Studies in Computational Intelligence 171

Chi-Keong Goh
Yew-Soon Ong
Kay Chen Tan (Eds.)

Multi-Objective Memetic Algorithms

Springer

Studies in Computational Intelligence, Volume 171

Editor-in-Chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

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

Vol. 150. Roger Lee (Ed.)
Software Engineering Research, Management and Applications, 2008
ISBN 978-3-540-70774-5

Vol. 151. Tomasz G. Smolinski, Mariofanna G. Milanova and Aboul-Ella Hassanien (Eds.)
Computational Intelligence in Biomedicine and Bioinformatics, 2008
ISBN 978-3-540-70776-9

Vol. 152. Jarosław Stepaniuk
Rough – Granular Computing in Knowledge Discovery and Data Mining, 2008
ISBN 978-3-540-70800-1

Vol. 153. Carlos Cotta and Jano van Hemert (Eds.)
Recent Advances in Evolutionary Computation for Combinatorial Optimization, 2008
ISBN 978-3-540-70806-3

Vol. 154. Oscar Castillo, Patricia Melin, Janusz Kacprzyk and Witold Pedrycz (Eds.)
Soft Computing for Hybrid Intelligent Systems, 2008
ISBN 978-3-540-70811-7

Vol. 155. Hamid R. Tizhoosh and M. Ventresca (Eds.)
Oppositional Concepts in Computational Intelligence, 2008
ISBN 978-3-540-70826-1

Vol. 156. Dawn E. Holmes and Lakhmi C. Jain (Eds.)
Innovations in Bayesian Networks, 2008
ISBN 978-3-540-85065-6

Vol. 157. Ying-ping Chen and Meng-Hiot Lim (Eds.)
Linkage in Evolutionary Computation, 2008
ISBN 978-3-540-85067-0

Vol. 158. Marina Gavrilova (Ed.)
Generalized Voronoi Diagram: A Geometry-Based Approach to Computational Intelligence, 2009
ISBN 978-3-540-85125-7

Vol. 159. Dimitri Plemenos and Georgios Miaoulis (Eds.)
Artificial Intelligence Techniques for Computer Graphics, 2009
ISBN 978-3-540-85127-1

Vol. 160. P. Rajasekaran and Vasantha Kalyani David
Pattern Recognition using Neural and Functional Networks, 2009
ISBN 978-3-540-85129-5

Vol. 161. Francisco Baptista Pereira and Jorge Tavares (Eds.)
Bio-inspired Algorithms for the Vehicle Routing Problem, 2009
ISBN 978-3-540-85151-6

Vol. 162. Costin Badica, Giuseppe Mangioni, Vincenza Carchiolo and Dumitru Dan Burdescu (Eds.)
Intelligent Distributed Computing, Systems and Applications, 2008
ISBN 978-3-540-85256-8

Vol. 163. Pawel Delimata, Mikhail Ju. Moshkov, Andrzej Skowron and Zbigniew Suraj
Inhibitory Rules in Data Analysis, 2009
ISBN 978-3-540-85637-5

Vol. 164. Nadia Nedjah, Luiza de Macedo Mourelle, Janusz Kacprzyk, Felipe M.G. França and Alberto Ferreira de Souza (Eds.)
Intelligent Text Categorization and Clustering, 2009
ISBN 978-3-540-85643-6

Vol. 165. Djamel A. Zighed, Shusaku Tsumoto, Zbigniew W. Ras and Hakim Hacid (Eds.)
Mining Complex Data, 2009
ISBN 978-3-540-88066-0

Vol. 166. Constantinos Koutsojannis and Spiros Sirmakessis (Eds.)
Tools and Applications with Artificial Intelligence, 2009
ISBN 978-3-540-88068-4

Vol. 167. Ngoc Thanh Nguyen and Lakhmi C. Jain (Eds.)
Intelligent Agents in the Evolution of Web and Applications, 2009
ISBN 978-3-540-88070-7

Vol. 168. Andreas Tolk and Lakhmi C. Jain (Eds.)
Complex Systems in Knowledge-based Environments: Theory, Models and Applications, 2009
ISBN 978-3-540-88074-5

Vol. 169. Nadia Nedjah, Luiza de Macedo Mourelle and Janusz Kacprzyk (Eds.)
Innovative Applications in Data Mining, 2009
ISBN 978-3-540-88044-8

Vol. 170. Lakhmi C. Jain and Ngoc Thanh Nguyen (Eds.)
Knowledge Processing and Decision Making in Agent-Based Systems, 2009
ISBN 978-3-540-88048-6

Vol. 171. Chi-Keong Goh, Yew-Soon Ong and Kay Chen Tan (Eds.)
Multi-Objective Memetic Algorithms, 2009
ISBN 978-3-540-88050-9

Chi-Keong Goh
Yew-Soon Ong
Kay Chen Tan
(Eds.)

