)
57. Davis, L.: Applying adaptive algorithms to epistatic domains. In: Proc. Intern. Joint Conf. Artificial Intelligence, pp. 162–164 (1985)
58. Goldberg, D.E., Lingle, R.: Alles, loci and the TSP. In: Proc. 1st Intern. Conf. Genetic Algorithm and Their Applications, pp. 154–159 (1985)

Chapter 24
Isolated Word Analysis Using Biologically-Based Neural Networks

Walter M. Yamada and Theodore W. Berger

Abstract. A dynamic synapse neural network (*DSNN*) for speech recognition system input filtering and the genetic algorithm (GA) used to optimize *DSNN* parameters is presented. *DSNNs* are trained to respond to a target word (TW) said by one female speaker or by 8 male and 8 female speakers. The response of the single-speaker trained *DSNNs* to all 16 speakers is similar to the 16-speaker-trained *DSNN* responses. TW training results in an ordering of the expected responses to the 9 words of the non-TW set. The ordering determined by single-speaker training matches the ordering determined by multi-speaker training; and in many instances, the single-speaker trained *DSNN* output matches the multi-speaker trained *DSNN* output. While searching the parameter space to best solve the isolated word recognition task, the GA implicitly searched the input space to find the input subset best describing the separatrix between TWs and non-TWs. Computation is decreased by concentrating optimization on this subset. The GA adapts as knowledge of this subset is learned. The GA begins as a random search, becoming a steady state GA and then a simple elitist GA over the course of optimization.

24.1 Neural Engineering Speech Recognition Using the Genetic Algorithm

Voiced-command interfaces (VCI) often use isolated word recognition (IWR) algorithms to classify input sound. Generally, a button is pushed to alert the computer to record sound – this sound contains a spoken command-word (TW) embedded within background sound. The IWR algorithm is responsible for determining which

Walter M. Yamada
Laboratory of Applied Pharmacokinetics, University of Southern California,
Medical School, 2250 Alcazar Street, CSC 134B Los Angeles, CA 90033
e-mail: yamada@bmsr.usc.edu

Theodore W. Berger
University of Southern California, Los Angeles, CA 90089-1111
e-mail: berger@bmsrs.usc.edu

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 637–670.
springerlink.com

one of a limited number of menu TWs was most likely said. This task is not trivial. Menus contain a variable number of items and each TW is of variable length and placement within the recorded sound. The acoustic environment is unknown, e.g. the noise and reverberation characteristics are unknown. Each example word is a function; but the exact form of the function is altered by a large number of factors. For example, gender, age, accent (both regional as well as foreign language induced), current state of health or physical exertion, volume of speech (whisper, normal speech, or yelling), familiarity with the recording device, and experience with the language and syntax of the VCI. Finally, the functions that represent each menu item are not of uniform distance from each other. Controlling for these factors still results in a variable input signal since human speech is variable. Therefore, a statistical and nonlinear approach to IWR is required that generally begins with the analysis of many thousands of example inputs. To make this problem tenable, engineered VCIs are designed to solve the IWR task under specific operating restrictions. Our own auditory system however, is capable of learning new items after very few presentations and is able to generalize that learning to new speakers, new acoustic settings, an altered vocalization, a novel accent, different ambient noise conditions, etc. Because traditionally engineered VCIs are not robust in the face of the general problem, whereas our own auditory system is, there is much interest in determining the mechanisms that allow for human hearing.

Human hearing occurs in two steps. The first step happens between the ear and the cerebral cortex. Sound passes through a non-linear spectral/temporal filter, with many thousands of channels, expressing many different characteristics, with several clearly defined processing centers (nuclei), each of which has a complicated circuitry allowing for intranuclear channel mixing; also, each nuclei communicates with the others to affect internuclear processing [42]. The second step happens within the cerebral cortex, where auditory imaging studies are used to correlate neural activity in particular places with speech intelligibility [7, 10]. The cerebral cortex hierarchically processes speech. Higher processing feeds back to lower processing centers to allow for context sensitive sound filtering, both intracortically [57] and subcortically [16]. Unfortunately, the high degree of complexity and an inability to observe much of it, renders reverse engineering the auditory system difficult. Instead, a forward approach is taken to model aspects of auditory neuron function, to use these models to support IWR tasks, and to then ascertain how well the model supports the task. In this chapter we present one such model and the optimization algorithms required for studying it. The application presented here is a bio-mimetic neural engineering model of sub-cortical auditory filter behavior.

A genetic algorithm (GA) has served to greatly speed up model development and testing. The GA is an optimization tool that is intuitive to apply, is model independent, and is well known within the application area of natural computation [3]. The GA requires the list of system free parameters and ranges and an equation that scores how well a particular individual (a complete set of valid parameter values) solves for a stated objective. The GA is a recursive algorithm that generates sequential sets of individuals (populations) that are increasingly better at solving for the objective. It begins by scoring a population using the objective function and mapping the scores

to fitness values. Then, based on each individual's relative fitness it is cloned, killed, or mated to reproduce more individuals, generating a new population and allowing the loop to be re-run. Each new loop is a generation. In our case, the objective function measures how well a neural model may be used to classify a set of words. Whether an individual is cloned or killed depends only on if it is one of the best or one of the worst of the current generation at solving the IWR task. However, at the heart of all GAs is a fitness-weighted random process: reproduction. During reproduction the bit representation of corresponding parameter values from *different* individuals has a high probability of being split and recombined (crossover) to form new individuals. After new individuals are formed there is a small probability that any single bit will be flipped (mutated). This loop is illustrated in figure 24.1.

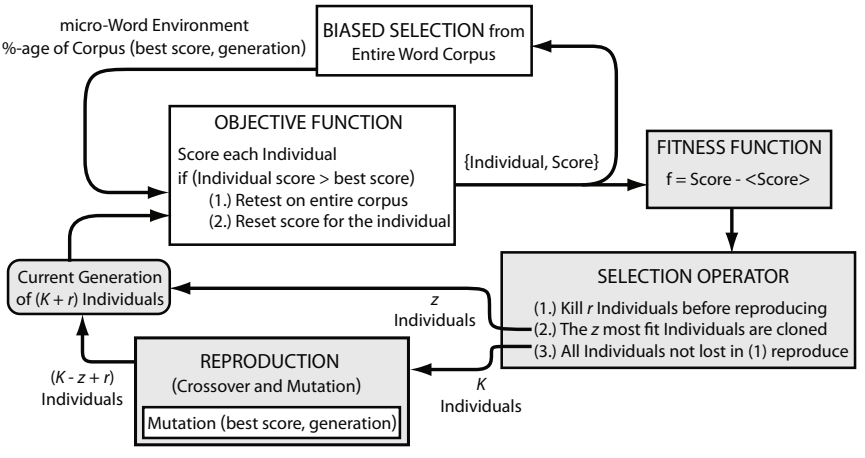


Fig. 24.1 The genetic algorithm used in this work is pictured above. At all times, $K = 40$, initially, $z = 40$ and $r = 32$, but as evolution progresses the value of z and r decrease, falling to 1. The grey boxes are controlled by the GA. Functionality of these boxes is explained in many places, e.g. [17, 33]. Functionality of the white boxes is explained in the text. Objective function: 24.4.2, Biased Selection of Words: 24.4.1, and Mutation and z: 24.4.4

The GA has many variations (different variable encoding, true or overlapping generations, adapting mutation and crossover parameters, multiple populations, etc.) each shown to be better applicable to particular classes of optimization problem. In figure 24.1, setting $z = K$, and foregoing re-scoring of the cloned individuals would result in a steady state GA with population overlap equal to r. When a single objective is being pursued – as is the case after the Biased Selector has converged to a stationary process – a simple GA suffices. In figure 24.1, setting $r = z = 0$, would result in such a GA. We refer to our GA as a modified elitist simple GA; the modification is $r = z = 1$.

Our free parameter space is modest: 210 parameters. Each parameter is encoded in an 8-bit binary string and converted to a real value on a specified range for use

in the model (Table 24.1). Computational overhead of the IWR problem during optimization lays almost exclusively in filtering many different sound examples many thousands of times. The first generation of these models was 'optimized' by an intuitive hand tweaking approach taking many months to finally 'converge' on a solution. They separated only four command-words and they incorporated simplistic approximations of biological dynamics that could be expressed in terms of linear functions and threshold devices. They also used a circuit construction algorithm that resulted in optimization for new menu items being a super-exponential task in terms of model complexity and numbers of input words. Nevertheless, these IWR systems were more noise-robust than state-of-the art speech recognition systems [25, 26, 27]. By redefining the IWR task to a set of binary operations (TW vs. all others), further simplifying the incorporated dynamics, and using a GA with the objective of having greater integrated TW output on a particular output lead vs. the other, time for parameter optimization was reduced to approximately two weeks; allowing for larger IWR systems to be designed [15, 35]. However the simple dynamics used were not useful for elucidating principles of human hearing and although they could be programmed for use for technological application [13], they offered no insight toward neural engineering (culturing) of biological hearing systems. Such insight requires that the designed systems be constructed with models that mimic neural parts as opposed to abstractions of neural functionality. The application reported here moves toward addressing these concerns; while also decreasing optimization time to approximately two days per model.

Biological synapses consist of complicated dynamic processes that constrain and support complex non-linear functional transformations that generally have no analytic description. The synapses used here incorporate models of both N-methyl-D-aspartate (NMDA) and α-amino-3-hydroxy-5-methyl-4-isoxazole propionate (AMPA) protein function [49]. This results in synapses resembling those found in the central nervous system. To optimize network performance, we developed an objective function that allows the GA to simultaneously conduct an implicit search of the input space and an explicit search of the model parameter space to:

1. find the smallest subset of input words that represents the separatrix between TWs and non-TWs, and
2. find the model parameter values that best solve for the IWR task on the entire word corpus, while
3. calling the Objective Function as few times as possible.

Simultaneous search is necessary because these two goals are inter-dependent. Each choice of input subset may imply different best parameter values, and vice-versa: each choice of parameter values classifies best a different input subset. Three measures of parameter fitness were calculated with the goal of applying evolutionary force in the direction of increasing information capacity, increasing accuracy, decreasing non-target word decision cost, and increasing network stability. The measurement weightings were determined by programmer experience. Fitness scores fell along a gradient of less to more useful network behaviors, with lowest scores applied to 'no-machines', and highest scores being associated with networks that

both classify the training set and also filter-out a portion of the non-TWs in the training set. Mutation rate and input subset were independently adjusted throughout training in order to control the degree to which optimization was directed and to focus optimization near the target/non-target word separatrix. Overtraining was implicitly prevented by use of the output PDFs across all networks, and by our preference for training with historically difficult to classify input (function of the Biased Selector).

Non-TWs could be discerned in two ways, either by correctly classifying them as non-TW or by filtering out (*not* responding) to the non-TW input sound. Training toward a filtering operation was accomplished by scaling the non-target-class decisions used to fill the confusion matrix by a cost or discernment factor. Because of the preponderance of non-target class input, after 100% classification was reached, but while optimization continued toward filtering-out non-target-class input, the calculated information capacity continued to increase because the ratio of TWs to non-TWs classified increased.

IWR analysis usually begins with an assumption that speech intelligibility is dependent upon recognizing particular patterns in the sound; analysis of the isolated speech is in terms of one or more of these classifiable sequences (usually spectral patterns or phonemes). We did not use these patterns for determining input encoding, network processing, or output analysis. We lumped nine of the ten input classes into a single "Negative Id" class. Furthermore, we used integrated output pulse trains (did not account for output dynamics) to score networks during training. Nevertheless, training resulted in a set of networks with discernible temporal (dynamic) output pattern clusters useful for input classification. These patterns are presented.

We show temporal output responses to filtered and pulse-encoded isolated speech of *DSNNs* constrained by equations approximating the functionality of NMDA and AMPA proteins and by the architecture of Figure 24.2. Emphasis is on dynamic synapse neural network (*DSNN*) formalism, optimization, and response visualization, i.e. the acoustic temporal processing task is converted into a spatial pattern recognition task. *DSNNs* have synapses consisting of a dynamic presynapse, generating an amplitude modulated pulse train for input to a single dynamic postsynapse. Each postsynapse is a variable resistor with instantaneous current proportional to a potential affected by multiple postsynapses, which are passively connected to each other and to a single cell body via a dendritic branch. The NMDA and AMPA equations are state transition models affecting the instantaneous postsynaptic membrane conductance. We show these networks have three cluster-types of trained I/O: first, multiple (distinct temporal, spectral, and phoneme description) input speech classes that elicit similar average temporal output $(n \mapsto 1)$; second , input speech classes that elicit a distinct average temporal output response $(1 \mapsto 1)$; and most often, multiple input speech classes that elicit a gradient of average temporal output response $([a \ldots n] \mapsto [A \ldots N])$.

ANNs are designed to provide efficient computation for well-defined tasks in regression analysis, classification and data processing. Biological neural networks, from which ANNs are inspired, are living tissue that has evolved physiological I/O

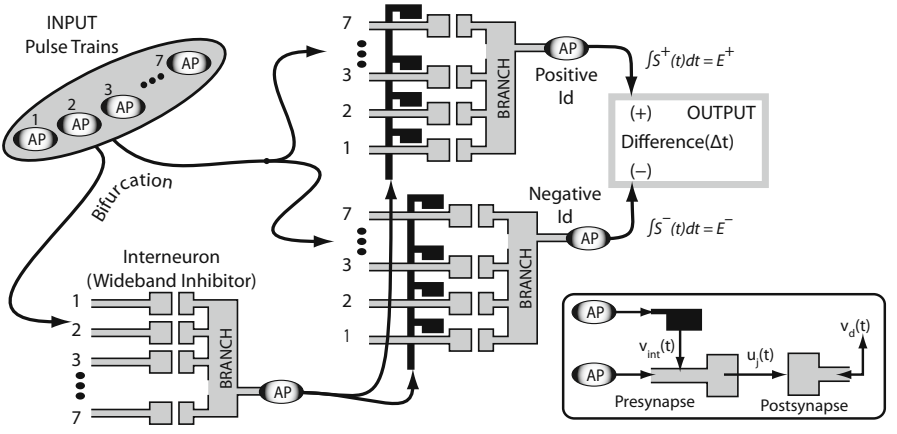


Fig. 24.2 *DSNN* architecture. Isolated speech samples were decomposed into seven continuous waveforms that were passed through an adaptive pulse encoder – oval shaded 'AP'. There are two output neurons, one with synapses trained to be responsive to a specific TW (Positive ID), the other with synapses trained to be responsive to all other input (Negative ID). Further detail is discussed throughout the chapter; signal flow through the Branch is illustrated in figure 24.4

descriptions that are interpretable as computation. The defined nature of the tasks for which ANNs are designed has allowed for the development of analytical tools for interpreting the computational ability of ANNs – there is as yet no such set of rules for construction, development, or interpreting evolution, of biological neural networks. The application herein is an open problem in both ANN and natural computation research: generalization from single-speaker training data to multiple-speaker performance. We compare averaged temporal output for single-speaker trained networks versus output of networks trained using sixteen speakers. The resulting output response clusters, and in many cases the actual average temporal output, was similar for both sets of trained networks. Speaker specific training resulted in speaker independent functionality. Our networks have the minimum architecture required to embody the speech recognition task and so are a suitable test of the computational ability supported by the synaptic dynamics relatively independent of other factors. We thus make progress toward determining rules for understanding how biological synapses may solve waveform classification problems. Our network is contrasted with the dorsal cochlear nucleus (DCN), which provides complex signal envelope processing necessary for human speech perception.

24.2 Input Description

The input set contains recordings from 8 male and 8 female speakers, saying 10 words, each word said 26 times. The recordings were taken from the Texas Instruments TI 46 Word Speech Database Speaker-Dependent Isolated Word

Corpus. Our aim is to generate pulse-trains typical of those seen on auditory nerve afferent fibers.

The inner ear is a large-scale parallel distributed processor with approximately 35,000 output channels [34]. The operation of the inner ear is to pulse-encode the presence of features in the filtered sound, with the purpose of supporting subsequent segregation of relevant from irrelevant sound sources [5, 14]. One successful body of work, Uysal, et al. [48], begins with an attempt to model the inner ear output. Leaky integrate and fire neurons (LIFs) preceded by a synapse (HC, [46]) are used to classify vowels. Their 7850 synapses are arranged in parallel and do not interact. Output is the order of neuron firing – the LIFs, however, receive summated input from multiple HCs. The HC output represents probable firing of auditory nerve fibers. The result is that appropriately thresholded LIF firing is more likely when a quorum of HCs output is high, allowing them to be synchrony detectors. Because each LIF is associated with a characteristic HC center frequency the likely mix of formant frequencies emerges, allowing for vowel classification.

However, the hearing ability of cochlear implant patients [30] and of normal hearing subjects listening to spectrally limited speech [11] or to speech-envelope modulated noise [12, 41], is proof that the input filter bank need not duplicate the number (or form) of the filters of a normal ear in order to support word recognition (*ergo* or to research the required neural complexity allowing for word recognition). Auditory fibers respond to sound with a wide variety of characteristics that have been well-described using statistical systems identification [40] derived, and experimentally validated, models (e.g. prediction of tuning curve amplitude dependence, phase locking, and onset/offset response to pulsed tones [9, 53]). This analysis has led to a high-level biophysical model of the acoustic transform that is descriptive through the entire spectral sensitivity of the inner ear [23, 53], and that has been validated in several vertebrate acoustic preparations [24, 38, 54]. This model describes inner ear acoustic waveform transduction as a filter followed by a pulse generator.

For our filter bank, we opted to use a Debauchies-4 level 6 wavelet decomposition, which results in seven channels of spectrally filtered continuous waveforms for input to the pulse encoder [37]. We mimicked two response characteristics of the pulse encoder: onset responses, and phase locking to the filtered-sound. The input pulse encoding is a running threshold,

$$\psi(t) = \begin{cases} k\psi & \text{if } (s_\kappa > \psi); \text{ also, an input pulse is generated,} \\ c\psi & \text{otherwise} \end{cases} \tag{24.1}$$

where, $s_\kappa(t)$ is one component of the wavelet filtered input, k is a constant > 1, and c is the exponential decay per time step that satisfies

$$ky \exp(-T_d/\tau) = y \tag{24.2}$$

where, y is the value of $\psi(t)$ when a pulse is generated, T_d is the desired average inter-pulse interval duration, and τ must be calculated for a given value of k. $T_d = 10msec$; resulting in the average pulse rate in response to noise being 100 per

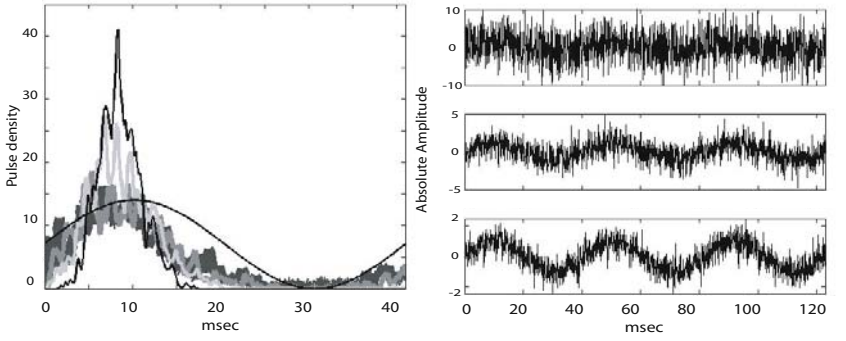


Fig. 24.3 Phase locking. The graphs on the left are peri-stimulus time histograms of the pulse encoder response to a sinusoidal input with added noise over 256 cycles. The sinusoid is $24.41Hz$. Noise is pseudo-Gaussian to approximately $2kHz$. The SNR of the signals, $10\log_{10}(E_s/E_n)$, is 4 (black), 1(light grey), -2 (medium grey), and -4 (dark grey). Three cycles at SNR equal to -4, 1 and 4 are shown in the graphs to the right. Average pulse rate per cycle is approximately 4 for all SNR levels, corresponding to a pulse rate of approximately $100Hz$

second. A value of k near to 1, generates a strong onset response characteristic, and for a noiseless sinusoidal input, also generates a periodic sequence of pulses with the pulses near to the peaks of the sustained sinusoidal input. A larger value of k yields the phase-locked response typical of auditory fibers most sensitive to low-frequency input.

The series of equations 24.1 (one per input channel), with $c = 1$, yields a reduced-channel network similar to [29]. In that study a single layer of neuron models was used to classify the ten French digits. Using the *order* of the first 20 neurons to spike (each neuron representing a spectral channel and one of three fixed thresholds), Loiselle, et. al. accomplished very good true-positive classification rates; and thereby convincingly argued for the merit of developing bio-inspired approaches to speech processing under certain constraints, namely, small and not-necessarily representative, training data sets and limited computational time. We only recently became aware of their research, but in some sense, this chapter answers their challenge to use more realistic biology and in particular, synaptic dynamics, to address speech recognition problems.

24.3 Synapse Connectivity: The Dynamic Synapse Neural Network

The nonlinear dynamic function of isolated biological synapses render them well-suited for passing energy represented by an optimal event sequence vs. other

event sequences [36]. In so far as we can assume a good neural information processor is one that transforms classes of input event sequences into classes of output event sequences, where each output class is a set of similar sequences distinguishable from any given sequence of a different output class, the optimal sequence passing feature of a *single* synapse renders it an unlikely candidate for information processing on two counts. First, there is no guarantee that each input class is a set of event sequences with low-enough variability that each sequence is approximately equal to any other in the class. Therefore the nonlinear nature of a synapse implies there is no guarantee that the output of each input sample maps to an output sequence that is near enough to the average of a recognized output class as to declare passage of a meaningful message. Second, the output of a synapse is 'interpreted' by a neuron as it generates action potentials, the action potential sequence presumably carries the passed information. However, this sequence has no 'negative' bit – the output sequence only consists of 'yes' and 'absence of yes' events (periods of silence). Isolated synapses are good filters, but a network is mandatory to study the role of synaptic dynamics in information processing. The network described in this section is a relatively stiff system of ordinary differential equations. However, we were able to estimate the dynamics of most of this system explicitely using a forward Euler approach (and a small time step, approx. $1/10msec$). The postsynapse however, required implicit solution using Heun's method [21, chapter 14].

Synapses are the "elementary structural and functional unit for constructing of neural circuits" [43]. As such, it is the definition of synaptic operation that ultimately limits both the physical structure and the information processing ability of a given neural circuit. A reasonably general mathematical description of the synaptic functional is,

$$\begin{aligned} u_j(t) &= f(u_j, \Sigma\delta_{\lambda\le t;j}, v_{int}, v_d) \\ v_{int}(t) &= g(\Sigma\delta_{\lambda\le t;int}) \\ v_d(t) &= h(\{u_j\}, v_d) \end{aligned} \tag{24.3}$$

In equation 24.3, u_j is the weight of the j^{th} synapse associated with a small patch of dendritic branch. Weight is the real number value describing the strength or ability to transmit a signal across the synapse. $\Sigma\delta_j$ and $\Sigma\delta_{int}$ are the event sequences (the mathematical equivalent of action potentials in neural tissue) that represent the input to the synapse. The λ are the times of each event in the sequences.[1] v_d is the dendritic branch voltage. The function f describes the j^{th} synapse transform, the function g describes the affect of an interneuron, and the function h describes the function of the dendritic branch. Referring to figure 24.2, synapses (see

[1] The notation, $\Sigma\delta_{\lambda\le t}$ (read as, the sequence of events at times λ less than t), is equivalent to the notation $\Sigma_\lambda \delta_{t-\Delta_\lambda}$ with $\Delta_\lambda = t - T(\lambda)$ and $T(\lambda)$ is the list of event times. Networks not containing dendritic branch function are notated by replacing v_d with v_{soma}, the somatic potential. Networks with no interneurons have $v_{int} = 0$; while networks with arrays of interneuronal interference replace v_{int} with Σv_{int}.

figure inset) in our *DSNNs* are lumped into three groups of seven, $j \in \{1, ...7\}$; with the synaptic group connecting the input to the interneuron having $v_{int}(t) = 0$, i.e. having no interneuronal connectivity onto itself.

24.3.1 DSNN Architecture

We are concerned with determining the computational ability of synapses (rather than the additional computational potential provided by various network architectures) therefore we limit the network to a single layer of synapses connecting the input to the output. These synapses are each considered to be communications channels, so we allow for our network to provide a 'stop transmission' signal prior to synaptic transmission, and a 'receiver modulation' signal at the tail-end of synaptic processing. These signals come in the form of, respectively, a pre-synaptic inhibition via an interneuron and a post-synaptic variable resistance that is sensitive to a parameter shared by multiple post-synapses.

The minimum number of input channels was gleaned from psychophysical literature; most significantly those studies relevant to the hearing of cochlear implant patients. These studies set the lower limit of spectral channels that can support speaker independent speech recognition to five to sixteen (depending on quantization) channels being sufficient to yield $> 90\%$ speech recognition ability for normal hearing subjects [31]. Our network supports seven channels because (after observation) we found that level-6 wavelet decomposition of the speech resulted in three to five channels carrying a significant component of the speech signal; with the remaining decomposition waveforms being amplified recording noise. We assume the minimum number of output channels is two; one that communicates 'Positive ID' $(+)$, and the other that communicates 'Negative ID' $(-)$. We applied the same interneuronal (stop transmission) signal to all presynapses of the network. Feedback inhibition is responsible for sharpening the acoustic filter response in noisy environments [28] and via a complex neural circuit determining the central transmission of complex-sound localization cues [19]. However, this feedback is largely initiated by brain nuclei that are two or more synapses removed from sound transduction and it includes non-auditory information [56]. Wideband feedforward inhibition of the first central synapse of the DCN [1] also plays an important role in peripheral auditory processing [56]; we mimic this functionality in the *DSNNs* by using the input pulse trains as the input to the interneuron. The interneuron do not receive a 'stop transmission' signal.

The resulting network consists of one inter- and two output neurons (Refer to figure 24.2). Each neuron receives a copy of each input pulse train via a synapse, for a total of twenty-one synapses in each network. Each neuron is described by a non-trainable adaptive threshold mechanism. Each input waveform corresponds to a unique event sequence, which is copied three times with one copy being received by a presynapse at each of the three neurons. Finally, our analysis of network responses assumes the output pulse sequences could potentially be input to another neuronal

network which supports an excitation / inhibition algorithm, Difference(Δt), for the purpose of speech recognition.

24.3.2 Synapse Functionality

The operation of chemical synapses [45] allows us to split equation 24.3 into two separable processes: pre- and post-synapse, with the pre-synapse transmitting a chemical signal to the postsynapse via a sequence of amplitude modulated aliquots (neurotransmitter).[2] *In this neuroengineering study of the computational capability of a single layer of synapses, we include synaptic functionality typical of most synapses, but that is not found in the peripheral auditory system.* We primarily draw from two sources in determining our pre-synaptic description: a review of short-term synaptic plasticity [58], and of presynaptic inhibition [50]. We primarily draw from two sources in determining our post-synaptic description: a recent review of glutamate excitation of postsynaptic membranes [49], and a numerical model that has been shown to describe well the post-synapse of an isolated synapse [44].

The amount of neurotransmitter released from the presynapse is proportional to the space-averaged intracellular calcium concentration taken to a power (1 to 4; depending on type of synapse) at the time of release. The process initiating transmitter release is the input action potential, which opens pores in the membrane, allowing calcium to flow into the terminal, and dramatically increasing the intracellular calcium concentration. This increase is of very short duration, therefore, in our model, $u_j(t)$ need be non-zero only at the time of an input action potential. Unidirectional transmission from pre- to post-synapse allows for the weight of the j^{th} synaptic connection, u_j, to carry the specific interpretation of neurotransmitter release magnitude. $u_j(t)$ is now independent of the dendritic branch voltage. These two modifications to equations 24.3 yield,

$$\begin{aligned} u_j(t) &= \begin{cases} f(u_j, \Sigma\delta_{\lambda \le t;j}, v_{int}) & \text{if } t = \lambda \\ 0 & \text{otherwise} \end{cases} \\ v_{int}(t) &= g(\Sigma\delta_{\lambda < t;int}) \\ v_d(t) &= h(\{u_j\}, v_d) \end{aligned} \tag{24.4}$$

The pre-synaptic terminal stores a limited supply of transmitter ready for release; after release, it takes time to replenish this store via uptake, transport, and metabolic mechanisms. Time is also required for intracellular buffers and membrane pump mechanisms to restore $[Ca^{2+}]$ to the resting value after action potential induced influx. During that time the residual calcium adds to any new calcium entering the terminal, resulting in the new incoming action potential releasing more neurotransmitter than the previous one did. This effect is mediated by a reaction-diffusion equation that can saturate. This process is called facilitation and we describe a fast (f_{fast}) and slow (f_{slow}) component of facilitation in our presynapses.