Multi-Objective Memetic Algorithms

Springer

Dr. Chi-Keong Goh
Department of Electrical and Computer Engineering
National University of Singapore
4 Engineering Drive 3
Singapore 117576
Email: ckgoh@nus.edu.sg

Dr. Kay Chen Tan
Department of Electrical and Computer Engineering
National University of Singapore
4 Engineering Drive 3
Singapore 117576
Email: eletankc@nus.edu.sg

Dr. Yew-Soon Ong
School of Computer Engineering
Nanyang Technological University
Block N4, 2b-39, Nanyang Avenue
Singapore 639798
Email: asysong@ntu.edu.sg

ISBN 978-3-540-88050-9 e-ISBN 978-3-540-88051-6

DOI 10.1007/978-3-540-88051-6

Studies in Computational Intelligence ISSN 1860949X

Library of Congress Control Number: 2008935504

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

Printed in acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

To our family

Preface

This book presents a very first comprehensive collection of works on multi-objective Memetic algorithm (MOMAs). The application of sophisticated evolutionary computing approaches for solving complex problems with multiple conflicting objectives in science and engineering have increased steadily in the recent years. Within this growing trend, Memetic algorithms (MAs) are, perhaps, one of the most successful stories, having demonstrated better efficacy in dealing with multi-objective (MO) problems as compared to its conventional counterparts. MAs are population-based metaheuristic search methods that are inspired by Darwinian principles of natural selection and Dawkins notion of a meme defined as a unit of cultural evolution that is capable of local refinements. In diverse contexts, memetic algorithms have also been used under the name of hybrid evolutionary algorithms, Baldwinian evolutionary algorithms, Lamarkian evolutionary algorithms, or genetic local search.

There is a large volume of works on the application of MAs to real-world problems- a fact that is reflected by the number of special sessions and issues in upcoming conferences and journals. Nonetheless, researchers are only beginning to realize the vast potential of multi-objective MAs and there remain many open topics in its design. This edited book represents the first endeavor to reflect the most recent advances in the field, and to increase the awareness of the computing community at large on this effective technology for MO problems. This edition consists of invited papers written by leading researchers in the field to demonstrate the current state-of-the-art in the theory and practice of MOMAs. The book is organized for a wide readership and can be read by engineers, researchers, senior undergraduates and graduate students who are interested in the field of MAs and MO optimization. The assumed background for the book is some basic knowledge of evolutionary computation.

This book is divided into four parts. Part I contains of two chapters, providing readers with an insight to the challenges of MO optimization and the implementation issues of MOMAs. The opening chapter by Gideon on "Evolutionary Multi-Multi-Objective Optimization - EMMOO" examines the optimization of different MO problems, which are coupled by common components.

This necessitates the consideration of multiple MO problems simultaneously to discover satisfactory designs. The next chapter "Implementation of Multiobjective Memetic Algorithms for Combinatorial Optimization Problems: A Knapsack Problem Case Study" by Ishibuchi et al discuss the various implementation issues in MOMAs for combinatorial optimization problems. Extensive studies are made to examine the impact of several factors such as the frequency of local search, the choice of initial solutions for local search and the handling of infeasible solutions.

The classification of subsequent chapters into three parts is based on how information and knowledge of the problem is utilized or exploited in MOMA.

- Knowledge infused in design of problem-specific operators.
- Knowledge propagation through cultural evolution.
- Information exploited for local improvement.