[2] Retrograde communication is biologically possible but is not modeled in our equations.

Table 24.1 Pre- and Post-synapse parameter values or ranges. K_x are unitless, τ values are in msec. The *NMDA* gate has a rise time of 3.5*msec* and decay time of 8.27*msec*

Fast Facilitation	$K_{f,fast} \in (0,100)$;	$\tau_{f,fast} \in (0,12.5)$		
Slow Facilitation	$K_{f,slow} \in (0,200)$;	$\tau_{f,slow} \in (0,125)$		
Interneuron Inhibition	$K_{int} \in (-200,0)$;	$\tau_{inhib} \in (0,50)$;	$2 \times \tau_{inhib}$	
Postsynaptic Membrane	$V_\infty = -70$;	$\tau_m = 1.372$;	$E_{reversal} = 140$;	
AMPA	$\tau_{o>>c} = 0.56$;	$\tau_{c>>*} = 15.76$;	$\tau_{*>>o} = 0.225$;	$K_{AMPA} = 0.83$
NMDA	$\tau_1 = 35.0$;	$\tau_2 = 5.0$;	$\tau_3 \in (0.1,7.5)$;	$\tau_4 = 1.0$
	$\tau_5 = 1.0$;	$\tau_6 \in (0.5,50)$;	$\tau_7 = 1.0$;	$\tau_8 = 3.75$;
	$\tau_9 = 1.5$;	$\tau_{10} \in (1,15)$;	$\tau_{11} = 250.0$;	$K_{NMDA} \in (900,15000)$

Synapses have processes that can reduce the width and height of incoming action potentials. These processes reduce the amount of calcium entering the presynapse during the action potential timecourse; which has the subsequent affect of reducing the amount of neurotransmitter released in response to the action potential as well as reducing the amount of residual calcium remaining at the time of the next action potential. We do not model these modulations as they are a relatively insignificant.[3] The primary process responsible for adjusting the amount of calcium entry into the presynapse (up to 85% of the total possible modulation in guinea pig and rat hippocampal CA3-CA1 synapses) is a *GABA*-ergic type-B synapse [47]. We describe this using a negative going alpha-function initiated at the time of an interneuron action potential (v_{int}). The constraints discussed here are applied to equation 24.4 resulting in the more descriptive equations 24.5 through 24.10.

$$f_{fast}(t) = \Sigma\delta_{\lambda \le t;j}(t) * K_{f,fast} \exp\left(-t/\tau_{f,fast}\right) \tag{24.5}$$

$$f_{slow}(t) = \Sigma\delta_{\lambda \le t;j}(t) * K_{f,slow} \exp\left(-t/\tau_{f,slow}\right) \tag{24.6}$$

$$\begin{aligned} v_{int}(t) = {} & \Sigma\delta_{\lambda < t;int}(t) * K_{int} \exp\left(-t/\tau_{int,slow}\right) \\ & -\Sigma\delta_{\lambda < t;int}(t) * K_{int} \exp\left(-t/\tau_{int,fast}\right) \end{aligned} \tag{24.7}$$

$$A_r(t) = 1 + f_{fast} + f_{slow} - v_{int} \tag{24.8}$$

$$u_j(t) = \begin{cases} 0 & \texttt{if } (A_r < \psi_R) \\ A_r & \texttt{if } (A_r > \psi_R) \texttt{ AND } (A_r < N_{store}) \\ N_{store} & \texttt{if } (A_r > \psi_R) \texttt{ AND } (A_r \ge N_{store}) \end{cases} \tag{24.9}$$

$$N_{store}(t) = \begin{cases} C_N N_{store} + K_N & \texttt{if } (A_r < \psi_R) \\ N_{store} - u_j & \texttt{if } (A_r \ge \psi_R) \texttt{ AND } (N_{store} \ge u_j) \\ 0 & \texttt{if } (A_r \ge \psi_R) \texttt{ AND } (N_{store} < u_j) \end{cases} \tag{24.10}$$

[3] These contributions can be modeled by setting the value of K_r in equation 24.8 to a function modeling the action potential shape modulation processes and attenuating each input impulse accordingly.

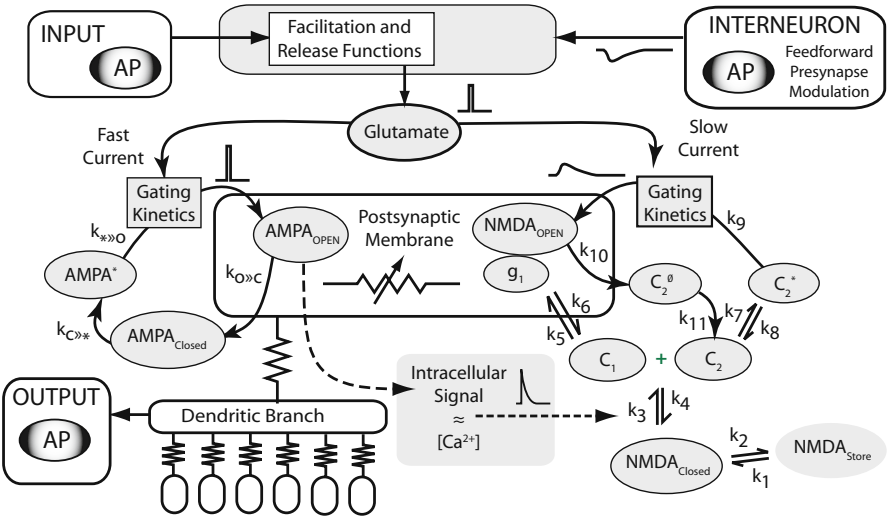


Fig. 24.4 Diagram of a single synapse embedded in the *DSNN* of Figure 24.2. To the upper left is one of seven input sequences ($\Sigma\delta_{\lambda\leq t;j}$ of equation 24.4). The remaining six input sequences have identical synapses onto the dendritic branch; only the postsynaptic membrane of these connections is drawn (bottom). To the upper right is the interneuronal input ($v_{int}(t)$ of equation 24.4). The circled Glutamate is $u_j(t)$ in equation 24.4. The Dendritic Branch equation is $h(\{u_j\}, v_d)$ in equation 24.4. k_x correspond to the τ_x of Table 24.1.

Variables in equations 24.5 to 24.10 are defined consistent to the presynaptic biology presented above. The convolution operation is indicated by an asterisk. ψ_R is an imposed threshold for synaptic transmitter release. Replenishment of the neurotransmitter store is a linear function ($+K_N$), however, the store has an exponential cycling decay ($C_N N_{store}$), i.e. vessicles that are ready for release but do not get released may become absorbed back into the intracellular matrix.

Analytic study of isolated synapses [4] and synaptic networks [55] reveal them to be highly complex processes. However, for simplicity of presentation, the complexity of the postsynaptic model is described herein by figure 24.4, where several interacting processes are drawn: AMPA, NMDA, and Dendritic Branch. All three processes are simplified functional descriptions of their biological counterparts [51, 52], rather than being an accurate kinetic representation. Both AMPA and NMDA are described by state transition models; the amounts of stuff represented is conserved. At equilibrium all AMPA is in the AMPA*, or activated, state. An impulse of glutamate, results in a percentage of AMPA* making a transition to AMPA$_{OPEN}$, changing the variable resistance of the postsynaptic membrane. The fraction of AMPA$_{OPEN}$ quickly falls, as it is converted to AMPA$_{Closed}$, which is then, subject to a slower time constant, converted once again to AMPA*.

Presynaptic activity is generally low enough to prevent saturation of the AMPA channel; the postsynaptic resistance change due to AMPA opening is relatively consistent. The model of NMDA expresses long term dynamics. We do not model learning with this network, eventually, all NMDA will be at equilibrium between two states, NMDA_{Closed} and NMDA_{Store}. However this transition is slow. k_3 is a function of the postsynaptic current, therefore, until a sufficient amount of glutamate is released from the presynapse, openning AMPA channels, a sufficient amount of NMDA cannot be converted to C_1 and C_2, in readiness to affect postsynaptic conductance. A fraction of C_1 is converted to g_1, slowing the conversion back to NMDA_{Closed}. A fraction of C_2 is activated in readiness to interact with subsequent glutamate pulses. The gating kinetics of NMDA are much slower than for AMPA; once initiated, the gate undergoes a slow rise and slower decay. The result is that the maximal conductance change due to the NMDA process happens after a delay from the initiating glutamate pulse (that resulted in transition to C_2^* and g_1). Since the initial C_2^* will have been reverting during that delay, first to C_2 and then to the closed state, for the NMDA process to affect a substantial change in postsynaptic conductance, pulses of glutamate must be received by the postsynaptic membrane during the delay period so that the amount of C_2^* remains high. k_{11} is small, allowing C_2^0 to act as a buffer that results in previously opened NMDA to maintain the level of C_2, and thereby, C_2^*. The Dendritic Branch process is a numerical buffer, necessary to maintain system stability.

24.4 Synapse Optimization

Network I/O was trained toward serving both a TW classification task and a TW dynamic modeling task. This was accomplished using a GA. Mutation rate was adjusted throughout training to increase or decrease the degree to which new generation parameter choice was focused on a local minimum. These adjustments are weakly analogous to a simulated annealing process (see [6, 8, 39]). Generational processing time was decreased by estimating the fitness of each network on a subset of the available input training set; highly fit networks were re-evaluated using a larger subset. Subset size was increased as training progressed and as relative generational improvement slowed. Input subset choice was skewed toward those input samples correlated with the lowest fitness scores – thereby focusing optimization at the separatrix between TWs and non-TWs. The non-TW decisions used to fill the Confusion Matrix were weighted by a factor on the interval (0,1], equal to the current sample output energy divided by the least responsive TW output energy. Because of the preponderance of non-TW input, after 100% classification was reached, but while optimization continued toward filtering-out non-TW input, the calculated information capacity continued to increase as the percentage of non-TWs eliciting high output energy decreased. Our preference for using samples *not* associated with high-scoring input sets had the effect of concentrating optimization for classification ability on those non-TW samples which produced larger than

average integrated output and were misclassified (non-TW input producing low-energy output were filtered out of consideration) and on TW samples that produced low output energy (as these would lower the threshold of discernment resulting in some non-TW misclassifications being counted that would otherwise have been 'filtered out' of consideration). These points are discussed below.

24.4.1 Biased Selection: Word Micro-environment

For computational efficiency, it is essential that each individual *DSNN* performance be probed using as small a subset of the corpus as possible. However, experience reveals that some input examples are much more easily classified than others, i.e. training subsets containing these examples yield much higher fitness scores than other subsets (for a given set of *DSNN* parameters); and because the *DSNN* transform is highly non-linear, every set of *DSNN* parameters has a different 'optimal' training subset. Therefore, poor choice of training subset can result in *DSNN* parameter sets being given falsely high fitness scores, encouraging evolution in a sub-optimal direction. We address this situation directly by limiting the potential fitness increase attributed to individual input samples, and indirectly via intra- and inter-generational testing of high-scoring *DSNN* parameter sets; 'Biased Selection' in Figure 24.1.

Our initial goal is to 'cull out' those input examples that are likely to result in uncharacteristically high fitness scores – before these examples have a chance to bias the optimization algorithm into a poor choice of state space locality. After this goal is reached, we can then attempt to determine the smallest input subset size that will accurately reflect the outcome of the entire corpus. Initially, the probability of choosing any given example is uniform over the entire corpus. This probability is adjusted whenever a micro-environment score is generated that is greater than the current best macro-environment score. The number of times that an example contributes to such a high score is tabulated, these numbers are used to create a sequence of intervals,

$$[1,n_1],[n_1+1,n_1+n_2],\ldots,\left[\Sigma_{j=1}^{N-1}n_j+1,\Sigma_{j=1}^{N}n_j\right].$$

Choosing an example reduces to generating a number from the uniform distribution on $[1,\Sigma_{j=1}^{N}n_j]$, and matching that number to the appropriate interval. The percentage of TW and non-TW examples chosen for input to a given *DSNN* increases throughout training as a function of the highest fitness score attained. If two generations passed without improvement, the number of samples increases by one seventh of the difference between the current number and the total number of example words of that class in the corpus. The purpose is to concentrate optimization at the separatrix of TW and non-TW class samples; rather than to optimize parameters to a small set of examples that is assumed *apriori* to best describe the input classes. If twenty-one generations passed without improvement, then the entire optimization was restarted and the number of samples chosen to fill the training set for both TW and non-TW word classes was reduced to 3.

24.4.2 Objective Function: Fitness Evaluation

Each member, w_i, of the training subset is or is not a TW, (respectively, $\in$ or $\notin$). The optimization routine observes the "Positive ID" and "Negative ID" pulse sequences (respectively, S_i^+ and S_i^-) of figure 24.2. The fitness function integrates S_i^+ and S_i^-, to yield the corresponding pairs, $\left(E_i^+, E_i^-\right)$. Finally, the classification result for each w_i is the $sign(E_i^+ - E_i^-)$. So that $\{w_i\}$ corresponds to an ordered list of length, I.

$$\text{Training Subset} = \{w_i\} \Rightarrow \left\{\left(<\in, \notin>, E^+, E^-, <+, ->\right)_i\right\} \tag{24.11}$$

We generate four histograms from the E-sequence data: $H_\in^+$, $H_\in^-$, $H_\notin^+$, and $H_\notin^-$; with respectively, mean values, $\bar{m}_\in^+$, $\bar{m}_\in^-$, $\bar{m}_\notin^+$, and $\bar{m}_\notin^-$. It is useful for the immediate discussion to define r for each histogram as, for example: $r_\notin^- = \text{rms}(E_{\notin,i}^- - \bar{m}_\notin^-)$. We also used the sequence in 24.11 to generate the Confusion Matrix [22], and used standard operations on the Confusion Matrix for determining how well a particular *DSNN* was able to classify $\{w_i\}$. However, during training the contribution of the non-TWs to the matrix was normalized.[4]

Normalized (Cost Weighted) Confusion Matrix, $P = \begin{vmatrix} P_{(\in,+)} & P_{(\in,-)} \\ P_{(\notin,+)} & P_{(\notin,-)} \end{vmatrix}$, such that,

$$\forall w_i \Rightarrow \notin, \text{ and for } \theta^+ = \min[E_i^+ \mid w_i \Rightarrow \in], \tag{24.12}$$

$$P_{(\notin, sign(E_i^+ - E_i^-))} \stackrel{+}{=} \begin{cases} \left(E_i^+ / \theta^+\right) & \text{if } E_i^+ < \theta^+ \\ 1 & \text{otherwise} \end{cases} \tag{24.13}$$

$$\text{and } \forall w_i \Rightarrow \in, \quad P_{(\in, sign(E_i^+ - E_i^-))} \stackrel{+}{=} 1.$$

Perfect filtering (discrimination of TWs by comparing E_i^+ to an absolute threshold) and classification[5] (discrimination according to $sign(E_i^+ - E_i^-)$) is achieved if training accomplishes the following two-part goal:

$$\textit{goal 1:} \quad \text{Complete overlap of } \left(\sqrt{I} * \bar{m}_\notin^- \pm r_\notin^-\right) \text{ and } \left(\sqrt{I} * \bar{m}_\in^- \pm r_\in^-\right) \tag{24.14}$$

$$\text{with, } \quad r_\in^- \rightarrow r_\notin^- \rightarrow \text{very small} \tag{24.15}$$

$$\textit{goal 2:} \quad \text{Complete separation of } H_\in^+ \text{ and } H_\notin^+ \tag{24.16}$$

$$\text{with, } \quad \bar{m}_\notin^+ << \bar{m}_\in^- \approx \bar{m}_\notin^- << \bar{m}_\in^+ \tag{24.17}$$

[4] The Confusion Matrix used here is a tabulation of all decisions; rather than a matrix of true- and false-, positive and negative rates.

[5] If both $min[H_\in^+] - max[H_\in^-] > 0$ and $max[H_\notin^+] - min[H_\notin^-] < 0$ are true, perfect classification is guaranteed. However, this constraint is much stricter than necessary because classification is determined from individual samples of $w_i \Rightarrow \left(<\in, \notin>, E^+, E^-\right)_i$. We also note, anecdotally, that generally, $\forall i$, $E^+ \approx E^-$; but that the magnitude of $E^{<+,->}$ depends on speaker-specific input factors – for example the sex or dialect of the speaker. Therefore, the strict population constraint on E^+ and E^- necessary to guarantee perfect classification is not a useful measure for network performance as it is almost never realizable.

Equations 24.14 encodes an understanding of biological circuits: stability in terms of event rate (energy) is a local operating imposition on global functionality.

Whereas all $w_i \Rightarrow \in$ represent the same word, each $w_i \Rightarrow \notin$ represents one of 9 words. Certainly, at the onset of training, when $I = 3$ for both $\in$ and $\notin$ classes, $H^{<+,->}_{<\in,\notin>}$ cannot be represented faithfully by mean and variance values (equations 24.14 to 24.17). We declared the following intervals,

$$(\min, \max)^-_{\in} = (\min[E_i^- \mid w_i \Rightarrow \in], \max[E_i^- \mid w_i \Rightarrow \in])$$
$$(\min, \max)^-_{\notin} = (\min[E_i^- \mid w_i \Rightarrow \notin], \max[E_i^- \mid w_i \Rightarrow \notin]),$$

and using the above declarations, we determined the percentage, $z_\in$, of TW samples that did not have E_i^- within $(\min, \max)^-_{\notin}$, and the percentage, $z_\notin$, of non-TW samples that did not have E_i^- within $(\min, \max)^-_{\in}$. We used the average of $z_\in$ and $z_\notin$ to estimate progress toward the goal stated in 24.14.

$$\text{score}_{(H^-)} = a_{(H^-)} + (b_{(H^-)} - a_{(H^-)})(1 - \frac{z_\in + z_\notin}{2}) \tag{24.18}$$

with $b_{(H^-)} > a_{(H^-)}$.

We determined $y_\notin$, equal to the percentage of non-TWs that yielded $E^+ > \min[E_i^+ \mid w_i \Rightarrow \in]$, and scored progress toward goal 24.16 using,

$$\text{score}_{(H^+)} = b_{(H^+)} - a_{(H^+)} y_\notin \tag{24.19}$$

$$P_{f/f} = \begin{vmatrix} 25 & 1 \\ 0 & 102.3 \end{vmatrix} \qquad P_{f/\forall} = \begin{vmatrix} 251 & 175 \\ 700 & 3270.0 \end{vmatrix} \qquad P_{\forall/\forall} = \begin{vmatrix} 386 & 31 \\ 116.4 & 2922.4 \end{vmatrix} \tag{24.20}$$

$$C_{f/f} = \begin{vmatrix} 25 & 1 \\ 0 & 234 \end{vmatrix} \qquad C_{f/\forall} = \begin{vmatrix} 240 & 164 \\ 60 & 3655 \end{vmatrix} \qquad C_{\forall/\forall} = \begin{vmatrix} 384 & 29 \\ 108 & 3606 \end{vmatrix} \tag{24.21}$$

Three Confusion Matrices are displayed in equation 24.20: $P_{f/f}$, for training on words from a single female speaker, $P_{f/\forall}$, the result of applying all 16 speaker's samples to the *DSNN* represented by $P_{f/f}$, and $P_{\forall/\forall}$, for training on all 16 speaker's samples. $P_{f/f}$ is produced from 260 samples, 26 of the word 'repeat'; $\Sigma_{f/f,\notin} = 102.3$ indicates the high degree of filtering applied to the non-target word output, as evidenced by complete separation of the $H^+_\in$ and $H^+_\notin$ in figure 24.5, left. $P_{f/\forall}$ and $P_{\forall/\forall}$ are discussed in section 24.5. Non-weighted classification is tabulated in equation 24.21. There are 413 samples of the word repeat but $\Sigma_\in P$ and $\Sigma_\in C$ for the right four matrices are not necessarily equal to 413 due to how ambiguity is logged. For P-matrices, ambiguous decisions are $(+$ **and** $-)$, for C-matrices, ambiguous decisions are **not** $(+$ **and** $-)$.

For the normalized Confusion Matrix, P,

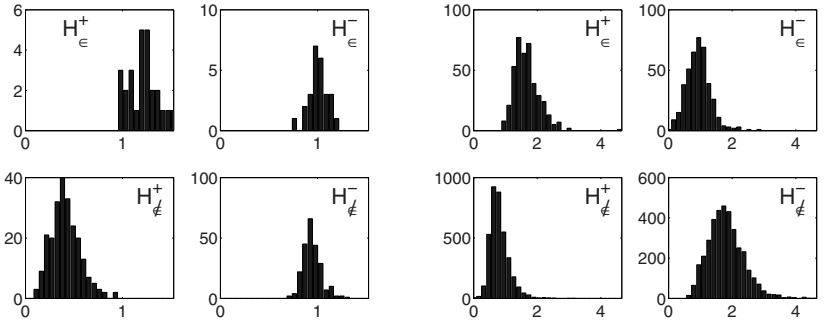


Fig. 24.5 Example population output statistics used for fitness scoring. The four *PDFs* on the left are the output of training on a single female speaker; the four *PDFs* on the right are for training using all 16 speakers. The Confusion Matrices are given in the boxed equations 24.20 and 24.21

$$\text{score}_P = \begin{cases} -P_{(\notin;-)}\left(\frac{b_p}{\Sigma_- P}\right) & \text{if } \Sigma_+ P = 0, \text{ and} \\ P_{(\in;+)}\left(\frac{a_P}{\Sigma_+ P} + \frac{2a_p + b_p}{2\Sigma_\in P}\right) + P_{(\notin;-)}\left(\frac{1}{\Sigma P_{\notin}}\right)\frac{b_p}{2} & \text{otherwise.} \end{cases} \tag{24.22}$$

In equation 24.22 the sum of all output represented in P is ΣP, the sum of all TW output is $\Sigma_\in P$, and the sum of all output that is classified as in-class is $\Sigma_+ P$. Specific entries in P are notated by the subscripted pair. Equations 24.18, 24.19, and 24.22, are summed by the Objective Function of figure 24.1 to determine the fitness score of each *DSNN*. The range of these functions, and therefore the balance of their importance toward parameters optimization, was determined by adjusting the values of $b_{(H^-)}$, $a_{(H^-)}$, $b_{(H^+)}$, $a_{(H^+)}$, b_p and a_p.

24.4.3 Score Function Rational

The Objective Function tested individual fitness by balancing signal discriminability (equation 24.19) and classification ability (equation 24.22), with system stability (equation 24.18). This is illustrated in figure 24.6.

At the onset of evolution, networks may express any of the functionalities described in figure 24.6; however, most commonly, optimization first progressed from the bottom left corner of the score-space, toward the circled 150. The ideal behavior of the network represented at that point is illustrated by the drawing at the bottom right of figure 24.6. Such a network would have good overlap of the two H^- histograms, and good separation of the $H^+_\in$ and $H^+_\notin$, with $H^+_\in > H^+_\notin$. Initial optimization is aided by the weighting function on P (equation 24.13). This equation has the affect of scoring Perfect No-machines better for greater separation of $H^+_\in$ and $H^+_\notin$, encouraging optimization toward networks with $H^+_\in >> H^+_\notin$. This initial phase of evolution occurs rapidly, generally within three to six generations. At the end of this phase, the population of networks consists of some classifiers, but mostly of Yes-machines and No-machines, with these three types of networks

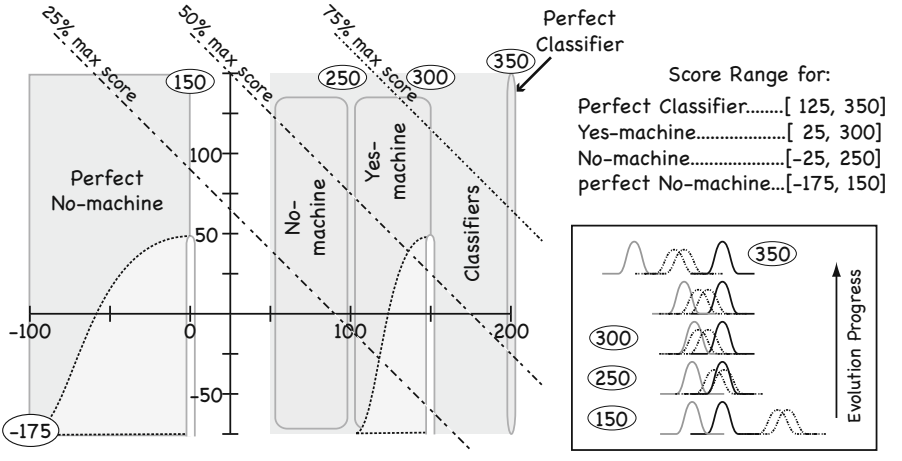


Fig. 24.6 Score values as they relate to network function. Ordinate = $\text{score}_{H-} + \text{score}_{H+}$, abscissa = score_P; fitness = abscissa + ordinate. Perfect No-machines are networks with $\Sigma P = \Sigma_-$; No-machines are networks with $\Sigma_+ P / \Sigma P \to 0$; Yes-machines are networks with $P_{(\notin;-)} / \Sigma_{\notin} P \to 0$ and $P_{(\in;+)} / \Sigma_{\in} P \to 1$; Classifiers are networks that have most weight in P on the diagonal; a Perfect Classifier has strictly diagonal P. The grey area of the plane represents the networks that can be generated; the light grey areas are highly unlikely. The range of score for any given type of network is given in the table to the top right; there is a bound on the lower right side of the Perfect No-machine and the Yes-machine score-spaces. The circled numbers are the score of idealized networks and where those networks lie in the plane; the graphs boxed to the bottom right are their ideal behavior (explained in the text). The graphs represent relative placement of, $H_{\in}^{-}$ and $H_{\notin}^{-}$ (dotted), $H_{\notin}^{+}$ (grey) and $H_{\in}^{+}$ (black). The score ranges depicted here are for: $a_{H^-} = -50$, $b_{H^-} = 50$, $a_{H^+} = 125$, $b_{H^+} = 100$, $a_P = 50$, $b_P = 100$

having respectively better $H-$scores. There is a 'quantal' leap in score as a small change in parameter (similar $H-$scores) converts a network from a No- to a Yes-machine, and from a Yes-machine to a classifier. This is apparent in figure 24.7. The other circled numbers in figure 24.6 represent ideal networks toward which evolution progresses within the score-space of each network type. The relative placement of the H-histograms of these ideal behaviors are shown in the graphs to the bottom right. Evolutionary force between network types is in the direction shown. Within each network type the evolutionary force along the score_P axis is proportional to, $P_{(\notin;-)} / \Sigma_- P$ for Perfect No-machines, and $P_{(\in;+)} / \Sigma_+ P$ for No- and Yes-machines. Therefore, for both Perfect No- and Yes- machines, evolutionary force along the score_P axis is in the direction of networks with $H_{\in}^{+} >> H_{\notin}^{+}$. For No-machines, evolutionary force is more complicated: $H_{\in}^{-}$ is pushed leftward of $H_{\in}^{+}$ and $H_{\notin}^{-}$ is pushed rightward of $H_{\notin}^{+}$ (score_P), while overlap of $H_{\in}^{-}$ and $H_{\notin}^{-}$ is maintained (score_{H^+}). The roughly equivalent significance of these two forces results in an upward diagonal evolutionary force, that encourages development toward ideal Yes-machines and classifiers, as pictured in the boxed inset of figure 24.6. The light grey 'highly

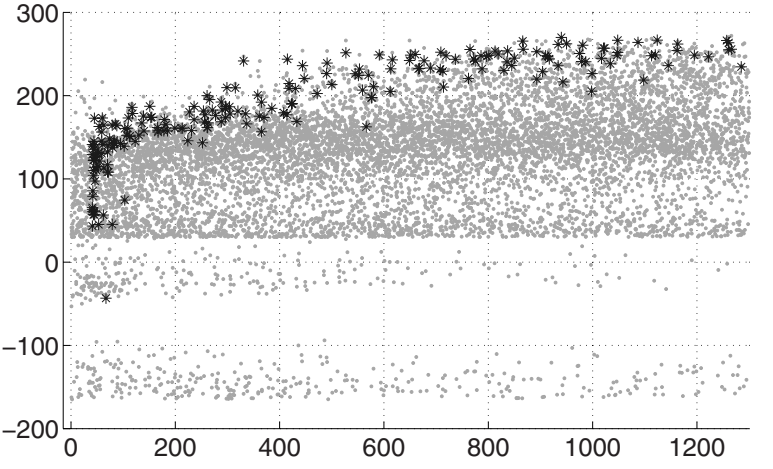


Fig. 24.7 Convergence progress for 7 *DSNNs* trained on all speakers to the target 'repeat' are overlayed. Every generated *DSNN* Micro-Environment score is plotted with a grey dot. Those *DSNNs* that merit testing on the complete Word Environment have score plotted with a black asterisk. The stratification of score values is explained in section 24.4.3. First generation micro-scores are not tabulated, second generation micro-scores do not initiate macro-testing

unlikely' areas of the Perfect No- and the Yes-machines score spaces is a result of the rightward evolutionary force along the score$_P$ axis: as the separation of $H_{\in}^{+}$ and $H_{\notin}^{+}$ increases, the increasing value of $P_{(\in;+)}/\Sigma_{+}P$, requires that $H_{\in}^{+}$ and $H_{\notin}^{+}$ become increasingly separated, so that both score$_P$ and score$_H$ are expected to increase together.

The familiar operations on confusion matrices [22] are,

$$\begin{aligned} \text{precision} &\cong P_{(\in;+)}/\Sigma_{+}P \\ \text{sensitivity; true positive rate} &= P_{(\in;+)}/\Sigma_{\in}P \\ \text{specificity; true negative rate} &\cong P_{(\notin;-)}/\Sigma_{\notin}P \\ \text{accuracy} &\cong (P_{(\in;+)} + P_{(\notin;-)})/\Sigma P \end{aligned}$$

The weights applied to the $\notin$ row of the Confusion Matrix in equation 24.13 render some of the calculations above approximate. However, in so far we can accept the above equations, the score$_P$ function is a measure which balances the relative importance of precision, sensitivity, specificity, and (due to the fractional representation of the overwhelming representation of words in the non-target class) accuracy – exactly what one would expect from a task-specific measure of classifier performance.

24.4.4 Genetic Algorithm: Mutation, Elitism, and Micro-environment

The mutation rate, p, number of cloned individuals, z, and population size, $K+r$, per generation were adjusted to restrain the GA from focusing on a local minimum prior to determining the appropriate probabilities of choosing specific words to represent the entire corpus. At the outset of training we assume that the generated scores are only loosely indicative of the true fitness of an individual (because the input subset does not yet emphasize the TW-nonTW separatrix) and that the most fit DSNNs have parameters that are sufficiently different as to describe local spaces far removed from each other. Therefore, initially we have as a goal to keep as many of these highly fit individuals as possible and to verify individual scores by re-scoring. This is accomplished by initially setting $z = 40$, and generating $r = 32$ new individuals. Mutation rate is initially high, meaning that the 32 new individuals represent novel genetic stock i.e. are unrelated to the cloned 40.

When the best score achieved is on the range $1/10$ to $3/4$ of the best score possible, z and r are inversely proportional to the ratio of the historical best score to the theoretical best score on the range of $(40,1)$ and $(32,1)$, respectively. For best score achieved greater than $3/4$ of the best score possible, $z = r = 1$. Similarly, mutation rate was a decreasing function of the historical best score. For the initial high mutation rate, $p_g = 0.5$, and the low mutation rate used after the GA has settled into a satisfactory local state space, $p_l = 0.05$, the equation defining the current mutation rate p, is,

$$\begin{aligned} q &= 1 - \log_{10}\left(1 + 9\left(\frac{\text{score}_{best} - 0.1}{0.1 * \text{score}_{max}}\right)\right) \\ p &= p_l + q(p_g - p_l) \end{aligned} \tag{24.23}$$

The duration of high mutation rate value is intended to coincide with the duration required for the Biased Selector to determine an initial distribution of probabilities for selecting Micro-Word-Environments that emphasize the separatrix between TW and non-TWs. In this sense, we have built into our optimization a self-determined initialization phase. After this phase has passed, if more than 21 generations pass without improvement (Micro-Word Environment scores are less than the historical best score) then the GA is re-initialized: mutation rate is set high, micro-word environment size is set to 3, and $z = 40$ and $r = 32$.

There are two types of highly fit *DSNNs* at the end of micro-environment testing, those with fitness high enough to elicit re-scoring using the macro-environment prior to a new generation being generated, and those that are high enough scoring to be cloned, but not fit enough to be re-scored prior to generation of a new set of *DSNNs and* word micro-environments being generated. Those that are high enough to be saved, are retested in the next generation with a new micro-environment. Macro-environement re-scoring may reveal the current set of *DSNN* parameters to be a historical best, but most often a significantly lower score is generated and those parameters are removed from the to-be-saved parameter list (and another *DSNN* is

substituted). The threshold to elicit macro-environment testing is the historical best score. Over the course of evolution, the micro-environment size increases, making it less likely to produce scores that elicit macro-environment testing. Operationally, there are two optimization loops running, one fast-loop which generates potential best parameter sets, and one slow-loop that tests the potential best parameter sets and resets various parameters of the fast-loop (threshold for macro-testing, elitism, population size, and mutation rate values).

24.5 Output Description and Analysis

The *DSNN* output is two event sequences, respectively, S^+ and S^-. We log the event times, sequentially bin them, and subtract the value of the $(-)$ from the corresponding $(+)$ bin, to generate a single discrete output function. Each input to the *DSNN* is an isolated word, and each output is a function in time. We take the output functions that correspond to the same input word, average them, and it is this average response to each word, and the differences or similarities between them, that are presented in this section. The input words are labeled as in the TI46 Word database:

ST = start	GO = go	RB = rubout	ER = erase	YS = yes
SP = stop	NO = no	HP = help	EN = enter	RP = repeat

DSNNs trained to a particular TW are noted by subscripting the appropriate label from the list above. For example, $DSNN_{ST}$ is the DSNN trained for TW = 'start'. The difference function defined above is noted by the TW label of the input word label, for example, the average difference response elicited by input examples of 'erase' from $DSNN_{RP}$ is,

$$RP(ER) = \forall w_i \Rightarrow ER,\ \exp\left[DSNN_{RP}(S^+ - S^-) \mid \Delta t\right] \tag{24.24}$$

24.5.1 Integrated Responses

In this subsection we describe the energy output of the trained *DSNNs* as a function of input word (with $\Delta t =$ the duration of the longest word in $\{w_i\}$). In the following subsections we describe the energy output of the trained *DSNNs* as functions of input word and time, as described by equation 24.24 (with $\Delta t = 20msec$).

The histograms of output energy for $DSNN_{RP}$ as a function of input word are plotted in figure 24.8. The top row is indicative that, for a relatively small speaker-specific optimization, the network converges to the goals stated in equations 24.14 to 24.17 (left two plots). In particular, E^+ for non-TWs is less than E^+ for TWs, and there is a tight interval within which E^- falls for all input. Also, this data indicate these goals support classification (third column). And finally, this data indicate that such a trained network will have an optimal input class, as measured by average total output energy in response to that word input versus other input (fourth column).

In the second row of figure 24.8, all sample words are applied to the *DSNN* of the first row (26 samples of each word from 8 male and 8 female speakers;

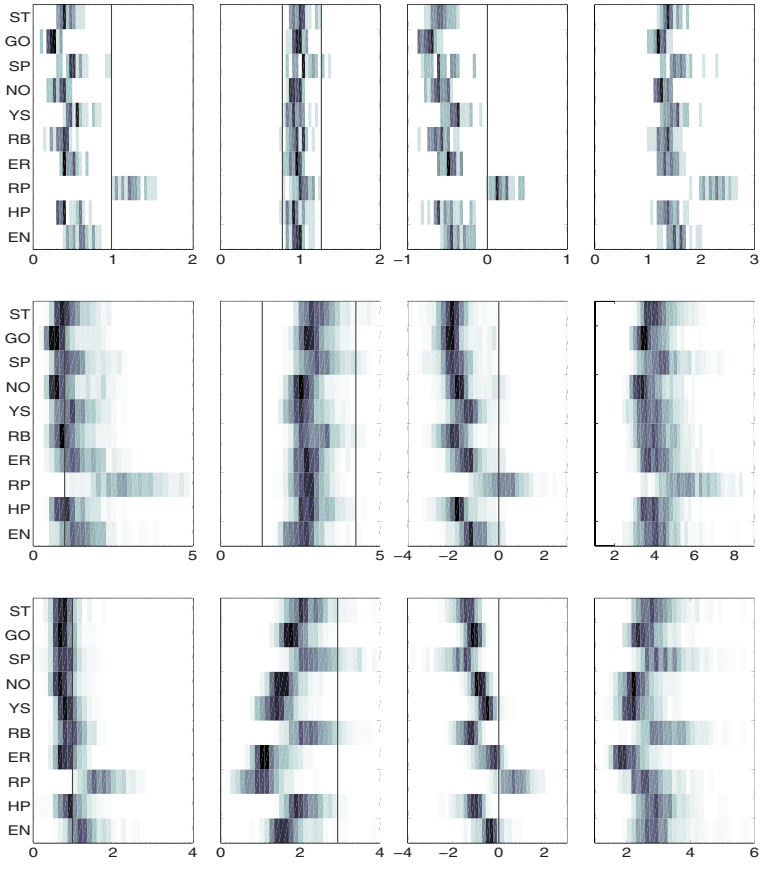


Fig. 24.8 Population output statistics of $DSNN_{RP}$ broken down into target word and non-target word subclasses: for training on a single female speaker (top four plots), for that same *DSNN* with sample words from all speakers applied to it (middle row), and for training on all 16 speakers (bottom four plots). This data is also pictured in figure 24.5. Here, the height of each histogram is indicated by shade intensity. The input labels for each histogram are printed on the ordinate; the abscissa corresponds to the response energy relative to the $min\left[E^+;\forall w_i \Rightarrow \in\right]$ (see equation 24.11). The columns of graphs are respectively, H^+, H^-, pdf$(E^+ - E^-)$, and pdf$(E^+ + E^-)$

approximately 4160 samples including the 260 used during training). The pattern of output energy versus input word remains similar. However, we note that the interval containing E^+ for non-TWs is increased, and the increased number of samples allow us to visualize more than one optimal output class – in this case the word, stop. From the classification results in table 24.2 middle row, the word stop is correctly classified at greater than 99%, indicating that this network has optimal Positive ID input, and optimal Negative ID input. Classification of all non-TWs is

Table 24.2 Positive-classification rates for the *DSNNs* of Figure 24.8. The true input word is given at the top of each column, the target word is *RP*. First row: training data for the single female speaker *DSNN*. Second row: validation of the single speaker *DSNN* on all 16 speakers. Third row: training data for a *DSNN* trained using examples from all 16 speakers. The *P*-matrices used during training corresponding to the first and third rows are given in equations 24.20. The last row is the average positive classification rate for each subclass during validation of 15 *DSNNs*, each one having been trained with a different speaker. The data given in the last row indicate that regardless of which of the 16 speaker's data sets is chosen to train the initial *DSNN*, the input words filtered will be the same, and the classification task will be reduced to the same subset of non-target words

ST	GO	RB	HP	SP	NO	YS	ER	EN	RP
-	-	-	-	-	-	-	-	-	96.2
-	-	0.2	0.5	0.7	4.3	2	2.4	4.3	57.8
-	-	-	-	-	-	3.6	11.1	11.1	93.0
0.21	0.34	0.16	0.24	-	1.56	4.95	6.18	14.31	70.44

validated above 95%; but noticeable misclassification rates are generated for four of the non-target words; the remaining non-target words being classified at near 100% rate. The classification rate for 'repeat' has fallen to 57.8% – certainly, adjusting the classification threshold leftward would better balance the error rates for TW and non-TW classes, but here we are interested in output patterns more than classification statistics according to equation 24.11 with an added variable threshold (Receiver Operating Characteristic).

The third row of plots is for a *DSNN* trained on all word samples. In some sense, these plots represent the ideal response to which validation data ought to be compared – unfortunately, training these systems is intolerably slow. The *DSNN* pictured in row one was achieved overnight on a desktop PC; whereas, *DSNNs* trained on all 4140 input samples require two to four weeks of dedicated processing time on seven comparably equipped computers. Indeed, one of the motivations for this work was to determine algorithms for decreasing the compute- and real-time required to optimize biologically based small-set speech classifiers. Row three of figure 24.8 is demonstrative of an important output characteristic: by defining classification in terms of equation 24.11 the optimal filtering properties of a *DSNN* do not necessarily coincide with the actual classification abilities. This is evidenced by the output energy for the words, 'enter' and 'erase', both of which have 11% error rate (Table 24.2). The word enter produces a relatively significant total output energy, but the word erase has significantly depressed output relative to all other word input (Row 3, Pane 4).

The result shown in figure 24.8 and table 24.2 is typical, and indicative of the following general patterns found in most of our trained *DSNNs*. First, optimization toward the TW versus a set of nine non-TWs is in effect optimization toward distinguishing the TW from a *subset* of the non-TWs. Some non-TWs are filtered-out (rendering their classification a moot issue), others are easily distinguished from the TW. Therefore it is possible to limit the optimization task to a corpus of manageable

size, without making prior assumptions regarding on which words, or particular examples of those words, the optimization should focus. Second, independent of classification based on differential network output, there is a subset of optimal input words, in the sense that the expected total network excitation in response to these words is greater than for others. Third, the overall pattern of integrated output after training on input words from a single speaker is similar to the pattern evoked from that *DSNN* by the input words of all 16 speakers, even those of opposite sex (i.e. very different formant frequencies and temporal waveform fine-structure), and follows similar trends of the pattern generated by training on all 16 speakers. Although generalization is not perfect; with regard to energy output, *DSNNs* formulated and trained in the manner explained in section 24.4 demonstrate generalization from speaker-dependent training to speaker-independent operation that is stronger than expected. Fourth, our trained *DSNNs* have bounded input and output. Training toward classification based on output energy difference alone tended toward driving the *DSNN* outputs to saturation or zero. However, *DSNNs* trained to the goals of equations 24.14 to 24.17 tend to have total output on a bounded range, with some words generating less, and some more, energy, but most responding with near average total output. In the following subsections we describe how the output energy is expected to be distributed as a function of time for each input word.

24.5.2 Dynamic Model Emergence Via Cost-Weighted Classification

Intelligible speech is coherent at the time scale of the syllable [18]. A weak implication of this is the variability of the (appropriately analyzed and described) set of sound waveforms elicited by all speakers if vocalizing a given word is low. This is the general assumption underlying all word recognition devices. We have a similar assumption to guide our output visualization, that underlying speech perception is a set of canonical word representations that can be expressed as the expectation of a distinct event sequence. The degree to which this second assumption is valid for our *DSNN* transformation is the underlying theme of this section. In this section we compare the expected temporal output of two *DSNNs*, one trained to recognize the word repeat as spoken by a single female speaker, the other trained to recognize the word repeat as spoken by all sixteen speakers.

The top four frames of figure 24.9 are the average output of $DSNN_{RP}$ trained on a single female speaker. The average responses for that speaker, $RP_{f/f}(\cdot)$, are plotted atop the average responses of this network to all sixteen speakers, $RP_{f/\forall}(\cdot)$. There are three response types: a negative going response, i.e. $RP(SP, ST, GO,$ and $HP)$, a primarily negative going response but also having a sharp positive going prelude, i.e. $RP(NO, ER,$ and $YS)$, and an oscillating response, i.e. $RP(RB, EN,$ and $RP)$. As expected, the single speaker training responses are 'noisier' than the sixteen speaker validation responses. Observation indicates that $RP_{f/f}(\cdot)$ versus $RP_{f/\forall}(\cdot)$ are similar: $RP(SP,$ and $ST)$ are near identical; $RP(RP,$ and $NO)$ are similar; and $RP(HP, GO,$ and $ER)$ each have a slightly exaggerated response. The response dissimilarities are: the $RP_{f/f}(YS)$ is approximately $80msec$ time delayed and has a half

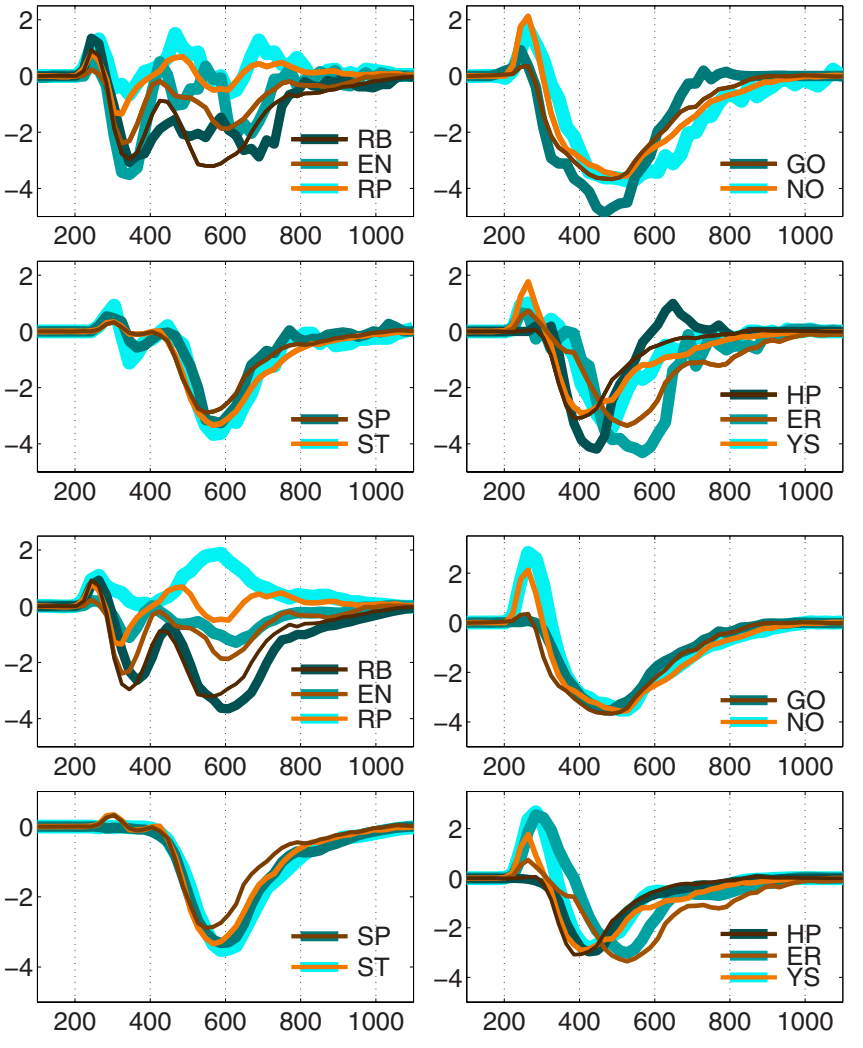


Fig. 24.9 Average temporal output for a *DSNN* trained on a single female speaker (Top half: blue thick lines), average temporal output of a *DSNN* trained on all speakers (Bottom half: blue thick lines), versus average temporal output for the single speaker trained *DSNN* responding to all sixteen speakers (thin orange to brown lines). Time is in msec

magnitude positive prelude as compared to $RP_{f/\forall}(YS)$; $RP_{f/f}(EN)$ is very similar to $RP_{f/\forall}(EN)$, with the exception of a positive going 'feature' that peaks just prior to 600*msec* in $RP_f(EN)$ that is averaged out of $RP_{f/\forall}(EN)$; this same feature is found in the $RP_{f/f}(RB)$ response, but not in $RP_{f/\forall}(RB)$. As expected, words with pronunciation requiring a mid-word full stop produce an oscillating response ('rubout' and

'repeat'); 'enter' is spoken with at least a partial stop, it also produces an oscillating response. We expected, but could not measure, the compilation of error due to syllable sequences e.g. different speakers vocalize the syllable sequence at different rates, so that the average of all speaker output at any given time after the start of vocalization includes the response to multiple syllables.

The bottom four frames of figure 24.9 compare $RP_{f/\forall}(\cdot)$ to the average responses of a *DSNN* trained on all sixteen speakers ($RP_{\forall/\forall}(\cdot)$). We note that the single speaker training required two days of optimization time (individual 4679; score = 344.65 training; score = 215.3 validation), whereas the sixteen speaker training required seven parallel GAs running for approximately two weeks on seven identical computers (individual 1048 of one of the parallel GAs; score = 266.51) . There are three dissimilarities in these responses: the positive prelude in $RP_{f/\forall}(ER)$ is practically missing, whereas it is quite prominent in $RP_{\forall/\forall}(ER)$, and the positive prelude for $RP_{f/\forall}(YS)$ is truncated by nearly 30% relative to $RP_{\forall/\forall}(YS)$; and possibly significant, $RP_{f/\forall}(RP)$ is an entirely different waveform than $RP_{\forall/\forall}(RP)$.

The isolated results displayed in figure 24.9 typify a very important general theme we see in all trained *DSNNs*. There are three cluster-types of trained I/O: first, multiple (distinct temporal, spectral, and phoneme description) input speech classes that elicit similar average temporal output $(n \mapsto 1; e.g. RP(ST) = RP(SP))$; second , input speech classes that elicit a distinct average temporal output response $(1 \mapsto 1; e.g. RP_{\forall/\forall}(RP))$; but most often, multiple input speech classes that elicit a gradient of average temporal output response $([a \ldots n] \mapsto [A \ldots N])$. The most obvious example of the last cluster type is displayed in the sequence of $RP_{f/forall}(RB) \leftrightarrow RP_{f/\forall}(EN) \leftrightarrow RP_{f/\forall}(RP)$. In such a case, the waveform describing the expected, canonical response, is very similar, however, the magnitude (and/or very low frequency modulation) of the response differs. Less obvious from the responses graphed here (but seen in other *DSNNs*) are the sequences in critical point movement, breadth of the negative going lobes, and the height of the sharp positive prelude.

Taken altogether, what we find is that any given optimization results in a small subset of words being distinguishable from the others based on output energy (previous section) and temporal response – single *DSNNs* as defined in figure 24.2 and constrained by dynamic operation as described in section 24.3.2 are weak classifiers with regard to capacity or total number of distinguishable input classes. However, most words elicit a response that tends towards a canonical output; our experience suggests that these *DSNNs* are very good ordering operators on the input set. Their output can support system level (multiple *DSNNs*) probabilistic classification schemes. In particular, in graphing the output responses, we found that there was an obvious order of the responses in terms of expected magnitude, critical point, and breadth of waveform – and most importantly, that the expected sequences were word dependent, but were consistent between single speaker and multiple speaker trained *DSNNs*. This suggests that our *DSNNs* are capable of quickly forming the basis for a probabilistic speech recognizer, without making *apriori* assumptions regarding the input or desirable output waveforms.

24.5.3 System Output Visualization

In the preceding sections, we emphasized *DSNNs* trained to a particular TW that demonstrated typical behavior. However, each TW indicates a unique separatrix between TW and non-TW subclasses. In this section we display the ordering of the input subsets determined by a set of ten *DSNNs* (**DSNN**$_{\forall/\forall}$), each one trained to a different TW of the ten word input set. Where possible, we also point out observed atypical behavior. To summarize this data, we visualize the functions using pseudo-color plots, rather than x-y plots. In figure 24.10 ten such plots are drawn, each one being the canonical response to the word labeled.

An example of each TW defining a distinct separatrix is found in the responses to the words, 'rubout' and 'stop'. In figure 24.10, $RB_{\forall/\forall}(RB)$ (5^{th} row, RB, an orange rectangle) is approximately equal to $RB_{\forall/\forall}(SP)$ (5^{th} row, SP); the timing of the response indicates that $DSNN_{RB}$ has converged onto a strong response to the vowels preceding the bilabial consonant. $SP_{\forall/\forall}(SP)$ (8^{th} row, SP) and $SP_{\forall/\forall}(RB)$ (8^{th} row, RB) are however, quite distinct (almost additive inverses of each other).

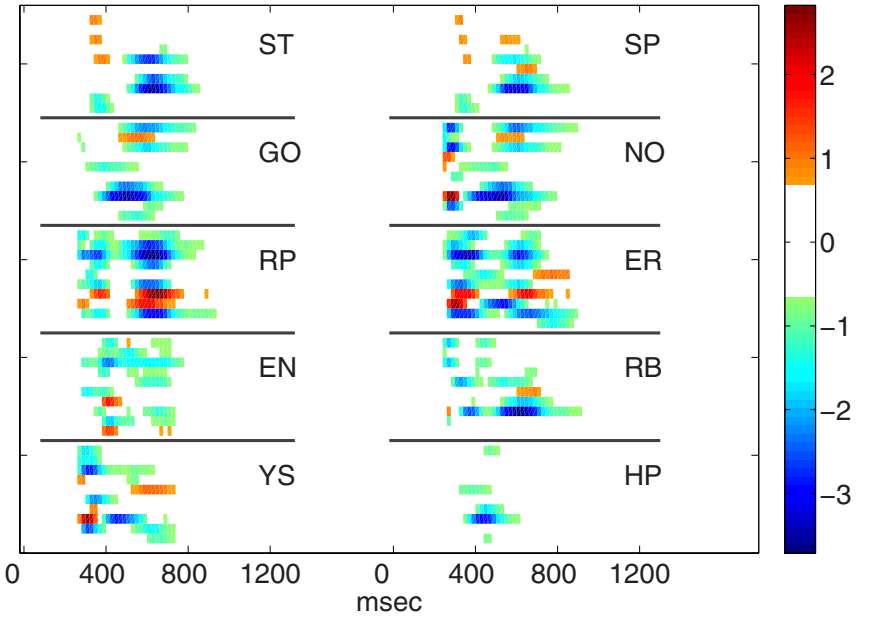


Fig. 24.10 The expected output of all DSNNs is shown here for each input word. The input word label is given to the right of each set of graphs. Graph order is the same for each set, from bottom to top: $EN, HP, RP, ER, RB, YS, NO, SP, GO, ST$. For example, the lower left set of color graphs are $EN(YS), HP(YS), RP(YS), ER(YS), ...ST(YS)$. The functions graphed here are for *DSNNs* trained on all available word samples. The third row of each plot corresponds to the x-y graph of the same label in the lower half of figure 24.9; as in that figure, event averages are for $\Delta t = 20msec$, averages near zero are not plotted

Not only does each TW indicate a unique separatrix, it also indicates a unique pattern of non-target subclass separation. An example is the response of $\mathbf{DSNN}_{\forall}(ST)$ and $\mathbf{DSNN}_{\forall}(SP)$ to the words 'start' and 'stop'. $DSNN_{EN}$, $DSNN_{HP}$, $DSNN_{RP}$, $DSNN_{ER}$, $DSNN_{GO}$, and $DSNN_{ST}$ have indistinguishable responses to the words 'start' and 'stop' (rows 1-3, 9, 10, of ST and SP); and $DSNN_{ER}$, $DSNN_{YS}$ and $DSNN_{NO}$, have similar responses (rows 4,6, and 7, of ST and SP, relative color magnitude difference between $600-700msec$). The distinguishing characteristics of these responses is found in two *DSNNs*: $DSNN_{RB}$, which has no significant response to the word 'start', but does respond to 'stop' (row 5), and $DSNN_{SP}$, which has a bimodal response to 'stop', but only a unimodal response to 'start'.

Likewise, $\mathbf{DSNN}_{\forall}(GO)$ and $\mathbf{DSNN}_{\forall}(NO)$ are remarkably similar – however, eight of the 10 trained *DSNNs* have response to NO that begins with a prelude to the main system response, five of these preludes are short duration negative bursts. Although three of these expected preludes are of modest magnitude, the sum total of the prelude response is a significant identifying feature of $\mathbf{DSNN}_{\forall}(NO)$.

$\mathbf{DSNN}_{\forall}(RP)$ and $\mathbf{DSNN}_{\forall}(ER)$ are somewhat similar, the onset of response ($< 400msec$) indicates an expected magnitude difference of approximately 8 events greater for $\mathbf{DSNN}_{\forall}(ER)$. After this initial phase, there is a cessation of activity on seven of the ten *DSNNs* of $\mathbf{DSNN}_{\forall}(RP)$ while at the same time $\mathbf{DSNN}_{\forall}(ER)$ is sparsely active with a strong negative response from $RP(ER)$. This is followed by a $100msec$ phase where, again, the two responses are very similar except in magnitude, during this time period $\mathbf{DSNN}_{\forall}(RP)$ is greater. These responses finish with a $100msec$ period during which time $EN(ER)$ and $YS(ER)$ show strong activity, whereas there is little to no activity from $\mathbf{DSNN}_{\forall}(RP)$.

Similar observation reveals the significant differences between any given two spatiotemporal response patterns. These diffferences are indicative that the overall expected output is not necessarily as important as where and when is the output for distinguishing the response of one word vs. another – an expected conclusion when using dynamic neural networks to analyze a signal. What was unexpected was the minimal amount of information required to train these differences – namely, no information regarding composition of the non-TW subset.