Part II considers how knowledge can be infused into design of problem-specific operators and algorithms. In the first chapter "Solving Time-Tabling Problems using Evolutionary Algorithms and Heuristics Search" by Srinivasan and Zhang, the evolutionary algorithm is combined with heuristics, which ensures that all constraints are satisfied, to solve the real-world time-tabling problem of the Electrical and Computer Engineering Department in the National University of Singapore. The next chapter "An Efficient Genetic Algorithm with Uniform Crossover for the Multi-Objective Airport Gate Assignment Problem" by Hu and Di Paolo, presents a genetic algorithn with a novel encoding scheme which represents the relative positions between aircraft in the queues to gates. The encoding scheme facilitates the design of a new uniform crossover operator that ensures the feasibility of new solutions. Knowledge of how noise affect solution assessment is utilized by Lee *et al* in the next chapter "Application of Evolutionary Algorithms for Solving Multi-objective Simulation Optimization Problems" in the design of a multi-objective optimal computing budget allocation (MOCBA) algorithm. Working within the framework of the MOEA, the MOCBA adapts the number of evaluations necessary based in statistical observations of the noisy solutions. An memetic framework incorporating aspects of wrapper and filter feature selection methods for the optimization of classifiers is presented in "Feature Selection Using Single/Multi-Objective Memetic Frameworks " by Zhu *et al*. In this chapter, the wrapper method is embodied within the evolutionary process while the filter method is introduced as a local learning procedure. The next two chapters presents a class of evolutionary algorithms that approximates computationally expensive fitness functions with meta-models to reduce computational time. Radial basis functions are used as local metamodels in "Multiobjective Metamodel-Assisted Memetic Algorithms" by Georgopoulou and Giannakoglou for the design of a gas turbine power plant and the aerodynamic design of a cascade airfoil. These local metamodels are also exploited as cost-free evaluation functions for the local search process to accelerate convergence. In the chapter "Multi-Objective Robust Optimization Assisted by Response Surface Approximation and Visual Data-Mining" by Shimoyama *et al*, the MOEA is hybridized with the Kriging model for response

surface approximation and the self-organizing map is applied in the final stage for easy visualization of complicated tradeoff information. A convergence accelerator operator comprising of a neural network which learns the inverse mapping between the decision and objective space is presented in "A Convergence Acceleration Technique for Multiobjective Optimisation " by Adra *et al.* The operator works by by first suggesting some desired solutions and then applying the neural network to predict the location of these solutions in the decision variable space.

Knowledge propagation in the form of cultural evolution is considered in Part III. The first article, "Risk and Cost Tradeoff In Economic Dispatch Including Wind Power Penetration Based on Multi-objective Memetic Particle Swarm Optimization" by Lingfeng and Singh, presents a Memetic particle swarm optimization approach to find a resonable tradeoff between system risk and operational cost for the energy dispatch problem of wind power penetration. In this chapter, different fuzzy membership functions are used to reflect the various desires toward the wind power penetration and a synchronous particle local search is also applied to improve convergence. The agents of the multiagent collaborative search algorithm (MACS) presented in "Hybrid Behavioral-Based Multiobjective Space Trajectory Optimization"by Vasile are similarly endowed with both individualistic and social behaviors. A domain decomposition technique is also incorporated to improve consistency and efficiency of the MACS. In the third chaper, a new algorithm inspired by the physical phenomena of particle mechanics is suggested for high-dimensional problems in "Nature-Inspired Particle Mechanics Algorithm for Multi-objective Optimization" by Feng and Lau.

The exploitation of information for local improvement is considered in Part IV. In "Combination of Genetic Algorithms and Evolution Strategies with Self-Adaptive Switching", Okabe *et al* suggest combining genetic algorithm (GA) and evolutionary strategies in a common MOMA framework, with GA as the global searcher and ES as the local search operator. Issues such as discretization error, self-adaptation and adaptive switching, arising from the combination of GA and ES are also discussed. The chapter "Comparison between MOEA/D and NSGA-II on the multi-objective travelling salesman problem" by Peng *et al* describes an algorithm that decomposes the MO problem into multiple subproblems using different scalarizing functions. The same functions are used by the 2-opt local search heuristic process. In "Integrating Cross-Dominance Adaptation in Multi-objective Memetic Algorithms" by Caponio and Neri, MO version of simulated annealing and Rosenbrock algorithm are applied as local search operators within the MOEA. The balance between local search and genetic operators are maintained by an adaptive probablistic scheme. The optimization of dynamic MO landscapes is considered by Ray *et al* in "A Memetic Algorithm for Dynamic Multiobjective Optimization". In this chapter, the MOMA utilizes an orthogonal epsilon-constrained formulation to deal with multi-objectivity and uses a sequential quadratic programming solver to improve tracking performance. An unique approach of incorporating co-evolution and local search into differential algorithm is described by Soliman *et al* in "A Memetic Coevolutionary Multi-objective Differential Evolution Algorithm". A multiobjective memetic algorithm

which incorporates a simulated annealing algorithm as the local search operator for aerodynamic shape optimization is described in "Multiobjective Memetic Algorithm and its Application in Robust Airfoil Shape Optimization" by Song. In this chapter, local search is performed on each objective function while treating other objective functions as constraints.