24.6 Discussion

Pattern classification tasks often begin with an analysis of the input signal. In particular, the first concern is to determine useful signal components or parameters of the signal (nonlinear filtering [20]) that can be measured or catalogued in such a way that the signals are more easily classified after measurement than before. Research in speech perception (human pattern classification of speech) has identified over 50 such measures [18]. A standard approach toward building VCIs is to analyze input speech signals in terms of one or more known measures with the aim of mapping the input sound as a function of time into a sequence of meaningful speech. In contrast to this approach, one may assume only that the input signal contains classifiable components (or that the input is a set of classifiable signals), and then after

applying a transform to the input, perhaps repeatedly, hypothesizing the number of input classes and to which elements each input belongs based on how well the output can be clustered and how distinguishable the clusters are from each other. In this chapter we took this second approach to explore the ability of a small number of biological synaptic models to duplicate a human hearing task: learning a word spoken by a single speaker. We then demonstrated that this oversimplified model of hearing generalizes to 15 other speakers. We chose to qualitatively present our results via visualization primarily because the aim of this work is the development of an appropriate input filter for a biologically based large scale waveform analysis system: we showed here that it is possible to quickly produce a transformation that orders the expected dynamic output as a function of the input words for subsequent classification while also 'filtering out' irrelevant input. Using an array of such produced *DSNNs* we demonstrated how the property of optimal sequence excitation of individual synapses is expressed by our networks as unique expectations in time and space of the passed energy. Finally, we showed that these canonical dynamic responses per word are distinguishable, despite that the response ordering resulting from individual *DSNN* optimization was independently determined. A genetic algorithm was central to this work. Initially, optimization was by random search as mutation rate was set very high. After finding a combination of input and model parameter values that carried out the classification task sufficiently well, search became more directed: mutation rate decreased, the number of local parameter spaces explored decreased, and the choice of input converged to a stationary process.

Canonical response differences are not easily measured by the usual measures, which operate on the entire output. Differentiating 'start' from 'stop' is an excellent example. These input are discernible from all others, but to distinguish them from each other requires analysis be focused onto a small section of the spatiotemporal output. It is also difficult to know from a neuro-philosophical perspective *the* appropriate statistical inference to apply to the output to support classification, as dendritic physiology tells us that non-linear signal and signal component operations (summation, deletion, relative delay, thresholding, etc.) are acceptable at the analysis stage of processing. Finally, human perception of isolated words via cochlear implant and simulations of, is the benchmark for the work begun here; however, human perception is based on contextual processing that will include many more layers of processing each of which have recursive input to various of the preceding layers.

The *DSNNs* reported here cannot be used as the sole computational component of a classification system. Canonical response differences are not easily measured by the usual measures, which operate on the entire output. Differentiating 'start' from 'stop' is an excellent example. These input are discernible from all others, but to distinguish them from each other requires analysis be focused onto a small section of the spatiotemporal output. It is also difficult to know from a neuro-philosophical perspective *the* appropriate statistical inference to apply to the output to support classification, as dendritic physiology tells us that non-linear signal and signal component operations (summation, deletion, relative delay, thresholding, etc.) are

acceptable at the analysis stage of processing. Finally, human perception of isolated words via cochlear implant and simulations of, is the benchmark for the work begun here; however, human perception is based on contextual processing that will include many more layers of processing each of which have recursive input to various of the preceding layers. However, our *DSNNs* are a suitable candidate model of the input transform (or filter) required to support development of a large biologically based contextual processing system. This is because they conform to several requirements. They are quickly and independently optimized so that adding new classes will not require re-optimizing the entire system from scratch. A subset of the input classes are "filtered out" of consideration by any given DSNN with the remaining input classes having an ordered output; so that we can expect novel input would either be filtered out, or would generate new expected patterns that fall within the original ordering. There is much interest in applying biological solutions to real-world problems, and in this sense, our *DSNNs* are an excellent example of where to begin neuro-engineering such solutions for waveform classification.

24.6.1 Sound Processing Via the Dorsal Cochlear Nucleus

There are myriad differences in detail between cochlear nucleus physiology and our *DSNNs*. However, from an information processing perspective, our *DSNNs* do have functionality in common with the DCN transform – withstanding that we filter the sound in a single layer of synapses, whereas the DCN contains apical and basal synapses (onto the pyramidal cells) each set of synapses having complete neural circuits, including interneurons, preceding them. First, there is a complex (sensitive to multiple spectral bands) inhibition of the spectrally narrow input. In the *DSNN* this comes via the feedforward inhibitory interneuron (which is sensitive to all seven of the inputs) onto the presynapses (each sensitive to a single spectral decomposition component); in the DCN this comes via multipolar cells of the Ventral CN. These would be the onset-C neurons that have very good envelope sensitivity, as measured by modulation depth. Second, there is a sensitivity to the spectral envelope of the sound, versus the fine temporal structure of the sound. For the *DSNNs* this sensitivity is inherent in the pulse encoding of the stimulus wavelet decomposition, in the DCN this comes via outer spiral fiber excitation of granule cells followed by granule cell parallel fiber excitation of the apical pyramidal dendrites. Third, there is a complex (sensitive to multiple spectral bands) excitation of the output. In the *DSNN* this occurs via the modulation of postsynaptic potentials by multiple synapses through the long-term kinetics of the NMDA model, in the DCN this comes from the granule cell axonal connection to the pyramidal cells, which is perpendicular to the frequency axis of the DCN. We further note that granule cell synapses contain functional NMDA receptors [2, 32]. The preceding points do not argue for our *DSNNs* being a model of DCN function – rather they point toward certain underlying neural processing that may be important toward understanding the neural complexity underlying waveform classification and that may be engineered using realistic synaptic mechanisms.

Acknowledgements. This work was supported by grants from the USA Office of Naval Research, and the USA Department of Defense/DARPA. Thanks to Shivani Pandya, Dr. Sageev George, and Dr. Hassan Namarvar, for computer programming of the encoded input files and much of the software framework within which our *DSNNs* run.

References

1. Arnott, R.H., Wallace, M.N., Shackleton, T.M., Palmer, A.R.: Onset neurones in the anteroventral cochlear nucleus project to the dorsal cochlear nucleus. JARO 5, 153–170 (2004)
2. Balakrishnan, V., Trussel, L.: Synaptic inputs to granule cells of the dorsal cochlear nucleus. Journal of Neurophysiology 99, 208–219 (2008)
3. Ballard, D.H.: An Introduction To Natural Computation. MIT Press, Cambridge (1999)
4. Berger, T.W., Eriksson, J.L., Ciarolla, D.A., Sclabassi, R.J.: Nonlinear systems analysis of the hippocampal perforant path-dentate projection. II. Effects of random train stimulation. Journal of Neurophysiology 60, 1077–1094 (1988)
5. Bregman, A.: Auditory scene analysis: the perceptual organization of sound. MIT Press, Cambridge (1990)
6. Cerf, R.: Asymptotic convergence of genetic algorithms. Adv. Appl. Prob. 30, 521–550 (1998)
7. Davis, M.H., Johnsrude, I.S.: Hierarchical processing in spoken language comprehension. Journal of Neuroscience 23(8), 3423–3431 (2003)
8. Davis, T.E., Principe, J.C.: A simulated annealing like convergence theory for the simple genetic algorithm. In: Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 174–181. Morgan Kaufman, San Mateo (1991)
9. de Boer, E.: Correlation studies applied to the frequency resolution of the cochlea. Journal Auditory Research 7, 209–217 (1967)
10. Démonet, J.-F., Thierry, G., Cardebat, D.: Renewal of the Neurophysiology of Language: functional neuroimaging. Physiol. Rev. 85, 49–95 (2005)
11. Dorman, M., Loizou, P., Rainey, D.: Speech intelligibility as a function of the number of channels of stimulation for signal processors using sine-wave and noise-band outputs. J. Acoust. Soc. Am. 102, 2403–2411 (1997)
12. Dudley, H.: Remaking speech. J. Acoust. Soc. Am. 11, 169–177 (1939)
13. Dibazar, A., Song, D., Yamada, W.M., Berger, T.W.: Speech recognition based on fundamental functional principles of the brain. In: Proceedings of the IEEE International Joint Conference on Neural Networks, vol. 2004(4), pp. 3071–3075 (2004)
14. Fay, R.R., Popper, A.N.: Evolution of hearing in vertebrates: the inner ears and processing. Hearing Research 149, 1–10 (2000)
15. George, S.T.: The Use of Dynamic Synapse Neural Networks for Speech Processing Tasks. PhD Thesis. Department of Biomedical Engineering, University of Southern California, Los Angeles, CA 90089 (2007)
16. Di Girolamo, S., Napolitano, B., Alessandrini, M., Bruno, E.: Experimental and clinical aspects of the efferent auditory system. Acta Neurochir. Suppl. 97(2), 419–424 (2007)
17. Goldberg, D.E.: The Design of Innovation: Lessons from and for competent genetic algorithms. Kluwer Academic Publishers, Boston (2002)
18. Greenberg, S.: The origins of speech intelligibility in the real world. In: Proc. of the ESCA Workshop on Robust Speech Recognition for Unknown Communication Channels, pp. 23–32 (1997)

19. Griffiths, T.D., Bates, D., Rees, A., Witton, C., Gholkar, A., Gren, G.G.R.: Sound movement detection deficit due to a brainstem lesion. J. Neurol. Neurosurg. Psych. 62, 522–526 (1997)
20. Haykin, S.: Modern Filters. Macmillan Publishing Co., New York (1989)
21. Koch, C., Segev, I. (eds.): Methods In Neuronal Modeling: from ions to networks, 2nd edn. MIT Press, Cambridge (1998)
22. Kohavi, R., Provost, F. (eds.): Glossary to The Special Issue on Applications of Machine Learning and Knowledge Discovery Process. Machine Learning, vol. 30, pp. 271–274 (1998)
23. Lewis, E.R., Henry, K.R., Yamada, W.M.: Tuning and timing of excitation and inhibition in primary auditory nerve fibers. Hearing Research 171, 13–31 (2002)
24. Lewis, E.R., Henry, K.R., Yamada, W.M.: Tuning and timing in the gerbil ear: Wiener-kernel analysis. Hearing Research 174, 206–221 (2002)
25. Liaw, J.-S., Berger, T.W.: Dynamic Synapse: A New Concept of Neural Representation and Computation. Hippocampus 6, 591–600 (1996)
26. Liaw, J.-S., Berger, T.W.: Robust Speech Recognition With Dynamic Synapses. In: IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on Neural Networks, Neural Networks Proceedings, 1998, vol. 3, pp. 2175–2179 (1998), doi:10.1109/IJCNN.1998.687197
27. Liaw, J.-S., Berger, T.W.: Dynamic Synapse: Harnessing the computing power of synaptic dynamics. Neurocomputing 26-27, 199–206 (1999)
28. Liberman, C., Guinan, J.: Feedback control of the auditory periphery: antimasking effects of middle ear muscles vs. olivocochlear efferents. J. Commun. Disord. 31, 471–483 (1998)
29. Loiselle, S., Rouat, J., Pressnitzer, D., Thorpe, S.: Exploration of rank order coding with spiking neural networks for speech recognition. In: Proceedings of International Joint Conference on Neural Networks 2005, vol. 4, pp. 2076–2080 (2005) ISBN: 0-7803-9048-2/05
30. Loizou, P.C.: Mimicking the human ear. IEEE Signal Processing Magazine, 101–130 (September 1998)
31. On the number of channels needed to understand speech. J. Acoust. Soc. Am. 106(4), 2097–2103 (1999)
32. N-Methyl-D-Aspartate receptors at Parallel Fiber Synapses in the Dorsal Cochlear Nucleus. Journal of Neurophysiology 76(3), 1639–1656
33. Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, Cambridge (1996)
34. Gacek, R.R., Rasmussen, G.L.: Fiber analysis of the statoacoustic nerve of guinea pig, can and monkey. Anat. Rec. 139, 455–463 (1961)
35. Namarvar, H.H., Liaw, J.-S., Berger, T.W.: A New Dynamic Synapse Neural Network for Speech Recognition. In: Proceedings International Joint Conference on Neural Networks 2001, vol. 4, pp. 2985–2990 (2001), doi:10.1109/IJCNN.2001.938853
36. Natschläger, T., Maass, W.: Computing the optimally fitted spike train for a synapse. Neural Computation 13, 2477–2494 (2001)
37. Percival, D.B., Walden, A.T.: Wavelet Methods for Time Series Analysis. Cambridge University Press, Cambridge (2000)
38. Recio-Spinoso, A., Temchin, A.N., van Dijk, P., Fan, Y.H., Ruggero, M.A.: Wiener-Kernel Analysis of Responses to Noise of Chinchilla Auditory-Nerve Fibers. Journal of Neurophysiology 93, 3615–3634 (2005)
39. Rigal, L., Truffet, L.: A new genetic algorithm specifically based on mutation and selection. Adv. Appl. Prob. 39, 141–161 (2007)

40. Schetzen, M.: The Volterra and Wiener Theories of Nonlinear Systems. John Wiley and Sons, New York (1980)
41. Shannon, R., Zeng, F.-G., Kamath, V., Wygonski, J., Ekelid, M.: Speech recognition with primarily temporal cues. Science 270, 303–304 (1995)
42. Shepherd, G.M. (ed.): The Synaptic Organization of the Brain. Oxford University Press, Oxford (1998)
43. Shepherd, G.M., Koch, C.: Introduction to synaptic circuits. In: Shepherd, G.M. (ed.) The Synaptic Organization of the Brain. Oxford University Press, Inc., Oxford (1998)
44. Shouval, H.Z., Bear, M.F., Cooper, L.N.: A unified model of NMDA Receptor-dependent bidirectional synaptic plasticity. PNAS 99(16), 10831–10836 (2002)
45. Stevens, C.F.: Neurotransmitter release at central synapses. Neuron 40, 381–388 (2003)
46. Sumner, C.J., Lopez-Poveda, E.A., O'Mard, L.P., Meddis, R.: Adaptation in a revised inner-hair cell model. J. Acoust. Soc. America 113(2), 893–901 (2003)
47. Thiels, E., Barrionuevo, G., Berger, T.W.: Induction of long-term depression in hippocampus in vivo requires postsynaptic inhibition. Journal of Neurophysiology 72, 3009–3016 (1994)
48. Uysal, I., Sathyendra, H., Harris, J.: A biologically plausible system approach for noise robust vowel recognition. In: 49th IEEE International Midwest Symposium on Circuits and Systems, MWSCAS 2006, vol. 1, pp. 245–249 (2006), doi:10.1109/MWSCAS.2006.382043
49. Watkins, J.C., Jane, D.E.: The glutamate story. British Journal of Pharmacology 147, S100–S108 (2006)
50. Wu, L.-G., Saggau, P.: Presynaptic inhibition of elicited neurotransmitter release. Trends In Neuroscience 20(5), 204–212 (1997)
51. Xie, X., Berger, T.W., Barrionuevo, G.: Isolated NMDA receptor-mediated synaptic responses express both LTP and LTD. Journal of Neurophysiology 67, 1009–1013 (1992)
52. Xie, X., Barrionuevo, G., Berger, T.W.: Differential expression of short-term potentiation by AMPA and NMDA receptors. Learning and Memory 3, 115–123 (1996)
53. Yamada, W.Y.: Second-Order Wiener Kernel Analysis of Auditory Afferent Axons of the North American Bullfrog and Mongolian Gerbil Responding To Noise. PhD Thesis, University of California at Berkeley. Committee in charge: Edwin R. Lewis, Kenneth Henry, Erv Hafter, and Geoffrey Owen (1997)
54. Yamada, W.M., Lewis, E.R.: Predicting the temporal response of non-phase-locked bullfrog auditory units to complex acoustic waveforms. Hearing Research 130(1-2), 155–170 (1999)
55. Yeckel, M.F., Berger, T.W.: Feedforward excitation of the hippocampus by entorhinal afferents: Redefinition of the role of the trisynaptic pathway. Proceedings of the National Academy of Sciences 87, 5832–5836 (1990)
56. Young, E.: Cochlear Nucleus. In: Shepherd, G. (ed.) The Synaptic Organization of the Brain, 4th edn. Oxford University Press, Oxford (1998)
57. Zatorre, R.J., Gandour, J.T.: Neural specializations for speech and pitch: moving beyond the dichotomies. Phil. Trans. Royal Soc. B 363, 1087–1104 (2008)
58. Zucker, R.S., Regehr, W.G.: Short-term synaptic plasticity. Annual Reviews in Physiology 64, 355–405 (2002)

Chapter 25
A Distributed Evolutionary Approach to Subtraction Radiography

Gabriel Mañana Guichón and Eduardo Romero Castro

Abstract. Automatic image registration is a fundamental task in medical image processing, and significant advances have occurred in the last decade. However, one major problem with advanced registration techniques is their high computational cost. Due to this restraint, these methods have found limited application to clinical situations where real time or near real time execution is required, e.g., intraoperative imaging, or high volumes of data need to be processed periodically. High performance in image registration can be achieved by reduction in data and search spaces. However, to obtain a significant increase in performance, these approaches must be complemented with parallel processing. Parallel processing is associated with expensive supercomputers and computer clusters that are unaffordable for most public medical institutions. This chapter will describe how to take advantage of an existing computational infrastructure and achieve high performance image registration in a practical and affordable way. More specifically, it will outline the implementation of a fast and robust Internet subtraction service, using a distributed evolutionary algorithm and a service-oriented architecture.

25.1 Introduction

In image processing, the interest often lies not only in analyzing one image but also in comparing or combining the information present in different images. In this context, image registration can be defined as the process of aligning images so that *corresponding* features can be related. The term image registration is also used to refer to the alignment of images with a computer model or the alignment of features in an image with locations in physical space. For this reason image registration is one of the fundamental tasks within image processing: by determining the transformation required to align two images, registration enables specialists to make quantitative comparisons.

Gabriel Mañana Guichón · Eduardo Romero Castro
National University, Carrera 45 N 26-85, Bogotá, Colombia
e-mail: {gjmananag,edromero}@unal.edu.co

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 671–700.
springerlink.com

From an operational point of view, image registration is an optimization problem and its goal is to produce, as output, an *optimal geometrical transformation* that aligns corresponding points of the two given views. Image registration has applications in many fields, including remote sensing, astro- and geophysics, computer vision, and medical imaging. The field of application to be addressed in this chapter is medical imaging, and in this field, this transformation is generally used as input to another system that can be, for instance, a fusion system or a subtraction system. For a complete overview of the different image acquisition systems and the relevance of registration in medical image interpretation and analysis, you may refer to Hajnal et al. [29], and references therein.

In many clinical scenarios, images of the same or different modalities may be acquired and it is the responsibility of the diagnostician to combine or fuse the images information to draw useful clinical conclusions. Without an imaging system this generally requires mental compensation for changes in patient position, the sensors used, or even the chemicals involved. An image registration system aligns the images and so establishes correspondence between features present in different images, allowing the monitoring of subtle changes in size or intensity over time or across a population. It also allows establishing correspondence between images and physical space in image guided interventions. In many applications a rigid transformation, i.e., translations and rotations only, is enough to describe the spatial relationship between two images. However, there are many other applications where non-rigid transformations are required to describe this spatial relationship adequately.

In terms of the algorithms used, the current tendency is to use automatic algorithms (i.e., no user interaction) [8], which requires the application of advanced image registration techniques, all characterized by their high computational cost. Due to this restraint, these methods have found limited application in clinical situations where real time or near real time execution is required, e.g., intra-operative imaging or image guided surgery. High performance in image registration can be achieved by reduction in data space, as well as reduction in solution search space. These techniques can decrease significantly the registration time without compromising registration accuracy. Nonetheless, to obtain a significant increase in performance, these approaches must be complemented with parallel processing. The problem is that parallel processing has always been associated with extremely expensive supercomputers, unaffordable for most medical institutions in developing countries. This chapter will describe our experience in achieving high performance in an affordable way, i.e., taking advantage of an existing computational infrastructure. More specifically, it will outline how this can be done by using open source software tools that are readily available. This will be illustrated by the use of a real case study: an online subtraction radiography service that employs distributed evolutionary algorithms for automatic registration.

The chapter is organized as follows. Section 25.2, *Background*, briefly describes the main aspects behind medical image registration and presents a general overview of available options to attain high-performance computing for scientific research. Next, section 25.3, *A Grid Computing Framework for Medical Imaging*, presents our experience in building a scalable computing framework for medical imaging,

by using a service-oriented architecture and open source software tools. To show the use of the framework in a real working situation, section 25.4, *Case Study: Automatic Subtraction Radiography using Distributed Evolutionary Algorithms*, will review the algorithms behind the implementation of the online system. Finally, section 25.5, *Discussion and Conclusions* draws some pertinent conclusions and summarizes our experience adopting a service-oriented approach and an open source developing model.

25.2 Background

As introduced in the previous section, the task of image registration is to find an optimal geometric transformation between corresponding image data. The image registration problem can be stated in just a few words: given a *reference* and a *template* image, find an appropriate geometric transformation such that the transformed template becomes *similar* to the reference. However, though the problem is easy to express, it is hard to solve. In practice, the concrete types of the geometric transformation, as well as the notions of *optimal* and *corresponding* depend on the specific application. In this section we summarize the main aspects involved in the registration process and review recent trends in high-performance computing (HPC).

25.2.1 The Image Registration Problem

Any image registration technique can be described by three main components [6]:

1. a *geometrical transformation* which relates reference and template images,
2. a *similarity measure* which measures similarity between reference and transformed image,
3. an *optimization scheme* which determines the optimal transformation as a function of the similarity measure.

Geometrical transformation refers to the mathematical forms of the geometrical mapping used in the registration process and can be classified by complexity into rigid transformations, where all distances are preserved, and deformable or non-rigid transformations where images are stretched or deformed. While the first is ideal for most fusion applications, and accounts for differences such as patient positioning, non-rigid transformations are used to take into account more complex motions, such as breathing or heartbeat.

The *similarity measure* is the driving force behind the registration process, and it aims to maximize the similarity between both images. From a probabilistic point of view, it can be viewed as a likelihood term that expresses the probability of a match between the reference and transformed image [15]. Like many other problems in computer vision and image analysis, registration can be formulated as an optimization problem whose goal is to minimize an associated cost function [6]:

$$C = -C_{similarity} + C_{transformation}, \tag{25.1}$$

where the first term characterizes the similarity between the images and the second term characterizes the cost associated with particular deformations. From a probabilistic point of view, the cost function in eq. (25.1) can be can be explained in a Bayesian context. In this framework, the similarity measure can be viewed as a likelihood term which expresses the probability of a match between the two images, and the second term can be interpreted as a prior which represents *a priori* knowledge about the expected deformation. This term only plays a role in non-rigid registration and in the case of rigid registration is usually ignored.

Several approaches can be used to optimize this function. They go from the use of standard numerical methods to the use of evolutionary methods, including some hybrid approaches. No matter what method is used, this always implies an iterative process whose computational cost is so high that prevents most applications from performing appropriately in real time situations. One possible way to solve this issue is to devise faster algorithms. Another way is to exploit the intrinsic parallelism that most methods convey.

Medical image registration spans numerous applications and there is a large score of different techniques reported in the literature. What follows is an attempt to classify the different techniques and categorize them based upon some criteria, for a complete analysis, please refer to, e.g., [2]. Maintz and Viergever [25] originally proposed a nine-dimensional scheme that can be condensed into the following eight criteria [22]: *image dimensionality*, *registration basis*, *geometrical transformation*, *degree of interaction*, *optimization procedure*, *image acquisition modalities*, *subject*, and *object*.

Image dimensionality refers to the number of geometrical dimensions of the image spaces involved, which in medical applications are typically two and three-dimensional, but may include time as a fourth dimension. For spatial registration, there are the 2D/2D, 3D/3D and the more complex 2D/3D registration (e.g., CT/X-ray).

The *registration basis* is the aspect of the two images used to perform the registration. In this category, registration can be classified into *extrinsic* and *intrinsic* methods. Registration methods that are based upon the attachment of markers are termed extrinsic methods, and in contrast, those which rely on anatomic features only are termed intrinsic. When there are no known correspondences as input, intensity patterns in the two views are used for alignment. A basis known as *intensity*- or *voxel*-based, has become in recent years the most widely used registration basis in medical imaging. Here there are two distinct approaches: the first reduces the image gray value content to a representative set of scalars and orientations (e.g. principal axes and moments based methods), the second uses the full image pixel content throughout the registration process. In general, intensity-based methods are more complex, yet more flexible.

The category *geometrical transformation* refers to the mathematical forms of the geometrical mapping used to align points in one space with those in the other. These include *rigid transformations*, which preserve all distances, i.e., transformations that preserve the straightness of lines - and hence planarity of surfaces - and all angles

between straight lines. Images are rotated and translated in two or three dimensions in the matching process, but not deformed in any way. This is ideal for most fusion applications, and accounts for differences such as patient positioning. Registration problems that are limited to rigid transformations are called *rigid registration* problems. In deformable or *non-rigid registration*, images are stretched to take into account complex motions, such as breathing, and any changes in the shape of the body or organs, which may occur following surgery, for example. Non-rigid transformations are important not only for applications to non-rigid anatomy, but also for inter-patient registration of rigid anatomy and intra-patient registration of rigid anatomy, in those cases where there are non-rigid distortions caused by the image acquisition procedure. These include *scaling transformations*, with a special case when the scaling is isotropic, known as *similarity transformations*; the more general *affine transformations* that preserve the straightness of lines and planarity of surfaces, as well as parallelism, but change the angles between lines; the even more general *projective transformations* that preserve the straightness of lines and planarity of surfaces, but no parallelism; *perspective transformations*, a subset of the projective transformations, required for images obtained by techniques such as X-ray, endoscopy or microscopy, and finally *curved transformations* which do not preserve the straightness of lines. Each type of transformation contains as special cases the ones described before it, e.g., *rigid transformations* are a special type of *non-rigid transformations*, and so on. Transformations that are applied to the whole image are called *global*, while transformations that are applied to subsections of the image are called *local*. Rigid, affine and projective transformations are generally global, and curved transformations are more or less local, depending upon the underlying physical model used.

Degree of interaction refers to the degree of intervention of a human operator in the registration algorithm. The fully automatic algorithm, which requires no user interaction and represents the ideal situation, is a central focus of the subtraction service presented in this chapter.

The *optimization procedure* is the method by which the function that measures the alignment of the images is maximized. Depending upon the mathematical approach to registration used, i.e., *parametric* or *non-parametric*, the optimization method will try to find an optimum of some function defined on the parameter space, or will try to come up with an appropriate measure, both for the similarity of the images as well as for the likelihood of a non-parametric transformation. The more common situation here is that in which a global extremum is sought among many local ones by means of iterative search. In parametric registration, popular techniques include traditional numerical methods like Powell's method [36], Downhill Simplex [24], gradient descent methods, as well as evolutionary methods like genetic algorithms [27], simulated annealing [39], and differential evolution [30].

Modalities refers to the means by which the images to be registered are acquired. Two-dimensional images are acquired, e.g., by X-ray projections captured on film or digitally, and three-dimensional images are typically acquired by tomographic

modalities such as computed tomography (CT), nuclear magnetic resonance (MRI), or positron emission tomography (PET). In medical applications the object in each view is some anatomical region of the body. In all cases we are concerned primarily with digital images stored as discrete arrays of intensity values. Registration methods used for like modalities are typically distinct from those used for differing modalities. They are called *mono-modal* and *multi-modal* registration, respectively.

Subject refers to patient involvement and there can be *intra-patient* registration involving images of the same patient, *inter-patient* registration involving images of different patients, and *atlas*. Atlas refers to registration between an image acquired from a single patient and an image constructed from an image database of many patients.

Finally, *object* refers to the particular region of anatomy to be registered, e.g., mandible.

The subtraction service presented in this chapter is an application of parametric registration of X-ray images, and uses an automatic and intensity-based registration method. Furthermore, a real encoding and distributed evolutionary algorithm is used to find the projective transformation that aligns the images.

25.2.2 High Performance Computing

Technology plays a critical role today to help academics to do research more effectively. Most research relies on high performance computing for data-intensive applications, information access, visualization and communications. The last decade has seen a considerable increase in the performance of computers and communication networks, mainly due to faster development of hardware and more elaborate software. However, there are still many problems associated with algorithms in the fields of science, engineering and business, which cannot be managed efficiently with the current generation of supercomputers. This inefficiency is reflected in several negative factors associated with the supercomputer: high cost[1], complex maintenance and administration, limited scalability, rapid obsolescence.

Another option is to link together an homogeneous set of high performance servers or personal computers [40] by means of a fast local area network. These are known as *computer clusters* and can provide a computing capacity similar to that provided by supercomputers, at a fraction of their cost.