We would like to express our appreciation to everyone who has made the publication of this edited book possible. First of all, we are grateful to all contributors, who are leading experts in the field of evolutionary computation, for their high-quality contributions. We are also grateful to Professor Janusz Kacprzyk for the opportunity to edit this book. We also acknowledge the editorial assistance from Springer during the preparation of the book.

May 2008

Chi-Keong Goh
Yew-Soon Ong
Kay Chen Tan

Contents

Part I

Introduction

Evolutionary Multi-Multi-Objective Optimization - EMMOO

Gideon Avigad

Mechanical Engineering Department, Ort Braude College of Engineering,
Karmiel, Israel
gideona@braude.ac.il

In this chapter the recently introduced multi-Multi-Objective Optimization Problem (m-MOOP) is described and a new evolutionary approach is suggested for its solution. The m-MOOP is a problem, which may be defined as a result of a demand to find solutions for several different multi-objective problems that are to share components. It is argued and explained here, why posing the m-MOOP as a common MOOP, is not an option and other approaches should be considered. The previously introduced Evolutionary Multi-Multi Objective Optimization (EMMOO) algorithms, which solve m-MOOPs, including the sequential, and the simultaneous one, are compared here with a new approach. The comparison is based on the loss of optimality measure.

In the chapter another extension to the suggested EMMOOs is considered and posed as a challenge. It is associated with a local search, which should be most important to the problem in hand both for improving results as well as for guarantying robustness. The chapter concludes with a discussion on the generic nature of the m-MOOP and on some possible extensions of the suggested EMMOOs to other fields of interest.

1 Introduction

Sharing components among products is an effective way to cut costs. Expenses are decreased through the reduction in components design time as well as through savings in manufacturing costs and inventory (e.g., [1]). Robertson and Ulrich, [2] point out that "By sharing components and production processes among products, companies can develop differentiated products efficiently, increase the flexibility and responsiveness of their manufacturing processes, and take market share away from competitors that develop only one product at a time." An example of the importance of sharing concerns Black & Deckers universal electric motor. According to Lehnerd [3], in the 1970s Black & Decker developed a family of universal motors for their power tools in response to a new double insulation safety regulation. Prior to that, they used different motors in each of their 122 basic tools with hundreds of variations. By paying attention to standardization and exploiting platform scaling around the motor stack length, material costs dropped from \$0.77 to \$0.42 per motor while labor costs fell from \$0.248 to \$0.045 per motor, yielding an annual savings of \$1.82M per year. Another example is Airbus, which has enjoyed a competitive advantage over Boeing due to improved

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 3–26.
springerlink.com

commonality, particularly in the cockpit. The A330 cockpit is common to all other Airbus types while Boeings 767-400 cockpit is common only with the 757. According to [4], " This has enabled the A330-200, a less efficient "shrink" of a larger aircraft, to outsell Boeings 767-400ER, a more efficient "stretch" design of a smaller aircraft."

Most of the attempts to share components between products are associated with the design of a product family. A product family is a group of related products that share common components and/or subsystems yet satisfy a variety of market niches (e.g., [5]). The level in the product hierarchy at which commonality is pursued varies; it can be focused on production processes (e.g., [6]), on modules (e.g., [7]), on product platforms (e.g., [8]) or on ***common components*** (e.g., [9]). According to [1], two types of component sharing can be used when selecting a product platform. The first is *component sharing*, in which one or more components are common to several products. The second is the sharing of "scaled" versions of components. Mathematically this can be described as *variable sharing*. The component sharing is directly related to the scope of this chapter.

Simpson [5] designates two main approaches to search for communality in product platforms, including single stage and two-stage methods. *Single-stage approaches* seek to optimize the product platform and the corresponding family of products simultaneously. On the other hand *two-stage approaches* optimize the platform first, and then, instantiate the individual products within the family during the second stage. An example for the two-stage approach is the work of Rai and Allada, ([10]). In their work, the first step of the procedure performs a multi-objective optimization using a multi-agent framework to determine the Pareto-design solutions for a given module set. The second step performs post-optimization analysis to determine the optimal platform level for a related set of product families and their variants. An example for the one-stage approach is a MOEA approach, which has been taken by Simpson and DSouza, ([11]). They used NSGA-II to facilitate a structured GA ([12]), with a commonalty as a part of a one-stage optimization algorithm. The objectives of the problem in [11] are the variation in design variables and a deviation function from a given goal. This means that the original multi-objective problem, which is common to all products, is posed as an auxiliary MOP, so that the degree of communality serves as an added objective.

The main difference between the m-MOOP approach, which was introduced in [13, 14], and other works on family of designs (e.g., [11]) lies in the consideration of different MOPs, which are coupled by common components. Here, the products ***do not share the same objective space*** and therefore neither a utility of objectives nor a common objective space setting is applicable. ***The products in the m-MOOP are not associated with different niches of the same market, as is the case of the family of designs, but rather with different markets***. To further explain the difference between an m-MOOP and a family of designs, it is noted that the same cockpit for different aircrafts is not an applicable example to the current approach. Rather, consider for example a search for a robotic arm to move an object from one place to the other, coupled with a search for a mobile robot to move another object. The objectives associated with the search for the robotic arm might be the minimization of the integral square of the end- effector's error and the minimization of its deflection. The search for a mobile robot may be associated with the maximization of the object transfer speed and the

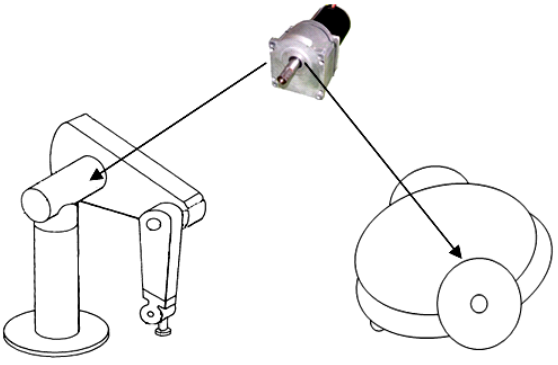


Fig. 1. A motor is shared by two products, each associated with a different MOP

minimization of the overall control force. The coupling between the MOPs is dictated by the need to use the same motor for both designs. This is illustrated in Fig. 1.

Till now there are two papers addressing the m-MOOP. In [13] the problem has been formulated, measures to assess the performances of different algorithms have been introduced and a sequential approach to solve the problem has been suggested. Drawbacks, which were identified in [13], were attended in [14] by introducing a simultaneous approach. It is noted that the name Multi-Multi Objective Optimization has also been used in [15]. The work of [15], has addressed a need for posing a problem as a multi-MOP. Nevertheless, the problem of [15] deals with one objective space and its mapping according to different contexts to multi problems.

This chapter is organized as follows. In sub-section 2.1, following [13], the problem definition is given. This is followed by two short discussions. The first, deals with the name m-MOOP and gives some justifications to it (sub-section 2.2). The second highlights the interesting aspects of the m-MOOP (sub-section 2.3). In sub-section 2.4, the loss of optimality measure, which has been introduced in [13], is explained. Sub-section 2.5, describes the existing approaches for the search for solutions to the m-MOOP and introduces a new one. In sections 2.6 and 2.7 a possible extension towards hybridization of the search for solutions to m-MOOPs and the generic nature of the problem are respectively, briefly discussed. Section 3, describes formerly obtained results and reports on some recent ones, which concern the new approach. Finally section 4 discusses the results and suggests possible directions for future work.