A number of teams have conducted studies on the cooperative use of geographically distributed resources conceived as a single powerful virtual computer [11]. This alternative approach is known by several names, such as, meta-computing, global computing, and more recently by *grid computing*. Internet and grid-based systems, whether their purpose is computation, collaboration or information sharing, are all instances of systems based on the application of fundamental principles of distributed computing. Grid computing is a set of standards and technologies that academics, researchers, and scientists around the world are developing to help

[1] The IBM Roadrunner, number one in the TOP500 list as of the end of 2008: about 130 million dollars (`http://en.wikipedia.org/wiki/IBM_Roadrunner`).

organizations take collective advantage of improvements in microprocessor speeds, optical communications, raw storage capacity, and the Internet. By using the technique to disaggregate their computer platforms and distribute them as network resources, researchers can vastly increase their computing capacity. Linking geographically dispersed and heterogeneous computer systems can lead to important gains in computing power, speed, and productivity.

When the unGrid[2] research project started back in 2004, there were already several thousand personal workstations in the main university campus (Bogotá), interconnected by a back bone of high speed fiber optics (FDDI). Studies carried out on a random sample of more than two hundred of these computers, showed that an average of 90% of the CPU cycles were wasted. Our research group - like most research groups at the university - requires a significant computing capacity and based on the fact that no other computing facilities were available at that time, we started the development of a software framework that would allow us to benefit the idle CPU cycles, and use it to build a general purpose computing grid.

There are many design choices to be made along the way to building a computing grid for research purposes. The first one and most determining factor, is related with a software layer that lies between the user applications and the operating system, a technology known as middleware.

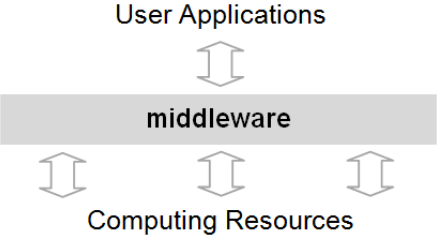


Fig. 25.1 The easiest way to integrate heterogeneous computing resources is not to recreate them as homogeneous elements, but to provide a layer that allows them to communicate despite their differences. This software layer is commonly known as *middleware*

This layer is responsible for hiding distribution and the heterogeneity of the various hardware components, operating systems and communication protocols. At its most basic level, middleware is nothing but a way of abstracting access to a resource through the use of an *Application Programming Interface* (API). Despite their benefits, distributed systems can be notoriously difficult to build. Perhaps the most obvious complexity is the variety of machine architectures and software platforms over which a distributed application must function. In the past, developing a distributed application entailed porting it to every platform it would run on, as well as managing the distribution of platform-specific code to each machine. Most of the computers in the University campus are used in academic related tasks and

[2] Please refer to `http://ungrid.unal.edu.co/`

use a variety of operating systems, a fact that clearly indicated the necessity of a *platform-independent* middleware. Additionally, the computing grid would be part of a bigger system[3] that uses a service-oriented architecture (SOA), so it should act as another service.

Another important aspect to be considered was related to the available computing infrastructure. Contrary to what happens with computers in a cluster, that are dedicated and under a single administration domain, the computers in the campus are shared and belong to many different domains. This meant that the computers would frequently enter and leave the grid at random, and therefore the middleware to use should allow us to build a loosely coupled system, in space (network addresses) and time (synchronization).

25.3 A Grid Computing Framework for Medical Imaging

Based upon the previous considerations, and a thorough analysis of the available middleware technologies at that moment, we decided to use the Java-based Jini[4] platform on top of which construct our computing grid. Jini, an open source technology that now is part of the Apache project River [3], is a service-oriented technology that provides a platform and protocol-agnostic infrastructure to build distributed systems. This includes services for the registration and discovery of other services, distributed events, transactional support like that one provided by relational database engines, and most important for us, the JavaSpaces service. This technology is a high-level coordination tool for gluing processes together into a distributed application by means of a minimal, yet powerful, programming interface. It is a departure from conventional distributed tools which rely on passing messages between processes (MPI), or invoking methods on remote objects (RPC), and produce inherently *tightly coupled* systems. The JavaSpaces technology uses a fundamentally different approach that views a distributed application as a collection of processes cooperating via the flow of objects into and out of one (or more) distributed shared-memory space. The space-based model of distributed computing has its roots in the Linda coordination language developed by Dr. Gelernter at Yale University [5].

A *space*, in this context, is a shared and network-accessible repository for objects that processes can use as persistent storage and exchange mechanism: instead of communicating directly, they coordinate by exchanging objects through spaces, as shown in Figure 25.2. Despite its minimal programming interface, JavaSpaces provides a unique set of features that allows for the construction of loosely coupled and transactionally secure distributed applications, and as we will show next, for the implementation of computing grids with automatic load balancing.

Spaces allow the exchange of objects, that is, self-contained entities that include data and related code. While in the space, objects are just passive data, but when read

[3] The system now includes services for data mining, machine learning, simulation and visualization, and image analysis. For detailed information please refer to `http://www.bioingenium.unal.edu.co/`

[4] Java, Jini and JavaSpaces are trademarks of Sun Microsystems Inc.

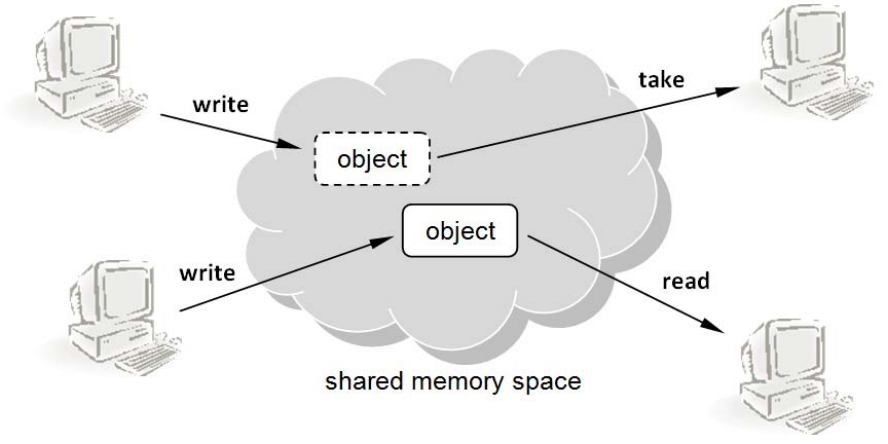


Fig. 25.2 Space-based coordination by means of a simple programming interface

or taken out from the space they transform themselves into standalone applications. This not only solves the problem of code distribution, but also gives us a powerful mechanism for building parallel computing servers. The application pattern use for this is known as the Replicated-Worker pattern [9], and involves a master process that divides a problem into smaller tasks and puts them into a space. The workers take and execute these tasks, and write the results back into the space. It is then the responsibility of the master to collect the task results and combine them into a meaningful overall solution.

It is worth pointing out a couple of important characteristics of this pattern. First, each worker process may execute many tasks, as soon as one task is computed a worker can take another task from the space and execute it. In this way, the replicated-worker pattern automatically balances the load: workers compute tasks in direct relation to their availability and capacity to do the work. Second, the type of applications that fit into the replicated-worker pattern scales naturally: more workers can be added and the computation speeds up, without rewriting the code. The Appendix section shows how the JavaSpaces API and the replicated-worker pattern can be used to build a *generic worker node*.

Another important issue that has to be addressed when implementing a distributed system is related to the following. So far, we have discussed several tools that Jini provides and that help obtain fault tolerance in the worker nodes. However, the set of Jini services run in a server computer, a situation known as a *single point of failure* (SPOF). This means that if this single server computer fails for some reason, the whole computing grid goes down. To avoid this situation, we have embedded our computing grid service, along with Jini, in a layered application (JEE). The service is then run in a cluster of six application servers using two open source frameworks: the application server JBoss [26] and the clustering tool Terracotta [41].

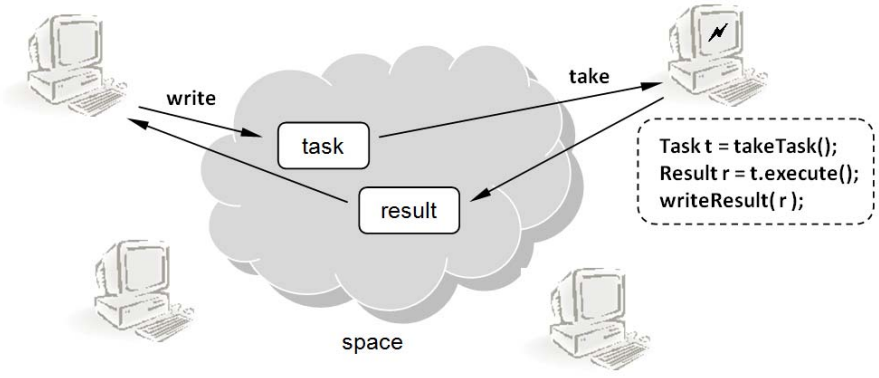


Fig. 25.3 Space-based replicated-worker design pattern

25.4 Case Study: Automatic Subtraction Radiography Using Distributed Evolutionary Algorithms

Digital subtraction radiography detects tissue mass changes by subtracting two digital radiographs. This method was shown to be very useful in early diagnosis of disease and follow-up examination [1]. When subtracting two radiographs taken over time, the image features which are coincident to both images can be removed and the small changes can be amplified to highlight their presence. For many years, digital subtraction radiography in dentistry has been used to qualitatively assess changes in radiographic density. Numerous authors have demonstrated the ability of this method to improve diagnostic performance for the detection of approximal dental caries, periapical pathology and periodontal disease, e.g. [14]. A large variety of odontological diseases result in destruction of mineralized tissues, which are relatively small in the initial progression of the disease. A reliable detection and follow-up examination necessarily requires a precise alignment of the two images for the tissue changes to be detectable.

Multiple works in the literarture address the problem of image registration by means of evolutionary algorithms. The 2D intensity-based method proposed by Gómez García *et al.* in [18], for instance, uses a $(\mu + \lambda)$ evolutionary strategy ($\mu = 250$, $\lambda = 50$) for optimization and a multiscale representation of the images to reduce the search space. For the same problem, Yuan *et al.* [43] propose a feature-based method that uses a (μ, λ) selection scheme ($\mu = 50$, $\lambda = 300$). Cordón *et al.* [17] extend the binary-coded CHC algorithm [33] to work with real-coded chromosomes and successfully apply it to 3D registration. Particularly, De Falco *et al.* [20] show the ability of Differential Evolution to perform well in satellite image registration and raise its possible use in medical image registration.

In this section we will evaluate a standard numerical technique, the Downhill Simplex method, and two evolutionary strategies: Genetic Algorithms and Differential Evolution.

25.4.1 Problem Statement

Different approaches have been proposed for correcting such geometrical distortions. It goes from manual correction to different devices used to ensure a consistent geometric projection which can be reliably reproduced over time. In daily medical practice, however, devices for adequate patient fixation are not available to clinicians, a drawback that has not allowed the application of this method to the series of routine examinations needed for progression estimation of lesions or treatments. In fact, since most clinicians do not pay attention to this issue, radiographic examinations generally produce strong geometrical distortions which makes it inappropriate to apply conventional correction approaches. Under these circumstances, standard numerical techniques for extrema searching like the Powell's [36] method or the Downhill Simplex method [24] usually yield irrelevant results.

In this section, an entirely automatic method is proposed for spatial radiographic alignment in those cases where a considerable amount of distortion is presented. The process starts by selecting one of the two images as the reference while the other is considered to be the template image. Afterwards, illumination differences are eliminated by means of an equalization algorithm explained below. Consecutive geometrical transformations are then performed on the template image, and the outcome is compared to the reference image using the correlation ratio as the similarity measure.

Conventional registration approaches have been successfully used in those situations where the patients head has been appropriately fixated, therefore producing images with little distortions. However, anatomical variations either from patient to patient or for the same patient in two different moments, have been a major inconvenience for radiographic subtraction to become an applicable method in routine evaluations. Our problem can be defined, therefore, as a multi-parametric search in a highly irregular space of possible transformations, for which conventional approaches have a high probability of remaining trapped in local extrema.

25.4.2 Parametric Transformations

Small tissue deformations are conveniently modeled using affine or projective transformations. The genetic algorithm presented below is based on a previous work by the authors [12]. In this work, affine transformations were used to register the images, i.e., only translation in the x and y axes, rotation in the z axis, and scaling were considered. Experimental results obtained in that opportunity, showed that the capacity of the algorithm to correctly register images, significantly deteriorated in the presence of very strong misalignments. Further studies allowed us to determine that affine transformations were not enough to properly model the acquisition geometry, and that also rotations in the x and y axes should be taken into account. The projective transformations applied in this work can be defined using homogeneous coordinates such as:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & dx \\ a_3 & a_4 & dy \\ a_5 & a_6 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} u/w \\ v/w \end{pmatrix} \tag{25.2}$$

Therefore, the new coordinates (x',y') of pixels (x,y) in the template image, are given by $x' = (a_1x + a_2y + dx)/(a_5x + a_6y + 1)$ and $y' = (a_3x + a_4y + dy)/(a_5x + a_6y + 1)$.

25.4.3 Similarity Measure

The mutual information measure, successfully applied to multimodal image registration [25, 38], assumes only statistical dependence between image intensities. It treats intensity values in a purely qualitative way, without considering any correlation or spatial information conveyed by nearby intensities. Mutual information tries to reduce entropy and this can be observed as a trend to form intensity clusters in the joint histogram. In the problem of radiography subtraction, since it deals with mono-modal images of natural tissue, the mutual information measure is under-constrained and a functional correlation can be assumed.

The concept of functional dependence, fundamental in statistics, provided us with the framework for the computation of similarity between the two images. To use this concept we consider images as random variables and interpret an image histogram as its probability density function. Furthermore, we consider the 2D histogram of a pair of images as their joint probability density function as proposed in [42]. Thus when a pixel is randomly selected from an image X having N pixels, the probability of getting an intensity i is proportional to the number of pixels N_i in X having intensity i, i.e.,

$$P(i) = \frac{N_i}{N}. \tag{25.3}$$

In order to define the joint probability density function of an image pair, we consider two images (X,Y) and a spatial transformation T that maps the set of pixels of Y, Ω_y, to the set of pixels of X, Ω_x. Since we are working with digitized radiographs, we can also assume that images X and Y take their intensity values from a known finite set $\mathscr{A} = 0,\ldots,255$:

$$X : \Omega_x \to \mathscr{A}$$

$$Y : \Omega_y \to \mathscr{A}.$$

Now, by applying transformation T to image Y, a new mapping is defined from the transformed positions of Y to $\mathscr{A}$:

$$Y_T : T(\Omega_y) \to \mathscr{A},$$

$$\omega \mapsto Y[T^{-1}(\omega)].$$

We now have to find the intensities that a given point of $T(\omega_y)$ simultaneously takes in X and Y_T . Since we are dealing with continuous spatial transformations, points of

the grid $T(\omega_y)$ do not, in general, transform to points of the grid ω_x . So, in order to define the joint probability density function of the images, we used the interpolation approach explained below, discarding the points of $T(\omega_y)$ that do not have eight neighbors in ω_x. If we denote by $T(\omega_y)^*$ the subset of accepted points and by $\tilde{X}$ the interpolation of X, we can define the image pair as the following couple:

$$Z_T : T(\Omega_y)^* \rightarrow \mathscr{A}^2,$$

$$\omega \mapsto \left(\tilde{X}(\omega), Y[T^{-1}(\omega)]\right),$$

and, in a similar way as we did for a single image in Eq. (25.3), their joint probability density function as:

$$P_T(i,j) = \frac{Card\left\{x \mid Z_T(x) = (i,j)\right\}}{Card\, T(\Omega_y)^*}. \tag{25.4}$$

On the other hand, the total variance theorem:

$$Var(Y) = Var\left[E(Y|X)\right] + E_X\left[E(Y|X=x)\right], \tag{25.5}$$

expresses the fact that the variance can be decomposed as a sum of two energy terms: a first term $Var\left[E(Y|X)\right]$ that is the variance of the conditional expectation and measures the part of Y which is predicted by X, and a second term $E_X\left[E(Y|X=x)\right]$ which is the conditional variance and stands for the part of Y which is functionally independent of X.

Now, based on the previous equation that can be seen as an energy conservation equation, we can define the *correlation ratio* as the measure of the functional dependence between two random variables:

$$\eta(Y|X) = \frac{Var\left[E(Y|X)\right]}{Var(Y)}.$$

Unlike the *correlation coefficient* which measures the *linear* dependence between two variables, the correlation ratio measures the *functional* dependence. The correlation ratio takes on values between 0 and 1, where a value near 1 indicates high functional dependence. Then, for a given transformation T, in order to compute $\eta(Y_T|X)$ we can use the following equation:

$$1 - \eta(Y_T|X) = \frac{E_X\left[Var(Y_T|X=x)\right]}{Var(Y_T)},$$

that by means of Eq. (25.4) and Eq. (25.5) can be expressed as:

$$1 - \eta(Y_T|X) = \frac{1}{\sigma^2}\sum_i \sigma_i^2 P_{x,T}(i),$$

where

$$\sigma^2 = \sum_j j^2 P_y(j) - m^2, \;\; m = \sum_j j P_y(j),$$

and

$$\sigma_i^2 = \frac{1}{P_x(i)} \sum_j j^2 P(i,j) - m_i^2, \;\; m_i = \frac{1}{P_x(i)} \sum_j j P(i,j).$$

The correlation ratio measures the similarity between two images, and since it is assumed to be maximal when the images are correctly aligned, it will be used to compute the fitness of the individuals that make up the algorithm population.

25.4.4 Optimization Problem

The problem faced is to find the transformation that maximizes the correlation ratio between a pair of images. The parameters to be found are the eight parameters that define the required projective transformation.

25.4.5 Interpolation Approach

In terms of linear interpolation, the reconstructed signal is obtained by convolution of the discrete signal (defined as a sum of Dirac functions) with a convenient selected kernel. We used spline interpolation due to its accuracy and acceptable computing speed. Spline interpolation of order n is uniquely characterized in terms of a B-spline expansion:

$$s(x) = \sum_{\kappa \varepsilon Z}^{\infty} c(\kappa) \beta^n (x - \kappa),$$

which involves integer shifts of the central B-spline. The parameters of the spline are the coefficients c. In the case of images with regular grids, they are calculated at the beginning of the procedure by recursive filtering. A three-order approximation was used in the present work.

25.4.6 Search Strategy

Evolutionary algorithms (EA) represent a subset of evolutionary computation and use mechanisms inspired by biological evolution: recombination, mutation, and selection. By simulating the natural selection process, where the fittest individuals are more likely to survive, these algorithms can be used to find approximate or even exact solutions to optimization problems. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the environment within which the solutions live, also known as the *cost function*.

Evolutionary algorithms are implemented as a computer simulation in which a population of abstract representations of candidate solutions evolves towards better solutions. These representations are called *chromosomes* (or the genotype of

the genome), and the candidates are called *individuals* or *phenotypes*. Traditionally, individuals are represented as binary strings, but as we shall see, real number encoding is also possible. The evolution usually starts from a population of randomly generated individuals and occurs in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population, recombined and mutated to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either an adequate fitness level has been achieved, a maximum number of iterations has been reached, or, as in our case, the available computational time is exausted.

Despite their computational cost, evolutionary algorithms have been chosen over standard numerical methods because of their strong immunity to local extrema, their intrinsic parallelism and robustness, as well as their ability to cope with large and irregular search spaces. In this section we compare two simple evolutionary algorithms categorized as *parallel iterative* [28]: a Genetic Algorithm (GA) and Differential Evolution (DE). Genetic algorithms are attributed to Holland (1975) [27] and Goldberg (1989) [7], while evolution strategies were developed by Rechenberg (1973) [21] and Schwefel (1995) [19]. A good and diverse set of GA examples is synthetized in Chambers [31], while a practical approach to Differential Evolution can be found in [30]. Both approaches mimic Darwinian evolution and attempt to evolve better solutions through recombination, mutation, and selection. However, some distinctions do exist. DEs are very effective in problems of continuous functions optimization, in part because they use real encoding and arithmetic operators. Since GAs generally encode parameters as binary strings and manipulate them with logical operators, they are more suited to combinatorial optimization.

Upon analyzing the most relevant works in this area, it can be concluded that the most crucial aspects refer to the selection of the coding scheme and the design of the fitness function. All seem to agree that for this kind of optimization problem, real-number encoding performs better than both binary and Gray encoding [34]. Accordingly for the problem at hand, in both evolutionary algorithms the chromosome has been coded as eight floating point numbers representing the set of parameters used in the projective transformation. The initial population includes an individual that is either the null transformation or the center of mass transformation, according to their respective fitness. The rest of the population is generated randomly within the search space.

The fitness of each individual, indicating the similarity between the transformed image and the reference image, is then computed using the correlation ratio previously described. Selection in the GA is performed as follows. The fittest ten percent of the population is selected to be part of the next generation, a facet known as *exploitation*. The rest of the individuals are the result of either crossver ($p_c = 0.85$ in our implementation) or random selection. In the case of crossover, the parents of each new offspring are selected by tournament (5% the size of the population) from the current population. Finally, leaving unmodified the individuals selected by elitism (the evolution history), new candidate individuals are mutated according to a predetermined probability ($p_m = 0.21$), known as the *exploration* characteristic.

Crossover in the GA is performed applying a convex operator as suggested by Davis in [32]. The genes of an offspring chromosome are then the result of a convex interpolation of the parameters of the two mates. A mutation operator is applied to guarantee that the probability of searching a particular subspace of the problem space is never zero. This prevents the algorithm from becoming trapped in local extrema [7]. The mutation operator used, known as *real number creep*, sweeps the individual adding or subtracting a Gaussian distributed random noise to each parameter [32]. The creep operator implemented is a neighborhood search that looks in the vicinity of a good solution, to see if a better one exists.

By contrast, in DE all individuals undergo mutation, recombination, and selection. Mutation starts by randomly selecting three individuals (*vectors* in DE terminology) and adding the weighted difference of two of the vectors to the third, and hence the name *differential mutation*. The resulting vector is called the *donor* vector. For recombination, a *trial* vector is developed from the elements of the target vector and the elements of the donor vector. The elements of the donor vector enter the trial vector with a given probability ($C_r = 0.5$). In this step, to ensure that the trial vector results effectively different from the target one, one of the elements of the donor vector is selected at random and entered directly into the trial vector. Finally, target and trial vectors are compared and the one with the higher similarity measure is selected to be part of the next generation.

This process is repeated in both algorithms until some stopping criterion is reached. In our example, given that we receive a large variety of cases, a predetermined similarity measure is ineffective as the only stopping criterion. For this reason, the actual stopping criterion in our case is given by a maximum number of allowed iterations that is computed as follows. The available processing time span is about 12 hours, and since each iteration takes an average of 250 ms, we can make an estimate of the overall number of iterations that can be performed from one day to the next. Also, according to tests carried out with synthetic images, it has been determined that to obtain acceptable results, at least 200 iterations are required. Based on these considerations and the number of images to be processed, we precompute the number of times the algorithm can be executed for each pair of radiographs in the daily batch. The algorithm is then executed the maximum number of times possible and the best result obtained is the one used for the subtraction process.

25.4.7 Algorithms Distribution

The evaluation of the fitness function consists in applying a projective transformation and then computing the corresponding correlation ratio. This computational intensive operation (see Figure 25.4) is required for each individual of the population. Since the operation can be computed independently for each individual, this part of the algorithm was parallelized and executed on the computational grid previously described. The execution of the evolutionary algorithms uses 120 worker nodes: general purpose workstations, a 1GHz processor on average and memory ranging from 256 to 512 Mbytes. The source code of the implemented distributed

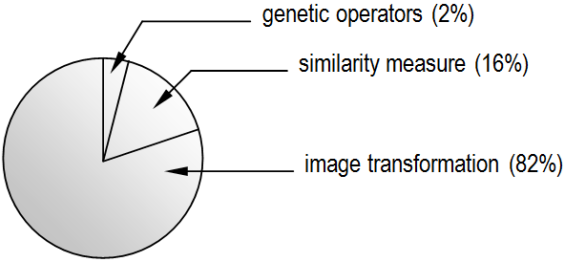


Fig. 25.4 Timing profile for the parallel iterative algorithm showing the percentage time required for each operation

algorithms, as well as additional documentation regarding the computational grid, can be found on the unGrid project site (`http://ungrid.unal.edu.co/`).

25.4.8 Algorithms Validation

To validate the correctness of the evolutionary algorithms implemented, two sets of experiments were conducted. In both cases, the algorithms were compared to a standard numerical method, the Downhill Simplex method devised by Nelder and Mead. This method was chosen because of its ease of implementation and because, amid the standard numerical optimization methods, it is the least sensitive to initial conditions. First, a series of synthetic images was created by applying a set of known transformations to ten reference radiographs. Then the transformed images were registered to the original ones to verify the ability of each algorithm to find the original values used in the transformation. In the second batch of experiments the algorithms were evaluated with pairs of images obtained from real radiographs.

The set of synthetic images was created by applying the transformations shown in Table 25.1, using a reliable image processing program. The algorithms were executed ten times for each pair of reference and template images. The GA and DE algorithms were executed on the computational grid, while the Downhill Simplex implementation was executed on a single machine of the same grid, using its full processor capacity.

For the set of synthetic images, the transformation values and correlation ratio obtained by the Downhill Simplex method, the Genetic Algorithm, and Differential Evolution are presented in Table 25.2, Table 25.3, and Table 25.4, respectively.

From these results, it can be concluded that for deformations in the expected range, both evolutionary algorithms outperform the Downhill Simplex algorithm and provide clinically acceptable registration accuracy. Moreover, Table 25.4 shows that the DE algorithm produces the most accurate results.

For the second experiment, a group of ten intra-oral radiograph pairs, taken on different occasions, was randomly selected from an unrelated study of periodontal therapy. No film holders or any other fixation device were mechanically coupled to the cone of the X-ray machine. Radiographs were digitized in a HP 3570 scanner

Table 25.1 Some combinations of rotation, scaling, and translation applied to the set of synthetic images

	θ_x	θ_y	θ_z	T_x	T_y	SF [a]
1	1	1	1	10	10	0.8
2	10	1	1	10	10	0.8
3	10	10	1	10	10	0.8
4	10	10	10	10	10	0.8
5	10	10	10	100	10	0.8
6	10	10	10	100	100	0.8
7	1	1	1	10	10	1.2
8	10	1	1	10	10	1.2
9	10	10	1	10	10	1.2
10	10	10	10	10	10	1.2
11	10	10	10	100	10	1.2
12	10	10	10	100	100	1.2

[a] SF: scale factor, angles expressed in degrees and translations in pixels.

Table 25.2 Values found by the Downhill Simplex algorithm

	θ_x	θ_y	θ_z	T_x	T_y	SF	CR [a]	$Error\ \%$
1	0.91	0.88	1.17	8.55	8.78	0.88	0.67	12.5
2	12.32	1.15	0.76	8.75	11.74	0.81	0.61	15.6
3	11.15	8.16	0.91	13.28	10.98	0.83	0.64	14.2
4	13.98	7.75	12.24	7.34	8.68	0.79	0.50	21.0
5	8.11	8.12	10.83	76.16	7.74	0.78	0.60	15.8
6	7.98	10.71	12.77	91.21	129.05	0.67	0.56	18.2
7	1.11	0.96	0.87	8.15	10.78	1.08	0.71	10.7
8	7.12	0.95	0.82	9.05	10.54	1.04	0.65	13.3
9	7.67	12.98	1.31	6.55	8.13	1.42	0.40	25.9
10	8.73	9.01	11.96	13.34	12.26	1.41	0.53	19.3
11	10.98	12.85	7.77	87.14	7.76	1.01	0.55	18.6
12	11.96	6.72	6.11	111.17	132.32	1.39	0.42	25.1

[a] CR: correlation ratio.

using a transparent material adapter at a resolution of 600 $\times$ 600 DPI, producing 724 $\times$ 930 pixel images.

Even though acquisition conditions are standardized as much as possible, illumination differences are inevitable. Thus, the histogram of the template image is equalized by using the reference image luminances. This transformation first computes the histogram of each image and then luminances are homogeneously distributed in the template image according to the levels found in the reference image. The properties compared in this experiment were accuracy, in terms of the similarity measure

Table 25.3 Values found by the Genetic Algorithm

	θ_x	θ_y	θ_z	T_x	T_y	SF	CR	$Error\ \%$
1	0.98	1.03	0.96	9.65	10.15	0.81	0.87	2.5
2	11.01	0.99	1.01	9.43	9.96	0.82	0.85	3.5
3	9.47	10.03	1.03	9.15	11.37	0.82	0.81	5.6
4	8.69	8.98	9.21	9.99	10.50	0.79	0.79	6.3
5	10.1	10.93	11.10	96.04	13.34	0.78	0.72	10.2
6	8.76	11.23	11.52	115.88	96.13	0.82	0.71	10.4
7	1.03	1.09	0.99	10.12	10.35	1.21	0.86	3.1
8	9.44	0.95	0.89	9.53	9.66	1.16	0.81	5.5
9	10.91	9.78	1.11	9.01	11.00	1.23	0.77	7.7
10	9.62	10.12	10.52	11.08	11.10	1.22	0.79	6.4
11	9.63	9.21	10.09	92.77	12.96	1.32	0.73	9.5
12	12.08	10.23	8.91	93.49	108.65	1.34	0.72	10.0

Table 25.4 Values found by Differential Evolution

	θ_x	θ_y	θ_z	T_x	T_y	SF	CR	$Error\ \%$
1	0.99	0.99	1.02	9.66	9.78	0.82	0.88	2.0
2	10.05	0.98	0.99	9.55	9.36	0.81	0.87	2.6
3	9.77	10.02	1.03	10.1	10.99	0.83	0.85	3.4
4	9.01	9.57	9.52	9.98	10.02	0.80	0.85	3.5
5	10.2	9.78	11.1	95.99	11.34	0.81	0.81	5.6
6	9.06	10.55	9.43	92.77	117.31	0.82	0.76	7.9
7	0.98	0.97	1.01	9.88	10.02	1.10	0.87	2.6
8	10.2	0.97	0.91	10.53	9.87	1.15	0.84	4.1
9	9.85	10.15	0.97	10.9	10.7	1.19	0.84	3.8
10	10.34	9.88	9.68	10.95	9.15	1.18	0.83	4.6
11	10.65	10.37	9.76	109.01	9.13	1.19	0.82	5.2
12	9.19	9.39	10.91	110.56	92.28	1.17	0.77	7.4

obtained, and efficiency, in terms of execution time and use of resources. All algorithms were coded in the same programming language and use the same routine to compute the correlation ratio between the transformed and reference images. For this comparison, the three algorithms were also executed ten times for each pair of radiographs. A summary of the results obtained is presented in Table 25.5:

As expected, the Downhill Simplex method appeared to be very sensitive to the initial parameters and not always converged to the global optimum. While in some executions it obtained better results than the EAs, in other executions it produced meaningless values and this is reflected in the low overall accuracy shown in Table 25.5. Again, the DE algorithm consistently outperformed the GA and for that reason it is the algorithm currently used in production. It is also important to note

Table 25.5 DS-GA-DE performance comparison

Property	DS	GA	DE
Average Correlation Ratio	0.63	0.81	0.83
Average Execution Time (secs)	52	50	48
Number of CPUs	1	120	120

that the computational grid, used to run the EAs, only uses the free CPU cycles of the computers that make it up.

Fig. 25.5 shows a pair of radiographs to be subtracted (top row). The bottom row displays subtraction without geometric correction on the left and with correction on the right. Null intensity level is shifted to 128 in order to make tissue changes easily observed. In this particular example, it can be appreciated that the match is precise enough to make objective measurements despite the fact that in the second radiograph, the fifth tooth (from left to right) is nearly hidden. The small spot, possibly an artifact, that appears in both images is observed in the resulting image in white, indicating that new tissue developed. In this image it can also be observed that a difference appears at the root of the third tooth which corresponds to new tissue developed after treatment. These changes are impossible to observe in the raw difference image (bottom left). Similarly, in this image the bone pattern is blurred and impossible to recognize, while in the resulting image the trabecular bone pattern is clear. For the entire set of test images, matching has been visually assessed by two experts in the field. They judged that the alignment was sufficiently accurate to get objective measurements while maintaining acceptable computation times.

For the GA, 4580 experiments were performed in order to guarantee a complete analysis of the parameter space. An experiment is the execution of the algorithm with a particular set of images and parameters, i.e. population size, tournament size and genetic operators probabilities. In this task the grid became an essential tool and allowed us to achieve a second level of parallelism. The first analysis was conducted to determine two basic parameters of the algorithm: population size and selection scheme used to choose the parents for crossover. The experiments showed that the optimum population size for this problem is 120. Two common selection options are tournament selection and elitism. In tournament selection of size N, N individuals are selected at random and the fittest is chosen. Elitism is a particular case of tournament selection where the size of the tournament equals the size of the population, so the best individual is always preserved. For this problem, tournament selection of size 12 is the best option for selecting the parents for a new generation. The other parameters analyzed were the crossover and mutation probabilities. The combination of probabilities that yielded the best results were 0.85 and 0.21 respectively.

Another advantage of the DE algorithm over the GA is that it only uses two parameters: the scale factor F, that controls the rate at which the population evolves,

Fig. 25.5 The upper row shows the two images to subtract. Bottom row shows the subtracted images: left without geometrical correction and right after automatic correction

and the uniform crossover probability C_r. This makes the analysis of the parameter space simpler and therefore tuning of the algorithm becomes easier. The values found for the DE algorithm are $F = 0.5$ and $C_r = 0.5$.

25.4.9 The Subtraction Service

The medical imaging community has a growing need for Internet-aware tools that facilitate interaction with remote data in a collaborative way. Such medical imaging data typically requires special-purpose tools that come in the form of stand-alone and non-portable applications, i.e., software to be manually installed and maintained on a local computer. This is possible provided there is an available binary version for the particular platform in use, or the source code is publicly obtainable and the user is responsible for gathering the required libraries and tools, and compiling the source code. There is also a growing need in the medical imaging community for Internet-aware software tools that facilitate collaborative data analysis and visualization. Research projects and clinical studies require a medium for a geographically dispersed scientific and clinical community to interact and examine medical imaging data via the Internet. Additionally, health care and medical research rely increasingly on the use of computationally intensive algorithms.

The service-oriented model proposed has many advantages over the stand-alone application model. First and foremost, it avoids the user having to manually install the software. All that is needed is a standard web browser and an Internet connection: the platform dependency is no longer an issue. In addition, updates are made on the server side and automatically propagate to all service users. By having redundant and clustered servers it is possible to attain high data availability, something difficult -if indeed possible- with a personal workstation. Finally, and most important, the service-oriented paradigm leverages interdisciplinary and collaborative work, one critical success factor in biomedical practice and research. This section will present a service-oriented model for medical imaging via the Internet. Services are accessed via a standard web browser, however the essential tools also work off-line. This is accomplished by using a local server and a local database, and synchronizing data when the user goes back online.

The proposed architecture for this model is basically an enhanced version of a standard client-server architecture (see Figure 25.6). The difference lies in the addition of a local server that allows for the basic services to keep functioning without an active Internet connection. The proxy component is responsible for providing the essential functionality when the user is disconnected from the Internet, and for synchronizing data with the server when the connection is active again. To accomplish this, the local server is connected to a lightweight relational database engine that allows the client application to store, search and recover data using the structured query language (SQL).

On the client side, services are accessed using a standard web browser. In our implementation, all services use a digitally-signed Java browser extension (“applet”) that takes care of the installation of the local server and other required tools such as the database engine (e.g. Apache Derby) and OpenGL[5] libraries. Additionally, the applet provides the tools for visualization, manipulation and light processing tasks such as image reconstruction and geometric transformations. On the server side, and behind a pool of web servers, there is a high-availability cluster that provides access

[5] OpenGL is a registered trademark of Silicon Graphics Inc.

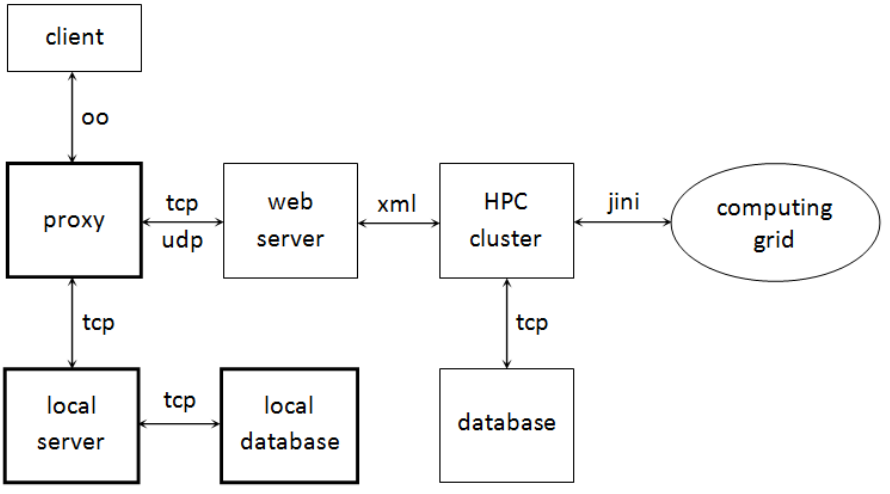


Fig. 25.6 Overall architecture of the subtraction radiography service, showing the protocols used for communication between neighbouring components

to the computing grid (HPC). The cluster is based on the Rocks cluster distribution [13] and Sun Grid Engine [37], and uses peer-to-peer technologies - a replicated, distributed, transactional tree-structured cache - to avoid the appearence of a single-point of failure (HA).

The subtraction service provides two modes of operation: an *interactive mode* and a *batch mode*. In the first mode, the user loads the images to register, and interactively drags, rotates and scales the template image to align it manually. Once registered, the images can be subtracted and the difference visualized. Since this is a lightweight operation, it is carried out completely on the client side. However, the user can choose to register the images automatically. In this case, the images are uploaded to the server (if not already there), and then registered by the aforementioned distributed algorithm. The parameters of the projective transformation are then sent back to the client application for visualization. With the help of the local server and database, the interactive mode keeps working even without an active Internet connection, provided the images reside in the local machine. This is what happens most of the time, either because the images were produced on the local machine, or because they were previously downloaded from the server. The synchronization process is the responsibility of the so called *proxy* element that the client application actually communicates with: it uploads the locally digitized images to the server, and downloads the images stored on the server to the corresponding client computers. Figure 25.7 shows the graphical user interface of the service.

In practice, the service is mostly used in the second or *batch mode*. In this mode the set of radiographs taken daily are digitized and uploaded to the service server where they are registered automatically by the same evolutionary algorithm. The job of the master process, executed in the cluster, is to generate the initial populations and send them to the computing grid for evaluation. Since the fitness of each

Fig. 25.7 Graphical user interface for the radiography subtraction service

individual can be evaluated independently from the others, this task is performed in parallel on the grid. Once evaluated, each population is collected by the corresponding master process that applies the genetic operators (mutation, recombination, selection), produces a new population and sends it again to the grid for evaluation. The process repeats until the stop conditions are met. The optimal transformations are then stored in the server database and sent to the client applications for visualization.

The use of Java and Internet technologies in medical imaging is not new. These technologies have been used in radiology teaching files, to access information in multimedia integrated picture archiving and communication systems (PACS), and for teleradiology purposes. However, all known approaches seem to assume the existence of a reliable and stable Internet connection, and this is not always possible.

25.5 Discussion and Conclusions

In this chapter, we have reviewed the main aspects involved in the task of medical image registration. Through a simple yet real case study, we have seen one effective way to build a high performance computational grid that benefits from an existing computational infrastructure. The framework is based on a loosely-coupled architecture and asynchronous communications. Using a service-oriented architecture that provides a shared and distributed memory space and other essential services (publication, discovery, transactions, leasing), we have implemented a general-purpose computing grid and shown its application in medical imaging. The computing grid is most appropriate for the parallel execution of computationally intensive jobs that can be divided in multiple independent tasks. Other crucial aspects, like security, storage management, and general system administration, are out of the scope of this

chapter and have been omitted. However, it is worth noticing that these are essential aspects that in research and clinical environments have to be properly addressed in order to provide secure and reliable medical imaging services.

To demonstrate the applicability of the computing grid in a real situation, we have presented the case study of automatic digital subtraction. In this situation, we evaluated two evolutionary algorithms as the search strategy to solve an expensive optimization problem. This is to find a global maximum of an unknown function that measures the similarity between two given images. In this evaluation, i.e., for this particular problem, differential evolution proved to be more performant and reliable than the genetic algorithm. The global structure of the algorithm is iterative, but since individuals in a population can be evaluated independently from the others, the most time-consuming stage of the algorithm is computed in parallel. This, and the simple yet powerful API of JavaSpaces, allowed us to easily implement the devised solution.

The high computational cost of the evolutionary algorithm in use was addressed by developing a distributed implementation. This implementation exploits the computational power of a set of personal computers arranged in a low cost computational grid. Since it can be deployed over an existent computational infrastructure, this approach can be affordably implemented in institutions with a low budget, like public and university hospitals.

The proposed service-oriented model for medical imaging is feasible and useful in research and clinical scenarios, and is used daily in the School of Dental Medicine. The implemented framework allows doctors to use up to date medical imaging techniques and high-performance computing power in routine clinical studies, by means of a standard web browser and without specialized training. Furthermore, the framework allows for new services to be obtained from the integration of existing services with different dynamics, such as 2D/3D/4D, and video processing tools.

We are currently working on evolving the architecture in use towards a *cloud computing* model, in which the common theme relies on integrated services over the Internet to satisfy the clinician computing needs. Regarding the algorithms used in registration, current and future work is related to further exploring hybrid evolutionary algorithms such as those presented in [4, 16, 35], their possible application to 3D curvature-based registration [23], as well as their distribution and parallelization.

Acknowledgements. The authors would like to give special thanks to Professors Fabio A. González, Germán J. Hernández, Luis F. Niño, and Mark J. Duffy for their invaluable help and advice.

Appendix

This section shows the use of the *Command* pattern and JavaSpaces to build a generic worker node. This pattern was first introduced by Gamma *et al.* [10] in object-oriented software design, and is used in a variety of domains. In our case it was used to create a worker application capable of servicing requests of any master process, in other words, to build a *generic worker*. In this context, to implement the Command pattern, a class must implement the following interface:

```
public interface Task
{
     Result execute();
}
```

To benefit from this pattern, a master process has to break the job at hand into `TaskEntry` objects and write them to the space. The `TaskEntry` class, given as an example here, implements the `Entry` interface (a *tagged* JavaSpaces interface, without methods) and the `execute()` method declared in the `Task` interface described before.

```
public class TaskEntry implements Task, Entry
{
     public Result execute()
     {
          . . .
     }
}
```

An outline of the `Master` class is presented below:

```
public class Master
{
     public void generateTasks()
     {
          for ( int t = 0; t < numTasks(); t++ )
          {
               writeTask( getTask( t ) );
          }
     }

     public void collectResults()
     {
          for ( int t = 0; t < numTasks(); t++ )
          {
               mergeResult( takeResult() );
          }
     }
}
```

In practice, the routines that generate tasks and collect results are run concurrently in two separate execution threads. This is because this process can be executed asynchronously: as soon as the tasks start to generate and are written into the space, worker nodes can start computing them without having to wait for this mapping process, therefore speeding up the whole computation.

A simplified version of the generic worker would then look like:

```
public class Worker
{
     public void run()
     {
          for ( ;; )
          {
               Task t = takeTask();
               Result r = t.execute();
               writeResult( r );
          }
     }
}
```

Again, in practice, the actual `Worker` class spawns several worker threads to compute multiple tasks concurrently, depending upon the number of processors (and cores) available. Internally, the method `takeTask()` calls the JavaSpaces method `take()` which is blocking, therefore releasing the processor where the worker thread is running.

For the sake of clarity we have also omitted some important details such as those regarding the use of transactions, leases, job priorities and caches. Transactions are used to guarantee that a master process acts as a standalone application: if it writes a number of tasks into the space, then it must receive the same number of results (complete success), or none (complete failure). That is, to avoid partial failure, workers compute every task under a transaction. If the task is completed successfully, then the worker writes the result into the space and commits the transaction, otherwise, the transaction is cancelled and the task is returned to the space. These semantics are provided by a *two-phase commit protocol* that is performed by the Jini transaction manager[6]. Using transactions, the pseudo code for the `Worker` class would now be:

```
public class Worker
{
     public void run()
     {
          for ( ;; )
```

[6] For detailed information on this protocol, please refer to the Jini Transaction Specification (`http://www.jini.org/transactions`).

```
            {
                createTransaction()

                try
                {
                    Task t = takeTask();
                    Result r = t.execute();
                    writeResult( r );

                    commitTransaction();
                }
                catch ( Exception e )
                {
                    cancelTransaction();
                }
            }
        }
    }
```

The use of transactions alone does not guarantee that partial failure will not occur. Let us suppose a worker node takes a task from the space and starts computing it. Then, if some component fails (e.g., the worker application, the operating system, the computer is shut down or disconnected from the net) before the task finishes execution, the transaction will not be committed, nor cancelled, and the result will never be written into the space, causing the master process to wait indefinitely. The solution to this situation lies in the use of *leases*.

Leasing in JavaSpaces provides a way of allocating resources for a fixed period of time, after which the resource is freed. In the context of our worker class, when it calls the JavaSpaces `take()` method to fetch a task object for execution, it supplies a lease parameter that specifies the amount of time that it will hold the task object. If the task is executed before the lease runs out, the result is written into the space and the operation finishes. Otherwise, the lease keeps being renewed until the tasks finishes. If the lease is not renewed, an indication that something went wrong in the worker node, the task is returned to the space by the Jini lease manager[7], so another worker node can compute it.

References

1. Petersson, A., Ekberg, E.C., Nilner, M.: An evaluation of digital subtraction radiography for assessment of changes in position of the mandibular condyle. Dentomaxillofacial Radiology 27, 230–235 (1998)

[7] For detailed information on leasing, please refer to the Jini Leasing Specification (`http://www.jini.org/leasing`).

2. Farag, A.A., Yamany, S.M., Nett, J., Moriarty, T., El-Baz, A., Hushek, S., Falk, R.: Medical Image Registration: Theory, Algorithm, and Case Studies in Surgical Simulation, Chest Cancer, and Multiple Sclerosis, ch. 1, vol. 3, pp. 1–46. Kluwer Academic/Plenum Publishers, New York (2005)
3. Apache. Apache river (2007), http://www.apache.org/river
4. Grosan, C., Abraham, A., Ishibuchi, H.: Hybrid Evolutionary Algorithms. Springer, Heidelberg (2007)
5. Gelernter, D.: Generative communication in linda. ACM TRansactions on Programming Languages and Systems 7(1), 80–112 (1985)
6. Rueckert, D.: Non-rigid Registration: Concepts, Algorithms and Applications. Biomedical Engineering, ch. 13, pp. 281–301. CRC Press, Florida (2001)
7. Goldberg, D.A.: Genetic algorithms in search, optimization and machine learning. Addison-Wesley Professional, Reading (1989)
8. Hawkes, D.J.: Registration Methodology: Introduction. Biomedical Engineering, ch. 2, pp. 11–38. CRC Press, Florida (2001)
9. Freeman, E., Hupfer, S., Arnold, K.: JavaSpaces Principles, Patterns, and Practice. Prentice Hall PTR, Englewood Cliffs (1999)
10. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Elements of Reusable Object-Oriented Software. Addison-Wesley Professional, Reading (1994)
11. Berman, F., Fox, G., Hey, A.J.G.: Grid Computing: Making The Global Infrastructure a Reality. Wiley, Chichester (2003)
12. Mañana, G., Romero, E., González, F.: A grid computing approach to subtraction radiography. In: IEEE International Conference on Image Processing, pp. 3225–3228 (2006)
13. Rocks Group. Rocks clusters (2008), http://www.rocksclusters.org/
14. Grondahl, H., Grondahl, K.: Subtraction radiography for the diagnosis of periodontal bone lesions. Oral Surgery 55, 208–213 (1983)
15. Lester, H., Arridge, S.R.: A survey of hierarchical non linear medical image registration. Pattern Recognition 32(1), 129–149 (1999)
16. Talbi, H., Batouche, M.: Hybrid particle swarm with differential evolution for multimodal image registration. In: IEEE International Conference on Industrial Technology, pp. 1567–1572 (2004)
17. Cordón, H.F., Damas, S., Santamaría, J.: A chc evolutionary algorithm for 3d image registration. In: De Baets, B., Kaynak, O., Bilgiç, T. (eds.) IFSA 2003. LNCS, vol. 2715, pp. 440–441. Springer, Heidelberg (2003)
18. Gómez García, H.F., González Vega, A., Hernández Aguirre, A., Marroquín Zaleta, J.L., Coello Coello, C.A.: Robust multiscale affine 2D-image registration through evolutionary strategies. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 740–748. Springer, Heidelberg (2002)
19. Schwefel, H.P.: Evolution and Optimum Seeking: The Sixth Generation. Wiley-Interscience, New York (1995)
20. De Falco, I., Della Cioppa, A., Maisto, D., Tarantino, E.: Differential evolution as a viable tool for satellite image registration. Applied Soft Computing 8, 1453–1462 (2008)
21. Rechenberg, I.: Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution, pp. 305–324. Frommann-Holzboog, Stuttgart (1973)
22. Beutel, J., Sonka, M., Kundel, H.L., Fitzpatrick, J.M., Van Metter, R.L.: Medical Image Processing and Analysis, vol. 2, pp. 447–513. SPIE Press, Belligham (2000)
23. Modersitzki, J.: Numerical Methods for Image Registration. Oxford University Press, Oxford (2004)

24. Nelder, J., Mead, R.A.: A simplex method for function minimization. The Computer Journal 7(4), 308–313 (1965)
25. Maintz, J.B.A., Viergever, M.A.: An overview of medical image registration methods. In: Symposium of the Belgian Hospital Physicists Association, SBPH/BVZF (1997)
26. JBoss. Jboss application server (2008), http://www.jboss.org/jbossas
27. Holland, J.H.: Adaptation in natural and artificial systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. MIT Press, Massachusetts (1992)
28. Bahi, J.M., Contassot-Vivier, S., Couturier, R.: Parallel Iterative Algorithms: From Sequential to grid Computing. Chapman & Hall/CRC, Boca Raton (2008)
29. Hajnal, J.V.: Introduction. Biomedical Engineering, ch. 1, pp. 1–8. CRC Press, Florida (2001)
30. Price, K.V., Storn, R.M., Lampinen, J.A.: Differential Evolution: a practical approach to global optimization. Springer, Heidelberg (2005)
31. Chambers, L.: The practical handbook of genetic algorithms: Applications, 2nd edn. Chapman & Hall/CRC, Boca Raton (2000)
32. Davis, L.: Handbook of genetic algorithms, 2nd edn. Chapman & Hall/CRC, Boca Raton (2000)
33. Eshelman, L.J.: Real-coded genetic algorithms and interval schemata, vol. 2, pp. 187–202. Morgan Kaufmann Publishers, Belligham (1993)
34. Gen, M., Cheng, R.: Genetic Algorithms and Engineering Optimization. Wiley-Interscience, Hoboken (2000)
35. Lozano, M., García-Martínez, C.: Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. Computers & Operations Research (in press, 2009)
36. Powell, M.: An efficient method for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal 7(2), 155–162 (1964)
37. Sun Microsystems. Grid engine (2008), http://gridengine.sunsource.net/
38. Viola, P., Wells III, W.: Alignment by maximization of mutual information. International Journal of Computer Vision 24, 137–154 (1997)
39. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
40. Sterling, T., Becker, D.: Beowulf (2008), http://www.beowulf.org/
41. Terracotta. Terracotta (2008), http://www.terracotta.org/
42. Pennec, X., Roche, A., Malandain, G., Ayache, N.: Multimodal image registration by maximization of the correlation ratio (1998), http://hal.archives-ouvertes.fr/
43. Yuan, X., Zhang, J., Buckles, B.P.: Evolution strategies based image registration via feature matching. Information Fusion 5, 269–282 (2004)

Chapter 26
Speeding-Up Expensive Evaluations in High-Level Synthesis Using Solution Modeling and Fitness Inheritance

Christian Pilato, Daniele Loiacono, Antonino Tumeo, Fabrizio Ferrandi, Pier Luca Lanzi, and Donatella Sciuto

Abstract. High-Level Synthesis (HLS) is the process of developing digital circuits from behavioral specifications. It involves three interdependent and NP-complete optimization problems: (i) the *operation scheduling*, (ii) the *resource allocation*, and (iii) the *controller synthesis*. Evolutionary Algorithms have been already effectively applied to HLS to find good solution in presence of conflicting design objectives. In this paper, we present an evolutionary approach to HLS that extends previous works in three respects: (i) we exploit the NSGA-II, a multi-objective genetic algorithm, to fully automate the design space exploration without the need of any human intervention, (ii) we replace the expensive evaluation process of candidate solutions with a quite accurate regression model, and (iii) we reduce the number of evaluations with a fitness inheritance scheme. We tested our approach on several benchmark problems. Our results suggest that all the enhancements introduced improve the overall performance of the evolutionary search.

26.1 Introduction

High-Level Synthesis (HLS) [8] is concerned with the design and implementation of digital circuits starting from a behavioral description, a set of goals and constraints, and a library of different types of resources. HLS typically consists of three steps: the scheduling, the resource allocation and the controller synthesis. The scheduling assigns each operation to one or more clock cycles (or control steps) for the execution. The resource allocation assigns the operations and the produced values to the hardware components and interconnects them using connection elements. Finally, the controller synthesis provides the logic to issue data-path operations, based on the control flow. Unfortunately, it is non-trivial to solve these problems as they

Christian Pilato · Daniele Loiacono · Antonino Tumeo · Fabrizio Ferrandi · Pier Luca Lanzi · Donatella Sciuto
Politecnico di Milano, Dipartimento di Elettronica ed Informazione
{pilato,loiacono,tumeo,ferrandi,lanzi,sciuto}@elet.polimi.it

Y. Tenne and C.-K. Goh (Eds.): Computational Intel. in Expensive Opti. Prob., ALO 2, pp. 701–723.
springerlink.com

are NP-complete and strongly interdependent. In addition, the high-level synthesis problem is multi-objective and most of the design objectives are contrasting by nature. Therefore, developers usually apply an iterative refinement cycle: at each step they (i) manually apply transformations, (ii) synthesize the design, (iii) examine the results coming from the synthesis of the design solution, and (iv) modify the design to trade off the design objectives. This process is usually called *design space exploration*.

Evolutionary algorithms (EAs) have been successfully applied [12, 21] to such complex explorations, since their behavior is very similar to the one of a designer: they iteratively improve a set of solutions (i.e. alternative designs) using the results of their evaluations as a feedback to guide the search in the solution space. In addition, EAs proved to work well on large optimization problems even if (i) the search space is constrained, (ii) there are few information available on EAs can also easily deal with different objectives, without the need of combining them into a single objective function. The main drawback of EA approaches is the need to evaluate a huge number of design alternatives. This is a serious concern as in HLS problems the solution evaluation is a very expensive process. To meet the time-to-market constraints we need to shorten the design process without reducing the quality of the solutions discovered.

In this paper we present an evolutionary framework to perform a fully automated design space exploration for HLS problems. In addition, to compute the fitness of the evolved solution we replace the usual expensive evaluation process with a cost model coupled with an inheritance fitness scheme. In particular, our approach extends previous works on the application of EAs to HLS [14, 21, 24, 25] basically in three respects: (i) while in previous works focused on evolutionary approaches to optimize a human designed objective function, we exploit *NSGA-II* [9], a multi-objective genetic algorithm, to perform a fully automated design space exploration; (ii) we exploit a regression model to perform a fast and quite accurate evaluation of the candidate solutions; (iii) to our knowledge, this is the first work that applied a fitness inheritance scheme to HLS in order to reduce the number of evaluations. We validated our approach on several benchmark problems. Our empirical results suggest that both the regression model introduced and the fitness inheritance scheme result in an improvements of the design space exploration process.

This chapter is organized as follows. After discussing relevant work in Section 26.2, we describe our approach in Section 26.3. In Section 26.4 we discuss the issues of the solution evaluation. Then, we present two different techniques to reduce expensive evaluations: cost modeling and fitness inheritance [29, 34]. The former technique, presented in Section 26.5, consists of replacing a part of the expensive solution evaluation with a prediction model that, given some relevant features of the solution, provides an estimation of its objective values, dramatically reducing the cost. The latter technique, detailed in Section 26.6, allows to reduce the number of fitness evaluations by replacing some of them with a surrogate, based on the fitness values of other individuals previously evaluated. Experimental evidences on a set of historical benchmarks for the HLS problem, both in terms of

quality of the solutions with respect to the design objectives and overall execution time of the exploration, are presented and discussed for each technique.