2 Problem Definition and Solution Approaches

2.1 Problem Definition

In the classical multi-objective search problem, such as dealt with in [16], the set of Pareto optimal solutions is sought from the set of all possible particular solutions. Any particular solution is characterized by specific values of the problem's decision variables representing a point in the problem's decision space. The set of Pareto optimal

solutions is found by comparing the performances of all particular solutions in the objective space, for non-dominance. The representation, in the objective space, of the set of non-dominated solutions is known as the Pareto front. The classical MOP is commonly formalized as follows:

$$\begin{aligned} &\min F(x) \\ &\text{s.t. } x \in X \subseteq S \subseteq R^n \end{aligned} \tag{1}$$

where x is the vector of decision variables. In general, x might be subjected to equality and/or inequality constraints, which commonly include some bounds on the decision variables. A solution $x \in X \subseteq S \subseteq R^n$, which satisfies all the constraints, is called a feasible solution. The set X, of all feasible solutions, is called the feasible region in the search space S. The MOP deals with minimizing y=F(x), the vector of K objective functions where,

$$F(x)=[f_1(x), f_2(x),\ldots, f_K(x)]^T \tag{2}$$

It can be shown that problems involving maximization, or a mixture of both min and max with respect to different objectives, may easily be transformed to the above problem. Furthermore, it should be noted that usually, due to contradicting objectives, there is no single solution to the above problem. The interest, in the classical MOP, is therefore on the trade-offs with respect to the objectives. The well-known concept of Pareto dominance supports exploring such trade-offs. The development of an optimality-based Pareto front in the objective space is based on a comparison between solutions using the idea of vector domination ([17]). Somewhat defiantly from the common MOP, the problem of the m-MOOPs is formulized as:

$$\begin{aligned} &\min F(x,y) \\ &\text{s.t.} \quad = \quad \begin{cases} x \in X \subseteq S \subseteq R_n \\ y=y_m;\ m=1,\ldots,n_{mop};\ y_m \subseteq U_m \subseteq R^{D_m} \end{cases} \end{aligned} \tag{3}$$

$$\text{where} \qquad x=[x_1,x_2,\ldots,x_n]^T \text{and} y_m=[y_m^1,y_m^2,\ldots,y_m^{D_m}]^T$$

$$\begin{aligned} &\text{where} \qquad F(x,y)=\{F^m(x,y_m), \text{for } m=1,\ldots,n_{mop}\} \\ &\qquad\qquad\ F^m(x,y_m)=[f_1^m(x,y_m),f_2^m(x,y_m),\ldots,f_{K_m}^m(x,y_m)]^T \end{aligned} \tag{4}$$

The m-MOOPs is a problem, which involves n_{mop} objective spaces and their associated design spaces. The design spaces possess communality through a common sub-space X. The relations between the design spaces and the objective spaces may be elucidated by considering a 2-MOOPs problem and by observing Fig. 2. The figure depicts two design spaces (left side of the figure). The search space of MOP_1 is associated with three parameters; x, y_1^1, y_1^2. The search space of MOP_2 is also associated with three parameters; x, y_2^1, y_2^2. Thus in equations 3 and 4, $D_1=D_2= K_1=K_2=2$. It is observed that x is the common component. Each such solution has its performances in its related objective space (see the points pointed at by arrows at the right panels of the figure). The performances are computed by using the objective functions of each MOP.

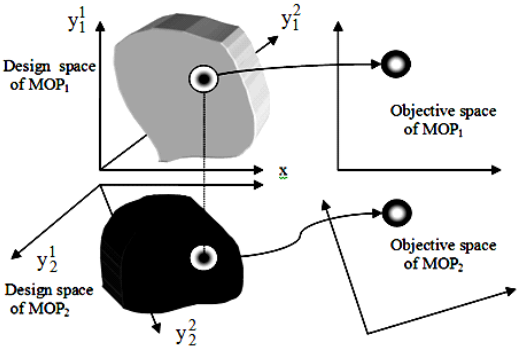


Fig. 2. The design and objective spaces of a bi-MOOP with a bi-objective problems

2.2 The Name m-MOOP- a Justification

The term "multi-MOOP" sounds like a pretentious name. MOP is fine, as it is recognized as a well defined and important problem, but m-MOOP? Is the name justified by its carrier? An advocate may argue that de-facto the problem may be posed as a MOP thus claiming that the name is unjustified. Here the view is different. This different view is based on two aspects. These aspects are related to: a. the design problem from which the m-MOOP orientates from, b. the possibility of using a posed MOP as a way to solve an m-MOOP.

***Aspect a*:** A MOP stays a MOP even though it is solved by posing it is as a single objective problem by aggregating the objective functions into a utility function! This means that the nature of the problem determines its name and not the approach by which it is solved. Each of the MOOPs of a multi-MOOP is a MOP by itself that may be attended by a totally different team of decision makers, possibly orienting from different fields of interest. The only relation between these teams may be a requirement posed by the firm executive board or the factory engineer in chief. No one would claim that designing an aircraft and designing a vessel that share between them some bolts (I am exaggerating to make the point) is a single MOP. Therefore de-yore, the problems are different but de-facto they are coupled by the requirement of commonality between their solutions.

***Aspect b*:** Posing the m-MOOP as a MOP is unrealistic from the algorithmic point of view. To elucidate this claim, examine a simpler problem where instead of considering an m-MOOP consider two single objective problems, coupled by common components. For example let one problem involve the optimization of the speed of a car and the other involves the optimization of the weight of a boat. Suppose that there is a need to use the same motor for both. Would anyone suggest that the two problems would be amalgamated into a MOP? MOP of what? There are no dominance relations here as the performances are not related to the same space. In the same manner in the case of MOPs; what is the dominance relation between two solutions, which there performances belong to several objective spaces? Maybe a surrogate space might be used to project

these performances into a single point? In any case for now it seems that approaching the solution of the problem by this way is problematic to say the least. Therefore based on the above, the name multi-MOOP seems suitable for this kind of problems.

2.3 The Interesting Aspects of m-MOOPs

m-MOOPs possess some very interesting characteristics that may result in various situations. These situations are most relevant to both the understanding of the problem as well as its solution. To elucidate these features a bi-MOOP bi-objective problem (i.e., each MOP of the m-MOOP, is associated with two objectives), is used. One of MOP_1's solution's performances in the related objective space is depicted as a blank point 'A' in Fig. 2 (left panel). A solution to the other MOP, MOP_2, which utilize the same component as the solution of MOP_1, has performances, which are depicted as a blank point 'B' in MOP_2 objective space (right panel). The performances are within the feasible spaces of the objective spaces of both MOPs. These feasible spaces are designated by encircling them with continuous curves. The Pareto front of each MOP, is designated by a thick curvature.

Improving the performances of solution 'A', by changing its parameters, may lead to a solution, which belong to the Pareto set. This means, that its performances will belong to the Pareto front, of MOP1. If the change of parameters include a change in the communal components, it is possible that the solution of MOP2 will be improved such that it belongs to the Pareto set of MOP2 and its performances would be a part of MOP2 Pareto front. These improvements are designated, in both panels of Fig. 3, by arrows pointing from the initial performances (designated by A and B) towards the improved performances (points A1 and B1 respectively). Another possible case is that improving the performances of a solution in MOP1, so that it becomes a part of the Pareto set of MOP1, may result in performances in MOP2, which are worst than the initial performances. This is shown by following the change of performances of A and B to A2 and B2 respectively. Yet another case may occur. While improving the performances of a solution A to performances at A3, the performances at B may be shifted to