26.2 Related Work

The common techniques used in high-level synthesis can be classified into three categories: *exact*, *heuristic* and *non-deterministic approaches*.

The *exact approaches* [7, 18] exploit mathematical formulations of the problem and may find the optimal solution. Unfortunately, their computational requirements grow exponentially with the size of the problem and are impractical for large designs.

The *heuristic approaches* [8, 28, 35] work on a single operation or resource at once and perform continuous refinements on the set of solutions. The decision process is deterministic, so they do not explore all the design alternatives, possibly leading to sub-optimal solutions. Furthermore, most of these techniques perform the scheduling and the allocation sub-tasks separately, with the scheduling usually performed as the first step.

To support scalability and to explore a larger set of alternative designs, several *non-deterministic approaches* (e.g., [20]), and in particular GAs [12, 14, 21, 24, 25], have been efficiently applied to HLS. Most of them focused on only one of the HLS sub-task. In [24], GAs are used to schedule the operation, while in [25] they are used to allocate and bind a scheduled graph. Grewal et al. [14] implemented a hierarchical genetic algorithm, where genetic module allocation is followed by a genetic scheduling. Araújo et al. [1] used a genetic programming approach, where solutions are represented as tree productions (rephrased rules in the hardware description language grammar) to directly create Structured Function description Language (SFL) programs. This work presents a different approach w.r.t. the previous ones, but it is difficult to control the optimizations. Krishnan and Katkoori [21] proposed a priority-based encoding, where solutions are represented as a list of priorities that defines in which order the operations should be chosen by the scheduling algorithm. However, they performed a single-objective optimization using a weighted average of the design objectives, that has been proved to be not effective [39]. Several works [12, 25] introduced a binding-based encoding, also for system-level synthesis [27, 36], where solutions are represented as the binding between operations and functional units where they will be executed. In some of these approaches the exploration can generate unfeasible solutions that have to be recovered or discarded, wasting time and computation resources.

EAs often require to evaluate a number of candidate solutions that might easily result computationally unfeasible. This generally happens in real-world problem and it is also the case of HLS. Accordingly, in the literature several *evaluation relaxation* [13] techniques have been introduced to speedup EAs: an accurate, but computationally expensive, fitness function is replaced by a less accurate, but inexpensive, surrogate. Following the early empirical design, theories have been developed to understand the effect of approximate surrogate functions on population

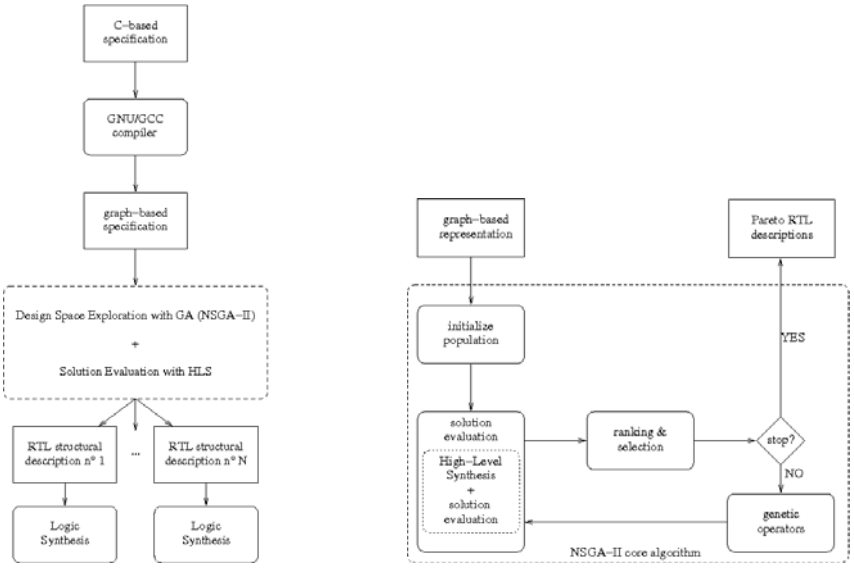
sizing and convergence time and to enhance speedups (see [31] for further details). The surrogate can be either endogenous [34] or exogenous [2, 19, 23]. *Fitness inheritance* [34] is one of the most promising endogenous approach to evaluation relaxation: the fitness of some proportion of individuals in the population is inherited from the parents. Sastry et al. [33] use a model based on least squares fitting, applied in particular to extended compact genetic algorithm (eCGA [16]). Chen et al. [6] present their studies on fitness inheritance in multi-objective optimization as a weighted average of parent fitness, decomposed in the different n objectives. Recent studies investigated the impact of fitness inheritance on real-world applications [11] and different exploration algorithms [30]. Exogenous surrogate are typically used in engineering applications [2, 10] and consists of developing a simplified model of the real problem to provide an inexpensive surrogate of the fitness function. In particular, in HLS several simplified models for area and timing have been proposed in the literature. In [26], simple metrics are proposed to drive the optimization algorithms, even if some elements are not correctly considered (e.g., steering logic or effects of optimizations performed by the logic synthesis tools). In [3] the area is estimated with a linear regression approach that is also able to model the effects of the logic optimizations. Unfortunately, most of the models proposed provide a poor guidance to the optimization process as they do not take into account the resource binding and the interconnections [5]. In this work we focus on data-flow applications that involve only area models, however we refer the interested reader to [4, 22] for timing estimation models.

26.3 Design Space Exploration with Multi-Objective Evolutionary Computation

The proposed methodology is shown in Figure 26.1(a). The inputs are the behavioral description of the problem in C language, a library of resource descriptions and a set of constraints to be met, specified in XML format. We exploit a customized interface to the GNU *GCC* compiler[1] to generate the related GIMPLE, representing the behavioral specification. From this, a combined control and data dependencies data structure (*CDFG*) is built. CDFG allows the identification of the operations that should be mapped and scheduled on the functional units described in the resource library provided as input, as well as of the precedences among them.

The core of our methodology, shown in Figure 26.1(b) and detailed in Section 26.3.1, is the design space exploration that exploits a multi-objective GA (*NSGA-II* [9]) to concurrently optimize the different design objectives: the area and the latency. This design space exploration iteratively improves a set of candidate solutions to explore the most promising design sub-spaces. Finally, a Register-Transfer Level (RTL) specification in a hardware description language (e.g. VHDL, Verilog or SystemC) is generated for each one of the non-dominated solutions contained into the final population resulting from the exploration algorithm.

[1] GCC - GNU Compiler Collection, *http://gcc.gnu.org*



(a) Overall Methodology (b) Design Space Exploration

Fig. 26.1 Methodology overview: squared boxes represent intermediate representations, rounded boxes represent tools. The dashed box represents the methodology described in this chapter

26.3.1 Design Space Exploration Core

The design space exploration core is shown in Figure 26.1(b). Initially a *population* of N candidate solutions is randomly created and evaluated through a complete high-level synthesis, with respect to the design objectives. In the current implementation, area and performance have been considered. Once the evaluation is completed, the solutions are sorted. After the initialization step, a new population of N candidate solutions is created. In particular, each element of the new population, called *offspring*, is generated by applying the common genetic operators (i.e., *crossover* and *mutation*) to the existing solutions, called *parents*. Finally, each offspring created is evaluated as well and added to the population. The resulting population of size $2N$ is then sorted again and the worst N solutions are discarded. All the steps described above, except the initialization, are thus iteratively repeated until the following stopping criterion is met. Whenever the set of best solutions is not improved in the last 10 iterations, the size of the population, N, is increased by 50%. When, even increasing the population size, no best solutions are found, the optimization process is stopped. At the end, the non-dominated solutions found by the exploration algorithm are returned.

26.3.2 Genetic Algorithm Design

In this section, we present the design of the NSGA-II that drives our design space exploration.

Solution encoding. In this methodology, the chromosome is simply a vector where each gene describes the mapping between the related operation in the behavioral specification and the functional unit where it will be executed. With this formulation both the resource allocation (i.e., the total number of required functional units) and the operations binding (i.e., the assignment of the operations to the available units) are encoded at the same time and all the information that is necessary to generate the structural description of the design solution is encoded. This encoding was introduced for the first time in [12] and is inspired to the approach proposed in [25]. The main advantage in using this encoding is that all genetic operators (see Section 26.3.2) create feasible solutions. In fact, the recombination of the operations binding simply results in a new allocation or binding. In this way, good solutions can be obtained just using common genetic operators, without needing procedures to recover unfeasible solutions.

Initial population. At the beginning of each run, an initial population of admissible resource bindings is created. It can be created by random generation or, to cover a larger design space and to speedup the exploration process, by generating some known solutions (e.g. the one with the minimum number of functional units or the minimum latency). This allows the algorithm to start from some interesting points and then to explore around to improve them.

Fitness function. To evaluate the solutions we used the following multi-objective fitness function:

$$F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix} = \begin{bmatrix} Area(x) \\ Time(x) \end{bmatrix} \tag{26.1}$$

where $Area(x)$ is an estimation of the area occupied by the solution x, $Time(x)$ is an estimation of the latency of the solution x, computed as the worst case execution time of the scheduled solution, in terms of clock cycles. The goal of the genetic algorithm is to find the best trade-offs with respect to this cost function.

Ranking and selection. The ranking of solutions is an iterative process. At iteration k, all the solutions are first sorted according to the *fast-non-dominated-sort*. Then, the non-dominated solutions are classified as solution at the k-level and removed from the solutions to be ranked. The process is repeated until all the solutions have been ranked. At the end of the evolutionary process, the whole set of solutions ranked as the best ones will be the outcome of the optimization. We refer to this set as the approximation of the *Pareto-optimal* set discovered by the evolutionary process.

Genetic operators. To explore and exploit the design space, the usual genetic operators are used, the *unary mutation* and the *binary crossover*. The two operators are applied respectively with probability P_m and P_c. *Mutation* is an operator used for finding new points in the search space. Mutation has been implemented with a

relatively low rate (e.g., P_m=10%) and it is applied as follows: each gene is modified with probability P_μ, changing the corresponding binding information. *Crossover* is a reproduction technique that mates two parent chromosomes and produces two offspring chromosomes. Given two chromosomes, a standard *single-point* crossover is applied with a high probability (e.g., P_c=90%). The crossover mechanism mixes the binding information of the two parent solutions.

26.3.3 Performance Measure

The outcome of multi-objective EAs is a set of solutions that represent the best estimate of the *Pareto front* in the objective space. Accordingly, evaluating and comparing the outcome of different EAs is not trivial as it is in single-objective optimization. In particular, several metrics have been introduced in the literature [38] with different features and aims. In general a performance metric can provide either a relative measure (e.g., Non Dominated Combined Set Ratio [37]) or an absolute measure (e.g., $\mathscr{S}$ metric [38]). The former type of metric are devised to compare only two set of solutions, while the latter allow to rank several set of solutions on a specific problem. In this work we used a performance metric that is equivalent to the $\mathscr{S}$ metric as (i) we need to compare several set of solutions and (ii) it is a scale-independent metric. In minimization problems with two objectives the $\mathscr{S}$ metric can be computed as the hyper-volume between the set and the anti-ideal objective vector [17]. Unfortunately in HLS, the anti-ideal objective vector is not always defined. Accordingly we set each objective of the anti-ideal vector to the worst value discovered during all the evolutionary runs. In addition, for a better readability we defined the performance measure as the area *complementary* to the $\mathscr{S}$ metric in the positive quadrant. Accordingly, the smaller is the used metric, the better the set of solutions is.

26.4 Solution Evaluation

The crucial point to obtain a fast and effective convergence of the exploration is the quality of the solution evaluation. In particular, the values of the fitness function should be as close as possible to the effective values that would be obtained through the actual implementation of the design on the target technology and the evaluation of the desired characteristics (e.g., area or latency). For this reason, the best fitness is obtained with a complete synthesis of each design solution. A complete synthesis includes the following two steps. First, a high-level synthesis flow is performed from a fully specified design solution to a structural description of the circuit. Then a logic synthesis step is applied to generate the circuit for the target technology from its structural description.

Our approach targets the Field Programmable Gate Array (FPGA) technology. A FPGA is a semiconductor device that can be configured by the customer or the designer after manufacturing. FPGAs are becoming an interesting alternative to Application Specific Integrated Circuits (ASICs) as they allow to customize the system

without the need of an expensive development process. In particular, FPGAs fit the needs of embedded systems design where they are used to develop accelerators specific to improve the performance of the applications that will be used on the systems. Nevertheless, the choice of this technology introduces additional difficulties in the design process that tools for high level synthesis and design space exploration need to address. FPGAs are composed by a set of configurable logic units, typically Look Up Tables (LUTs) with four inputs and one output, that are used to represent the logic functions, and a series of flip-flops. These elements are organized in Configurable Logic Blocks (CLBs) that communicates through a programmable interconnection network. Modern FPGAs may also feature dedicated blocks for some type of operations (e.g. hardware multipliers) and embedded memories. The generation of a FPGA based design requires to process the specification of the circuit in a hardware description language with a synthesizer, like Integrated Software Environment (ISE) for Xilinx devices or Quartus for ALTERA solutions. A FPGA synthesis tool follows several stages. The first stage (synthesis) transforms the specification into a set of logic primitives and memory elements for the reconfigurable device. The second stage (mapping) maps these basic components to the specific device available. In the last stage (routing) the blocks are connected together and with the input/output pins. The initial data on the occupation are available after the first stage. This process, however, is quite expensive in terms of time. Depending on the complexity of the design, a single logic synthesis may require hours to be completed. As a result, the solution cannot be evaluated by simply adding the contribution of the allocated components, but the effects of the logic synthesis step has to be somehow considered. In previous works, only the area of the functional units (i.e. the resources that performs the operations) and registers were included in the solution evaluation, as they were considered much more relevant than interconnection elements (e.g., multiplexers). However, recent studies [5, 15] demonstrated that the area of the interconnection elements has by far outweighed the area of the functional units. In ASICs, this brings undesirable side-effects, like an unacceptable propagation due to the long wires determined by an inefficient components placement. In FPGAs, this situation is critical for area calculation, since a large amount of LUTs may be used to connect and wire the functional blocks. This strongly motivates the design of techniques that take into account the amount and size of interconnection elements. Not considering them could lead to an inaccurate area estimation and to a final solution that does not meet the area constraints.

Unfortunately, due to the complexity of analysis and the interdependence of the synthesis steps, all the information is available only after the complete synthesis of the design solutions. Some examples of the computational effort required to produce a complete synthesis for various designs with our HLS tool and the Xilinx ISE version 10.1 are reported in Table 26.1. We used a system with a Intel Core 2 Duo T7500 CPU (2,2 GHz, 4 MB of second level cache) and 2 GB of memory. These results clearly show that the complete synthesis cannot be efficiently included into any black-box optimization algorithm that usually performs a huge number of design evaluations.

Table 26.1 Examples of computational effort for the complete synthesis of common benchmarks for high-level synthesis

Benchmark	HLS time (s)	Logic Synthesis time (s)	Total time (s)
arf	0.35	100.72	101.07
bandpass	0.99	28.50	29.49
chemical	0.37	122.03	122.40
dct	1.02	133.70	134.72
dct wang	1.04	124.45	125.49
dist	0.96	248.54	249.50
ewf	0.39	121.35	121.74
fir	0.07	43.61	43.80
paulin	0.06	32.19	32.25
pr1	0.84	121.70	122.54
pr2	1.19	176.08	177.27
tseng	0.05	14.00	14.05
Avg.	0.61	105.57	106.18

This motivates us to investigate different solutions to reduce the execution time of the solution evaluation, limiting the impact on the quality of the final solutions. If we reduce the time required to evaluate a design solution, more alternatives could be analyzed in the same time and a larger portion of the design space can be explored.

In the following sections we introduce two different techniques to speed-up the evaluation process. In Section 26.5 we discuss how to replace the fitness computation with a cost model to avoid the expensive logic synthesis step to evaluate candidate solutions. In particular, we show how the accuracy of the cost model affect the overall performance. Then, in Section 26.6 we investigate the application of a *fitness inheritance* inheritance mechanism to reduce the number of evaluations performed without degrading the performance of the evolutionary process.

26.5 Building Cost Models

In this section, the time-consuming logic synthesis step is substituted with a model of performance and area, based on relevant features of the structural descriptions obtained by the high-level synthesis step. To compute the performance, it is necessary to count the control steps required by the design to execute all the operations, which correspond to the number of clock cycles required to execute the design. To compute the area, then, it is necessary to perform the logic synthesis of the specifications produced by the HLS flow. As described in Section 26.2, the typical approach in literature is to build a fitness surrogate that, considering some features of the design, is able to estimate its occupation. In the following, we present two possible cost models for the area: one linearly combines the number of functional units present in the design and their area and counts the memory elements, the other one is a linear regression that also takes into account interconnections.

However, even if the solution modeling allows to reduce the time required to evaluate a solution, it introduces an approximation that could affect the explorations. For this reason, the accuracy of the models and the quality of the designs obtained by the exploration using these models will be discussed and analyzed, respectively, in Section 26.5.2.1 and Section 26.5.2.2.

$$\#A.Area.FF = \sum_{R \in A.DataPath.Registers} sizeof(R) + log_2(A.FSM.NumControlStates)$$

$$\#LUT = \sum_{F \in A.DataPath.FunctionalUnits} F.Area$$

$$A.Area = \#A.Area.FF + \#LUT$$

Fig. 26.2 Simplified model to estimate area occupation for the structural design A

26.5.1 Cost Models

One of the simplest models used in HLS flows counts the number of functional units and memory elements. An area estimation in terms of LUTs for each type of functional unit (e.g. adder, subtractors, multipliers) can be easily obtained through the synthesis of such elements. The linear combinations of these values provides an initial estimation of the overall occupation of the design [21]. The HLS flow can instead estimate the number of single bit flip-flops by counting the number of registers required by the design, for both the data-path and the state encoding registers of the control-FSM. Consequently, the first estimation model we adopted to compute the area of a design is shown in Fig. 26.2. This model is easy to develop and allows a very fast estimation of the area occupation of the design. However, it can only model how the exploration affects the number of functional units or registers, and does not account for the effects of the interconnection elements. The contribution of the controller is limited to the number of memory elements required to encode the state. The logic to compute the outputs or the transition function is ignored. Such a solution was proposed, several years ago, mainly for data-intensive designs targeting ASIC technology, considering that the interconnections and the controller had a reduced impact on these designs.

Nevertheless, as discussed in Section 26.4, recent studies demonstrated that this approach is not applicable with FPGAs [5], and it is becoming inefficient also for ASICs [15]. We thus investigated more detailed models for generating the required values to verify if it is possible to obtain better approximations. We started with an already existing area model for FPGAs [3], and generalized it for several reasons. First, the vendors (e.g., Xilinx or Altera) offer tools with different approaches to translate the structural descriptions into the logic functions and to interconnect the logic blocks. Second, the devices, even if provided by the same vendor, can use different architectures (e.g., LUTs with a different number of inputs). So, we introduced a generic model and a methodology to specialize it, in order to address different vendors' tools and different devices. The final area model we used for fast estimation is shown in Figure 26.3. For each architecture A the model divides the area into two main parts: the Flip-Flop part and the LUT part. While the Flip-Flop part is easy to estimate using the same formula of the previous approach, the LUT part is a little more complex. Four main parts contribute to the global area in terms of LUT: FU, FSM, MUX and Glue. The FU part corresponds to the contribution

$$\#FF_{FSM} = \lceil \alpha_1 * log_2(A.FSM.NumControlStates) + \beta_1 \rceil$$
$$\#FF_{DataPath} = \sum_{R \in A.DataPath.Registers} \alpha_2 * sizeof(R) + \beta_2$$
$$A.Area.FF = \alpha_3 * \#FF_{FSM} + \beta_3 * \#FF_{DataPath}$$
$$\#LUT_{FSM} = \lceil \alpha_4 * A.FSM.NumControlStates + \beta_4 * A.FSM.Inputs + \gamma_4 * A.FSM.Outputs + \delta_4 \rceil$$
$$\#LUT_{FU} = \sum_{F \in A.DataPath.FunctionalUnits} \alpha_5 * F.Area + \beta_5$$
$$\#LUT_{MUX} = \sum_{M \in A.DataPath.Mux} \lceil \alpha_6 * M.Input + \beta_6 * sizeof(M) + \gamma_6 \rceil$$
$$\#LUT_{Glue} = \lceil \alpha_7 * \#LUT_{FSM} + \beta_7 * A.DataPath.NumRegisters + \gamma_7 \rceil$$
$$A.Area.LUT = \alpha_8 * \#LUT_{FSM} + \beta_8 * \#LUT_{FU} + \gamma_8 * \#LUT_{MUX} + \delta_8 * \#LUT_{Glue} + \varepsilon_8$$
$$A.Area = \alpha_8 * A.Area.LUT + \alpha_9 * A.Area.FF$$

Fig. 26.3 Linear regression model to estimate area occupation for the structural design A

of the functional units and so its value is still the sum of the area value of each functional unit. The other three parts (FSM, MUX, Glue) are obtained by using a regression-based approach:

- the FSM contribution is due to the combinational logic used to compute the output and next state;
- the MUX contribution is due to the number and size of multiplexers used in the data-path;
- the Glue contribution is due to the logic to enable writing in the flip flops and to the logic used for the interaction between the controller and the data-path.

The model is then specialized for the particular vendor's tools and devices by using a linear regression approach similar to [3], obtaining an accurate estimation of the design objectives, if properly adapted. For this reason, one of the main drawbacks is that, each time the designer changes the experimental setup, it requires an initial phase of tuning, that could be time-consuming and error-prone.

26.5.2 Experimental Evaluation

In this section, the models are validated by using the set of benchmarks presented in [7] and targeting a Virtex XC2VP30 FPGA. The logic synthesis is executed with Xilinx ISE ver. 10.1. We performed the coefficient extraction for the model based on linear regression and its validation using two data-sets, each one composed by different hardware architectures of the benchmarks. The resulting model is shown in Fig. 26.4.

The error of the two models is discussed in Section 26.5.2.1, while their impact on the final estimates of the Pareto-optimal set is analyzed in Section 26.5.2.2. In particular, we demonstrate which model is better to drive the optimization process carried on by the genetic algorithm.

$$\#FF_{FSM} = \lceil log_2(A.FSM.NumControlStates) \rceil$$
$$\#FF_{DataPath} = \sum_{R \in A.DataPath.Registers} sizeof(R)$$
$$A.Area.FF = \#FF_{FSM} + \#FF_{DataPath}$$
$$\#LUT_{FSM} = \lceil 1.99 * A.FSM.NumControlStates - 0.24 * A.FSM.Inputs + 1.50 * A.FSM.Outputs - 9.97 \rceil$$
$$\#LUT_{FU} = \sum_{F \in A.DataPath.FunctionalUnits} F.Area$$
$$\#LUT_{MUX} = \sum_{M \in A.DataPath.Mux} \lceil (0.79 * sizeof(M)) \rceil$$
$$\#LUT_{Glue} = \lceil 0.7 * \#LUT_{FSM} + 1.10 * A.DataPath.NumRegisters \rceil$$
$$A.Area.LUT = \#LUT_{FSM} + \#LUT_{FU} + \#LUT_{MUX} + \#LUT_{Glue}$$
$$A.Area = A.Area.LUT + A.Area.FF$$

Fig. 26.4 Model used to estimate area occupation for the FPGA design *A* using Xilinx ISE ver. 10.1. and targeting a Virtex XC2VP30 FPGA device

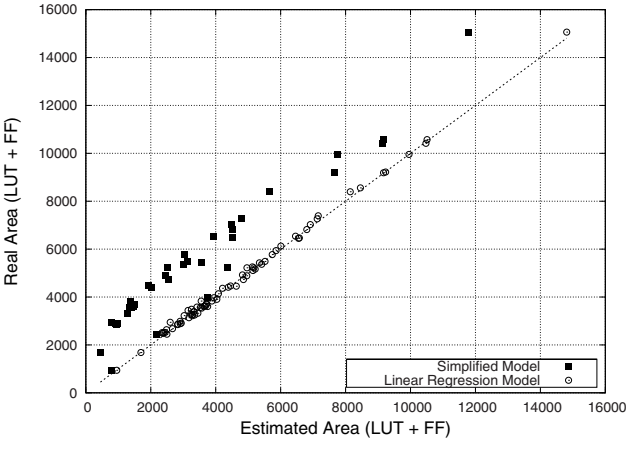


Fig. 26.5 Validation of the FPGA area models

26.5.2.1 Accuracy of Models

Figure 26.5 presents the validation for the models described in the previous section. The dashed line represents the ideal situation, where the estimated values are equal to the real ones, obtained with an actual synthesis on the target device. Squared dots represent the values associated to the first model, where only functional units and registers are considered. Round dots, instead, represent the values obtained with the linear regression model. We validated the models on a data-set composed of 73 designs that represent different architectures of the benchmarks described in [7] and shown in Table 26.1.

The plot shows that the simplified model systematically underestimates the real values. This happens because the contribution due to multiplexers and steering logic

Table 26.2 Comparison of the models applied to a set of benchmarks

Benchmark	Simplified			Linear Regression		
	Area	#Pareto Points		Area	#Pareto Points	
		NSGA-II DSE	Synthesis		NSGA-II DSE	Synthesis
arf	63,010	7	6	**60,341**	9	9
dct	111,392	8	6	**107,808**	14	14
dct wang	115,167	11	9	**110,198**	16	16
dist	158,716	14	13	**157,049**	20	20
ewf	73,969	10	9	**72,634**	13	13
pr1	72,990	9	8	**70,978**	12	12
pr2	162,130	17	14	**154,503**	19	19

is not considered. On the other hand, the model based on linear regression approximates the real values with a good accuracy. In particular, the simplified model shows an average error of 43.39±20.00%, while the maximum error is 73.35%. The model based on linear regression, instead, has an average error equal to 2.22±2.20%, with a maximum error of 11.85%. Thus, we can confirm that is able to accurately estimate all the area contributions of a structural description and that it can be effectively integrated in the proposed methodology to drive the exploration algorithm.

26.5.2.2 Performance of the Methodology

The error information is insufficient to determine which model should be preferred. We need to evaluate the effects of the adoption of the models on the resulting estimates of the Pareto-optimal set. The more accurate is the model, the better it would drive the design space exploration, resulting in a better estimate of the Pareto-optimal set. However, even a simple model might be enough to perform an effective design space exploration, if it would be able to identify and consider the most relevant features of the design.

Consequently, we performed different experiments, alternatively adopting different area models. Each experiment consists of 100 generations and involves a population of 100 candidate solutions. The results averaged over 10 runs are shown in Table 26.2, where the column *Area* measures the quality of the non-dominated set discovered. In particular, the lower is this value, the better is the outcome of the optimization processes. *NSGA-II DSE* and *Synthesis* values represent, respectively, the Pareto points coming out from the exploration algorithm and the results after their actual synthesis. The results show that the linear regression model systematically outperforms the simplified model. The reason is that the linear regression model is more accurate, and it is able to consider the effects on the solution evaluation of all the components contained in the final architecture. Furthermore, having a model that generates a more accurate fitness function results in a larger number of points in the estimate of the Pareto-optimal set.

Some interesting approximations of the Pareto-optimal curve are also graphically compared in Fig. 26.6. In Fig. 26.6(a) and Fig. 26.6(b) we see that the linear regression

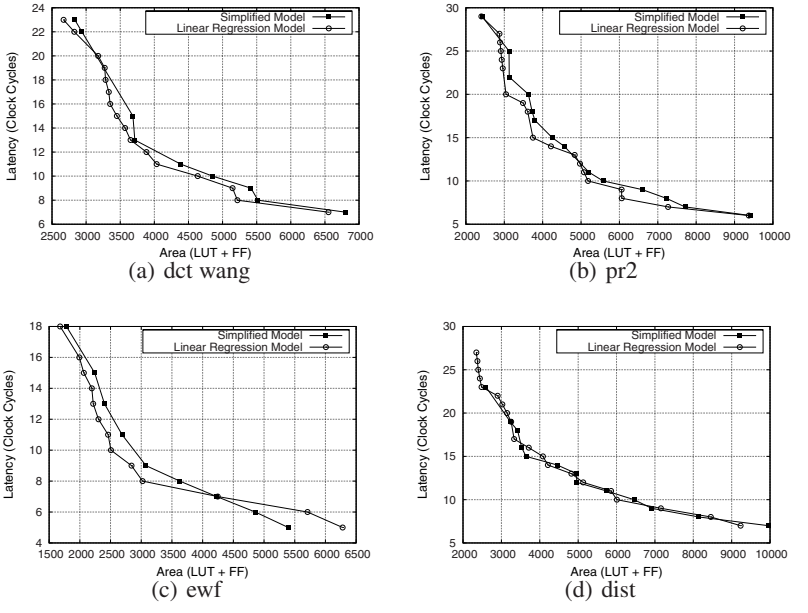


Fig. 26.6 Examples of comparison of obtained Pareto curves using the different models

model systematically outperforms the simplified one in terms of quality of the Pareto-optimal set. In Fig. 26.6(c), for large designs, the model that considers only functional units and registers obtains better results. In fact, in this region of the design space, the impact of the multiplexer is limited (about 15-20%) and a fitness function focused only on functional components and registers is more suitable to drive the exploration algorithm. In the other region of the space, where few functional units are in the designs, the multiplexers have a larger impact (about 70-75%) and the fitness function that takes into account their occupation obtains better results. Finally, in Fig. 26.6(d), the multiplexers are not so relevant for the design. As a result the two models are almost equivalent, as also shown by the similar values in Table 26.2.

26.6 Fitness Inheritance

Table 26.1 shows that also the HLS step is computationally intensive, even if much less than the logic synthesis one. HLS impact becomes bigger as the dimension of the problem grows. We thus expect that, when applied to larger problems, it could become another significant bottleneck of the methodology.

For this reason, in this section we exploit *fitness inheritance* to substitute all the steps of the complete synthesis by interpolating the fitness of previously evaluated individuals. *Fitness inheritance* is a technique totally orthogonal to the solution proposed in the previous section. The individuals used for fitness inheritance can, in fact, be evaluated with any approach (e.g., actual synthesis or modeling with one of the proposed models). The key idea is that, with this approach, we try to limit the

overall number evaluations rather than reducing the time required for a single evaluation (i.e. the synthesis steps, HLS or logic synthesis). Interpolation is usually much less time consuming, thus we can save some of the time required for a complete synthesis.

Note that this technique is less dependent on the problem than solution modeling. In fact, to build the model, the designer should identify the relevant features of the design solutions, synthesize the related hardware descriptions and establish a correspondence. On the contrary, fitness inheritance is only based on the definition of the chromosome encoding and the fitness of previously evaluated individuals.

However, to produce an effective surrogate, we needed to carefully take into account some aspects. In particular, we focused our attention on the percentage of individuals to be estimated, on the parents to choose and on how to combine their fitness. We present and discuss these aspects in Section 26.6.1, and then compare the quality of some different solutions in Section 26.6.2.

Provided a proper analysis of these aspects, the results show that fitness inheritance is able to consistently reduce the execution time of all the methodology. We also demonstrate that, if the parameters are not correct, the method can even degrade rather than improving the performance of the exploration algorithm.

26.6.1 Inheritance Model

In the proposed approach, only in the first, initial population the fitness of all the individuals is evaluated. In the subsequent populations, only the fitness of a portion of the population is evaluated, while the remaining ones inherit the fitness through interpolation of the values already computed. In particular, the fitness of individual Ind_i is inherited with probability p_i. To compute the fitness estimation for Ind_i, we need to calculate the distance between it and all the individuals that can be effectively used for the estimations. The estimation can be based on the ancestors, i.e., all the individuals that have been effectively evaluated starting from the first generation, or on the parents, i.e., all the individuals that have been effectively evaluated only in the latest generation. In both the cases, we will call this set S in the rest of the section. The fitness value of Ind_i is thus estimated as follows. The chromosome of Ind_i is mapped onto a binary vector of size N, where each variable of the vector is uniquely related to a gene of the chromosome. The vector is instantiated by the following delta function:

$$\delta_{i,j}^{k} = \begin{cases} 0 \; if \; Ind_i[k] = Ind_j[k] \\ 1 \; otherwise \end{cases} \tag{26.2}$$

where $Ind_i[k]$ is the value associated to the *k-th* gene of the individual Ind_i. After the delta function has been computed for all the N genes of the chromosome, the distance $d_{i,j}$ between individual Ind_i and individual Ind_j is calculated as follows:

$$d_{i,j} = \frac{\sum_{k=1}^{N} \delta_{i,j}^{k}}{N} \tag{26.3}$$

this function is normalized with the size of chromosome, so its value is always between 0 and 1. The distance $d_{i,j}$ measures the similarity of two individuals. If these are totally different (there is not any matching gene), the value will be 1. On the other hand, if the two individuals are identical, the value will be 0. Only individuals that are considered *neighbors* in this space will be kept for the fitness estimation. We call r the maximum distance that an individual should have to be kept. The name r is used to remember the term *radius*, since the region delimited by this value can be imagined as a N-dimensional hyper-sphere centered at individual Ind_i. All the individuals $Ind_j \in S$, having distance smaller than the radius r, can be considered as points inside this hyper-sphere. Therefore, all these individuals will be considered for estimation and the distance value is modified as follows:

$$d'_{i,j} = \begin{cases} d_{i,j} \ if \ d_{i,j} \leq r \\ 1 \quad if \ d_{i,j} > r \end{cases} \tag{26.4}$$

where all individuals outside the hyper-sphere are equivalent to points at infinite distance and they will not be considered for estimation. To perform the estimation, we require a minimum number of points in this region. If there are not enough points, it means that there is no sufficient local information to estimate and individual. So, it will be really evaluated. If there are enough points, instead, the estimation can be performed on the set S' of points, selected as follows:

$$Fit_i^z = \frac{\sum_{j \in S'} f(Fit_j^z) * g(1 - d'_{i,j})}{\sum_{j \in S'} g(1 - d'_{i,j})} \tag{26.5}$$

for each objective z. Fit_k^z is the value of the objective z for the individual Ind_k and $(1 - d'_{i,j})$ is used as a measure of closeness between individuals. f and g are functions that change the contribution of the two terms. We formulated the term $(1 - d'_{i,j})$ in this way since the distance $d'_{i,j}$ does not go to *infinite*, but has a value between 0 and 1. Therefore, we consider the values associated to 1 equivalent to an *infinite* distance (i.e., no contribute to the fitness). As explained above, this weighted average is computed for all the objectives considered in the optimization. The resulting value is then returned to the genetic algorithm, which can so proceed. A flag is also associated to the individual Ind_i to remember that the fitness has been estimated and not really evaluated. This allows the algorithm to identify the estimated individuals when needed.

In particular, in the last generation the fitness of all the individuals are tested for evaluation. Individuals that have already been evaluated will be skipped, while the estimated individuals will be effectively evaluated. Thus, when the exploration ends, all the individuals on which the final non-dominated set is computed will have a real fitness value associated.

26.6.2 Experimental Evaluation

In this section, we evaluate different aspects related to fitness inheritance and compare several parameter settings. The parameters for the GA are the same used in

Table 26.3 Comparison of the weighting functions about fitness evaluations and execution time

Benchmarks	Model	w/o inheritance #Eval.	w/o inheritance Exec. Time(s)	Ancestors #Eval.	Ancestors Exec. Time (s)	Ancestors diff (%)	Parents #Eval.	Parents Exec. Time(s)	Parents diff (%)
arf	linear	9,639	1,064.65	4,721	1,211.68	+13.81%	7,567	888.66	-16.53%
	quadratic			5,048	1,327.19	+24.65%	8,160	801.86	-24.68%
	exponential			5,330	1,374.83	+29.13%	9,027	985.59	-7.42%
dct	linear	11,150	3,677.55	7,263	5,074.26	+37.98%	7,758	2,699.86	-26.58%
	quadratic			7,758	5,492.04	+49.34%	7,131	2,360.44	-35.81%
	exponential			7,319	5,135.87	+39.65%	7,867	2,625.69	-28.60%
dct wang	linear	10,837	3,470.16	6,945	4,906.20	+41.38%	7,348	2,385.12	-37.27%
	quadratic			7,479	5,456.37	+57,24%	7,758	2,549.29	-26.53%
	exponential			6,160	4,083.51	+17,68%	7,312	2,689.09	-22.51%
dist	linear	12,683	3,907.81	8,315	6,181.55	+58.18%	7,812	3,402.46	-12.93%
	quadratic			7,607	5,254.36	+34.46%	7,358	3,048.29	-21.99%
	exponential			8,376	5,995.74	+53.43%	7,801	3,590.28	-8.13%
ewf	linear	9,575	1,165.55	6,218	2,074.99	+78.03%	6,518	814.60	-30.11%
	quadratic			6,392	2,127.26	+82.51%	64,18	790.00	-32.22%
	exponential			6,256	2,095.66	+79.80%	6,578	889.94	-23.65%
pr1	linear	9,773	2,542.35	8,358	5,514.68	+116.91%	7,548	2,058.11	-19.05%
	quadratic			6,834	3,879.37	+52.59%	6,912	1,878.11	-26.13%
	exponential			6,681	3,594.25	+41.48%	7,154	1,958.15	-22.98%
pr2	linear	10,610	4,044.71	6,423	4,718.46	+16.66%	6,958	3,589.25	-11.26%
	quadratic			6,930	5,086.70	+25.76%	7,198	3,578.02	-11.54%
	exponential			6,937	5,119.57	+26.57%	7,277	3,547.66	-12.59%
					Avg.	+46.53%		Avg.	-21.82%
					Std. Dev.	25.90%		Std. Dev.	8.81%

Section 26.5.2.2. In all the experiment, the fitness evaluation uses the linear regression model. In Section 26.6.2.1 we present, discuss, and compare different functions to weight the fitness contributions of the evaluated individuals. In Section 26.6.2.2 we apply fitness inheritance both to the ancestors and to the parents and compare the results. Finally, we analyze the effects of different inheritance percentages (p_i) and distance rates (r).

26.6.2.1 Weighting Functions

We considered three weighting functions (i.e., function g in Eq.26.5) for inheritance: linear, quadratic and exponential. The first model is computed as follows:

$$Fit_i^z = \frac{\sum_{j \in S'} Fit_j^z * (1 - d'_{i,j})}{\sum_{j \in S'} (1 - d'_{i,j})} \tag{26.6}$$

where the fitness of the evaluated individuals are linearly combined with the related distances $1 - d'_{i,j}$ from the candidate individual Ind_i. While, the second model is computed as follows:

$$Fit_i^z = \frac{\sum_{j \in S'} Fit_j^z * (1 - d'_{i,j})^2}{\sum_{j \in S'} (1 - d'_{i,j})^2} \tag{26.7}$$

where the quadratic function in $(1 - d'_{i,j})$ is used to increase the weight of distance, similarly to the Physics equations for gravity or magnetism. However, we adopt a

Table 26.4 Comparison of the weighting functions about quality of the results

Benchmarks	Model	w/o inheritance Area	w/o inheritance #Pareto	Ancestors Area	Ancestors #Pareto DSE	Ancestors #Pareto Synth.	Parents Area	Parents #Pareto DSE	Parents #Pareto Synth.
arf	*linear*	63,157	9	63,756	9	9	63,633	11	11
	quadratic			63,633	11	11	**62,729**	12	12
	exponential			65,097	9	9	64,941	12	12
dct	*linear*	113,526	14	113,151	14	14	113,732	12	12
	quadratic			**111,598**	11	10	113,732	12	12
	exponential			114,550	13	11	113,469	10	10
dct wang	*linear*	112,868	16	113,351	12	12	112,782	15	15
	quadratic			114,778	17	17	**112,484**	14	14
	exponential			114,389	13	13	113,911	14	14
dist	*linear*	169,487	20	168,955	18	17	170,731	19	19
	quadratic			171,706	18	17	170,578	18	18
	exponential			169,708	21	20	**167,900**	19	19
ewf	*linear*	72,634	13	76,946	14	12	**73,245**	13	13
	quadratic			75,503	13	12	74,366	11	11
	exponential			77,000	13	13	74,184	13	13
pr1	*linear*	75,405	12	76,286	11	10	75,168	11	11
	quadratic			76,878	11	10	75,083	11	11
	exponential			76,580	12	11	**75,000**	12	12
pr2	linear	156,906	19	158,110	19	18	154,903	18	18
	quadratic			161,812	21	19	**154,186**	19	19
	exponential			162,195	22	20	160,800	20	20

proportion with $(1-d)^2$ and not $(1/d)^2$, that allows dealing with *infinite* distance as described above. The last model is computed as:

$$Fit_i^z = \frac{\sum_{j \in S'} Fit_j^z * (e^{1-d'_{i,j}} - 1)}{\sum_{j \in S'} (e^{1-d'_{i,j}} - 1)} \tag{26.8}$$

where the distance is exponentially weighted, emphasizing even more the contribution of the nearest individuals to the fitness estimation of Ind_i. These functions have been applied both to the ancestors and to the parents. The distance rate has been set to $r = 0.20$ and the inheritance rate to $p_i = 0.5$. In the former case, the set S of individuals considered increases generation by generation, while, in the latter case, the size is constant and related to the size of the population. When the ancestors are used, the inheritance model analyzes all the elements of the set for distance calculation, and the time required for fitness inheritance could overcome the time required by the function evaluation itself. Thus, in this case, fitness inheritance reduces the number of evaluations, but may also degrade the overall execution time of the methodology. At opposite, if the methodology is applied only to the parents, both the number of evaluations and the execution time of the methodology are significantly reduced. Since less individuals are available for computing the inheritance information (see Eq. 26.4), the number of evaluations is larger than with the ancestors. Table 26.3 shows the data about the number of evaluations and about the overall execution time.

Table 26.5 Comparison of different inheritance rate about quality of the exploration, fitness evaluations and execution time

Benchmarks		w/o inheritance				w inheritance					
		Area	#Pareto	#Eval.	Exec. Time(s)	Area	#Pareto DSE	#Pareto Synth.	#Eval.	Exec. Time(s)	diff (%)
	p_i										
	0.20					64,088	12	12	**7,681**	**834.00**	-21.66%
	0.30	63,157	9	9,639	1,064.65	**62,724**	11	11	8,852	961.30	-9.71%
	0.40					62,990	11	11	8,812	981.43	-7.82%
arf	0.50					63,239	11	11	8,259	913.65	-14.18%
	0.55					62,820	10	10	8,160	888.66	-16.53%
	0.60					64,275	12	12	8,702	965.97	-9.27%
	0.70					63,654	11	11	8,655	961.38	-9.70%
	0.20					113,487	14	14	8,158	3,248.20	-11.67%
	0.30	113,526	14	11,150	3,677.55	113,909	16	16	7,789	2,874.11	-21.85%
	0.40					113,104	15	15	7,441	2,581.47	-29.80%
dct	0.50					113,732	12	12	7,131	2,360.44	-35.81%
	0.55					112,223	11	11	**7,062**	**2,236.98**	-39.17%
	0.60					114,051	16	16	7,325	2,514.02	-31.64%
	0.70					**111,080**	12	12	7,587	2,636.42	-28.31%
	0.20					113,952	13	13	8,258	3,025.20	-12.82%
	0.30	112,868	16	10,837	3,470.16	**111,487**	14	14	8,126	2,854.01	-17.76%
	0.40					113,536	15	15	7,887	2,741.36	-21.00%
dct wang	0.50					112,484	14	14	7,758	**2,549.29**	-26.54%
	0.55					114,283	15	15	**7,747**	2,569.10	-25.97%
	0.60					112,842	17	17	7,854	2,698.47	-22.24%
	0.70					113,706	16	16	7,981	2,747.11	-20.84%
	0.20					166,060	17	17	7,414	3,658.10	-6.39%
	0.30	169,487	20	12,683	3,907.81	**161,432**	17	17	7,401	3,698.43	-5.36%
	0.40					167,804	18	18	**7,333**	3,154.01	-19.29%
dist	0.50					170,578	18	18	7,358	**3,048.29**	-21.99%
	0.55					167,801	17	17	7,441	3,341.22	-14.50%
	0.60					167,472	19	19	7,551	3,418.99	-12.51%
	0.70					170,151	16	16	7,547	3,507.67	-10.24%
	0.20					74,623	13	13	9,147	847.10	-27.32%
	0.30	**72,634**	13	9,575	1,165.55	74,143	12	12	7,765	858.36	-26.36%
	0.40					73,609	13	13	7,010	802.19	-31.17%
ewf	0.50					74,366	11	11	6,418	790.00	-32.22%
	0.55					74,053	11	11	**6,211**	**767.41**	-34.16%
	0.60					73,234	12	12	6,478	789.23	-32.29%
	0.70					73,023	13	13	6,441	796.59	-31.66%
	0.20					75,308	13	13	8,012	2,236.39	-12.03%
	0.30	75,405	12	9,773	2,542.35	**74,309**	9	9	7,477	2,056.47	-19.11%
	0.40					75,319	10	10	7,087	1,969.78	-22.52%
pr1	0.50					75,083	11	11	6,912	1,878.11	-26.13%
	0.55					74,888	9	9	**6,898**	**1,789.56**	-29.61%
	0.60					75,045	10	10	7,101	1,941.02	-23.65%
	0.70					74,831	10	10	7,011	1,867.53	-26.54%
	0.20					160,628	20	20	8,101	3,856.12	-4.66%
	0.30	156,906	19	10,610	4,044.71	**150,630**	22	22	7,812	3,785.54	-6.41%
	0.40					158,903	21	21	7,485	3,696.36	-8.61%
pr2	0.50					154,186	19	19	7,198	3,578.02	-11.54%
	0.55					154,241	15	15	**7,012**	3,547.10	-12.30%
	0.60					155,714	21	21	7,025	3,423.11	-15.37%
	0.70					159,150	21	21	6,894	**3,326.98**	-17.74%
										Avg.	-20.43%
										Std. Dev.	9.13%

Finally, Table 26.4 compares the quality of the results with the different weighting functions. As in Section 26.5.2.2, the area delimited by the approximated Pareto-optimal curve gives a qualitative evaluation of the explorations. The results show that the quadratic function is the most efficient solution to weight the fitness

Table 26.6 Comparison of different distance rate about quality of the results

Benchmarks	r	w/o inheritance Area	w/o inheritance #Pareto	w inheritance Area	w inheritance #Pareto DSE	w inheritance #Pareto Synth.
arf	0.10	63,157	9	63,549	10	10
	0.20			63,626	11	11
	0.25			63,307	11	11
	0.50			**62,906**	11	11
dct	0.10	113,526	14	**112,626**	14	14
	0.20			113,116	13	13
	0.25			112,630	17	17
	0.50			113,234	14	14
dct wang	0.10	112,868	16	113,347	13	13
	0.20			115,027	15	15
	0.25			**111,391**	17	17
	0.50			112,979	13	13
dist	0.10	169,487	20	171,159	19	19
	0.20			168,305	16	16
	0.25			**168,195**	17	17
	0.50			169,162	19	19
ewf	0.10	72,634	13	73,156	14	14
	0.20			73,302	12	12
	0.25			**71,693**	11	11
	0.50			75,009	13	13
pr1	0.10	75,405	12	76,607	13	13
	0.20			**75,372**	11	11
	0.25			76,214	12	12
	0.50			77,654	12	12
pr2	0.10	156,906	19	157,005	19	19
	0.20			154,814	18	18
	0.25			158,718	21	21
	0.50			**153,866**	18	18

contributions. In fact, this function emphasizes the individuals closer to the candidate more than the linear function. With respect to the exponential function, which (strongly) emphasize only very similar individuals, it also consider more distant contributions (always inside the radius).

26.6.2.2 Parameter Analysis

In this section, different inheritance rates (p_i) and different distance rates (r) are studied. The parameters for the GA are the same used in Section 26.5.2.2. The fitness evaluation uses the linear regression model and exploits inheritance on *parents* with the *quadratic* weighting function in all the experiments.

Table 26.5 shows the results of explorations where fitness inheritance is applied with different inheritance rates. Note that values of p_i between 0.40 and 0.55 provides a good trade-off between the quality of the exploration and the related execution time. The reason is that, with lower values, few individuals are chosen for inheritance. On the other hand, with higher values, the number of really evaluated individuals is limited. When there are not enough similar individuals (at least 10), we swap the fitness evaluation to the HLS flow and the area model. Therefore, the execution time is not reduced as expected. The results obtained in our experiments

are also consistent with the optimal proportion for inheritance derived in [32], defined as follows:

$$0.54 \leq p_i* \leq 0.558 \tag{26.9}$$

Finally, Table 26.6 reports the results obtained with $p_i = 0.5$ while changing the distance rates r. Almost all the considered rates give good results. However, values comprised between 0.20 and 0.25 perform best. In fact, with lower values, limited information is available for inheritance, while, with higher values, additional noise is introduced in the interpolation.

26.7 Conclusions

In this work, we presented an evolutionary approach to HLS design space exploration problem based on NSGA-II, a multi-objective evolutionary algorithm. We exploited two orthogonal techniques, surrogate fitness and fitness inheritance, to reduce the time necessary to the expensive solution evaluations. The fitness surrogate was computed with a linear regression model that takes into account the contributions of all the components of the design (e.g., interconnections or glue logic) and the effect of the optimizations introduced by the logic tool: replacing the logic synthesis process with such a surrogate model, we can save a lot of computational time. Fitness inheritance was used to reduce the number of evaluations, by evaluating only a fixed portion of the population. We validated our approach on several benchmarks and our results suggest that both the proposed techniques allows to speed-up the evolutionary search without degrading its performance. At the best of our knowledge, this is the first framework for the HLS design space exploration that exploits at the same time a surrogate fitness model as well as a fitness inheritance scheme.

References

1. Araújo, S.G., Mesquita, A.C., Pedroza, A.: Optimized Datapath Design by Evolutionary Computation. In: IWSOC: International Workshop on System-on-Chip for Real-Time Applications, pp. 6–9 (2003)
2. Barthelemy, J.F.M., Haftka, R.T.: Approximation concepts for optimum structural design - a review. Structural Optimization (5), 129–144 (1993)
3. Brandolese, C., Fornaciari, W., Salice, F.: An area estimation methodology for FPGA based designs at SystemC-level. In: DAC: Design Automation Conference, pp. 129–132. ACM, New York (2004)
4. Chaiyakul, V., Wu, A.C.H., Gajski, D.D.: Timing models for high-level synthesis. In: EURO-DAC 1992: European Design Automation Conference, pp. 60–65. IEEE Computer Society Press, Los Alamitos (1992)
5. Chen, D., Cong, J.: Register binding and port assignment for multiplexer optimization. In: ASP-DAC: Asia South Pacific Design Automation Conference, pp. 68–73 (2004)
6. Chen, J.H., Goldberg, D.E., Ho, S.Y., Sastry, K.: Fitness inheritance in multi-objective optimization. In: GECCO: Genetic and Evolutionary Computation Conference, pp. 319–326 (2002)

7. Cordone, R., Ferrandi, F., Santambrogio, M.D., Palermo, G., Sciuto, D.: Using speculative computation and parallelizing techniques to improve scheduling of control based designs. In: ASPDAC: Asia South Pacific Design Automation Conference, pp. 898–904. ACM, Yokohama (2006)
8. De Micheli, G.: Synthesis and Optimization of Digital Circuits. McGraw-Hill, New York (1994)
9. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
10. Dennis, J., Torczon, V.: Managing approximate models in optimization. In: Alexandrov, N., Hussani, M. (eds.) Multidisciplinary design optimization: State-of-the-art, pp. 330–347. SIAM, Philadelphia (1997)
11. Ducheyne, E., Baets, B.D., Wulf, R.D.: Is fitness inheritance useful for real-world applications? (2003)
12. Ferrandi, F., Lanzi, P.L., Palermo, G., Pilato, C., Sciuto, D., Tumeo, A.: An evolutionary approach to area-time optimization of FPGA designs. In: ICSAMOS: International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, pp. 145–152 (2007)
13. Grefenstette, J.J., Fitzpatrick, J.M.: Genetic search with approximate function evaluation. In: International Conference on Genetic Algorithms, pp. 112–120. Lawrence Erlbaum Associates, Inc., Mahwah (1985)
14. Grewal, G., O'Cleirigh, M., Wineberg, M.: An evolutionary approach to behavioural-level synthesis. In: CEC: IEEE Congress on Evolutionary Computation, 8-12, pp. 264–272. ACM Press, New York (2003)
15. Gu, Z., Wang, J., Dick, R.P., Zhou, H.: Unified incremental physicallevel and high-level synthesis. IEEE Trans. on CAD of Integrated Circuits and Systems 26(9), 1576–1588 (2007)
16. Harik, G.: Linkage Learning via ProbabilisticModeling in the ECGA (1999)
17. Huband, S., Hingston, P.: An evolution strategy with probabilistic mutation for multi-objective optimisation. In: IEEE Congress on Evolutionary Computation, CEC 2003, pp. 2284–2291. IEEE Press, Piscataway (2003)
18. Hwang, C.T., Leea, J.H., Hsu, Y.C.: A formal approach to the scheduling problem in high level synthesis. IEEE Trans. on CAD of Integrated Circuits and Systems 10(4), 464–475 (1991)
19. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Comput. 9(1), 3–12 (2005)
20. Kollig, P., Al-Hashimi, B.: Simultaneous scheduling, allocation and binding in high level synthesis. Electronics Letters 33(18), 1516–1518 (1997)
21. Krishnan, V., Katkoori, S.: A genetic algorithm for the design space exploration of datapaths during high-level synthesis. IEEE Trans. Evolutionary Computation 10(3), 213–229 (2006)
22. Kuehlmann, A., Bergamaschi, R.A.: Timing analysis in high-level synthesis. In: ICCAD: International Conference on Computer-Aided Design, pp. 349–354. IEEE Computer Society Press, Los Alamitos (1992)
23. Llor'a, X., Sastry, K., Goldberg, D.E., Gupta, A., Lakshmi, L.: Combating user fatigue in iGAs: partial ordering, support vector machines, and synthetic fitness. In: GECCO: Conference on Genetic and evolutionary computation, pp. 1363–1370. ACM Press, New York (2005)

24. Mandal, C., Chakrabarti, P.P., Ghose, S.: Design space exploration for data path synthesis. In: International Conf. on VLSI Design, pp. 166–170 (1996)
25. Mandal, C., Chakrabarti, P.P., Ghose, S.: GABIND: a GA approach to allocation and binding for the high-level synthesis of data paths. IEEE Transaction on Very Large Scale Integration System 8(6), 747–750 (2000)
26. Meribout, M., Motomura, M.: Efficient metrics and high-level synthesis for dynamically reconfigurable logic. IEEE Trans. Very Large Scale Integr. Syst. 12(6), 603–621 (2004)
27. Palesi, M., Givargis, T.: Multi-objective design space exploration using genetic algorithms. In: CODES: International Symposium on Hardware/ software Codesign, pp. 67–72. ACM, New York (2002)
28. Paulin, P.G., Knight, J.P.: Force-directed scheduling for the behavioral synthesis of ASICs. IEEE Trans. on CAD of Integrated Circuits and Systems 8(6), 661–679 (1989)
29. Pilato, C., Palermo, G., Tumeo, A., Ferrandi, F., Sciuto, D., Lanzi, P.L.: Fitness inheritance in evolutionary and multi-objective high-level synthesis. In: IEEE Congress on Evolutionary Computation, pp. 3459–3466 (2007)
30. Reyes-Sierra, M., Coello, C.: A study of fitness inheritance and approximation techniques for multi-objective particle swarm optimization 1, 65–72 (2005)
31. Sastry, K.: Evaluation-relaxation schemes for genetic and evolutionary algorithms. Master's thesis, General Engineering Department, University of Illinois at Urbana-Champaign, Urbana, IL (2001)
32. Sastry, K., Goldberg, D.E., Pelikan, M.: Don't evaluate, inherit. In: GECCO: Genetic and Evolutionary Computation Conference, pp. 551–558. Morgan Kaufmann, San Francisco (2001)
33. Sastry, K., Lima, C.F., Goldberg, D.E.: Evaluation relaxation using substructural information and linear estimation. In: GECCO 2006, pp. 419–426. ACM, Seattle (2006)
34. Smith, R.E., Dike, B.A., Stegmann, S.A.: Fitness inheritance in genetic algorithms. In: SAC: Symposium on Applied computing, pp. 345–350. ACM Press, New York (1995)
35. Stok, L.: Data Path Synthesis. Integration, the VLSI Journal 18(1), 1–71 (1994)
36. Teich, J., Blickle, T., Thiele, L.: An evolutionary approach to system level synthesis. In: CODES Workshop, p. 167 (1997)
37. Wanner, E., Guimaraes, F., Takahashi, R., Fleming, P.: A quadratic approximation-based local search procedure for multiobjective genetic algorithms. In: IEEE Congress on Evolutionary Computation, CEC 2006, pp. 938–945 (2006), doi:10.1109/CEC.2006.1688411
38. Zitzler, E., Optimization, M., Zrich, E.H., Thiele, L., Deb, K.: Evolutionary algorithms for multiobjective optimization: Methods and applications. PhD thesis (1999)
39. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8(2), 173–195 (2000)

Index