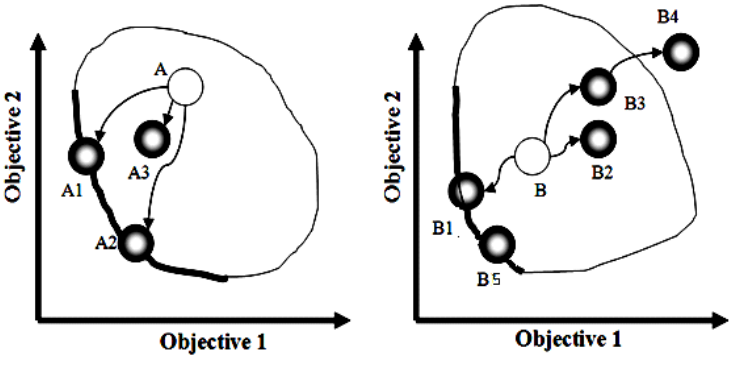


Fig. 3. Interesting cases associated with the m-MOOP

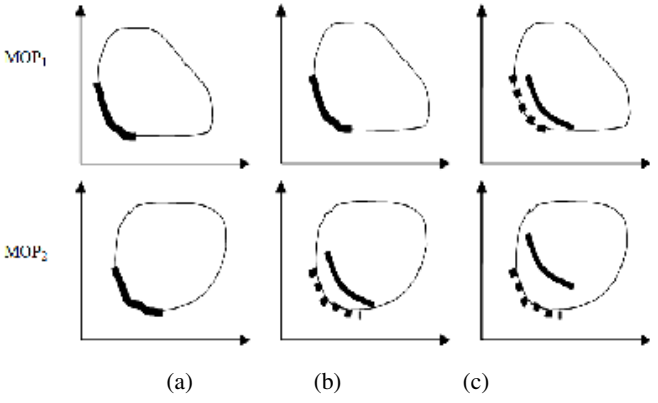


Fig. 4. Different situations, which might be associated with an m-MOOP

B3 or even B4. This implies that it is possible that a solution belonging to a Pareto set of one MOP is associated (through the common components) to an infeasible solution in MOP2 and vice versa. One last phenomenon that should be highlighted is when a solution is changed such that its solution' performances are shifted from point A1 to A2. As a result the best performances in MOP2 may be shifted from B1 to B5; nonetheless they might also be shifted from B5 to any other point in the objective space, including point B4!

Based on the above different phenomena that might be associated with solving m-MOOPs, consideration should be given to the understanding of what are the purposes of the optimization. Naturally it would be very good if the solution to the m-MOOP would correspond to solutions, which belong solely to the Pareto fronts of the problem's MOPs. Nonetheless, as discussed in the previous section, it is not always the case. Some possible situations are depicted in Fig. 4. In Fig. 4a, all solutions for the m-MOOP belong to the MOPs fronts, which are designated by bold curves. In Fig. 4b, the solutions to the m-MOOP belong to the Pareto set of one MOP (shown as bold curve in the upper panel) but are sharing components with solutions of the other MOP, which are not part of its MOP's front (this is designated in the lower panel, by a bold and dashed curves for the shifted and the Pareto front, respectively). In Fig. 4c, the solutions are not associated with none of the Pareto sets of the individual MOPs. This is designated in both panels by bold curve (shifted fronts) and dashed curves (the individual MOPs' Pareto fronts). In the latter case, trying to improve the Pareto set of one MOP causes a deterioration of performances in the other MOP. An important note is that the cases, which are depicted in Fig. 4b and Fig. 4c, might be both, solutions to the same m-MOOP.

So, if an m-MOOP may have several possible situations associated with its solution, what do we aim for? The answer is; to the best possible. So, what is the best? The answer is not straightforward. One view which has been one of the main issues in product family design (e.g., [18], is to try and find common components such that the performances in all products are not much worst than when no demand for commonality exists. In such works, it has been recognized that commonality often causes a

penalty with respect to individual product performance (e.g., [19]). According to [19], the design challenge is how to maximize commonality and optimize the family products, while satisfying individual constraints and minimizing ***performance losses***. In [13] a fundamental measure that allows assessing the results and comparing between approaches to solve an m-MOOP, has been introduced and termed: Loss Of Optimality Measure (LOM).

2.4 The Loss of Optimality Measure (LOM)

The LOM is based on the proximity indicator, ([20]) and has been initially presented in [14]. For each MOP, a representing set OS_m, m=1,., n_{mop} is found such that it is spread diversely on the MOP's Pareto front. It serves as the optimal approximation set. If the set may not be found analytically the representing set is a set found by individually running each MOP to find OS^*_m, m=1,. . ., n_{mop}. The minimal Euclidian distance between the j-th evolved solution within the m-th MOP with a representative of a set, AS_m, which is the front achieved in the last generation (of the m-MOOPs algorithm), $D^m_{j,AS_m \to OS^*_m}$ is found. The measure termed Loss of Optimality Measure, LOM, is computed as follows:

$$\mathrm{LOM} = \max(\mathrm{LOM_m})$$
$$\text{where } \mathrm{LOM_m} = \frac{D^m_{j,AS_m \to OS^*_{textm}}}{B^m} \quad m=1,\ldots,n_{mop}$$

where B^m is the maximal Euclidean distance between solution performances within the m-th MOP. This serves as a scaling factor. A lower LOM means less loss of optimality and is viewed as an advantage. The LOM for a bi-objective problem is depicted in Fig. 5. It is noted that considering the worst case over all MOPs and solutions, coincides with the notion of Pareto. That is, the uncertainty towards the preferences of the objectives is not removed, and ***any solution*** might be selected as a result of such a later-in-design, articulated preferences. Naturally, when the time to decide arrives, a DM may choose a solution associated with the smallest LOM (by taking a min over LOM_m).

When comparing between algorithms and approaches to solve an m-MOOP, other measures have been also suggested in [13] and [14]. These measures are associated

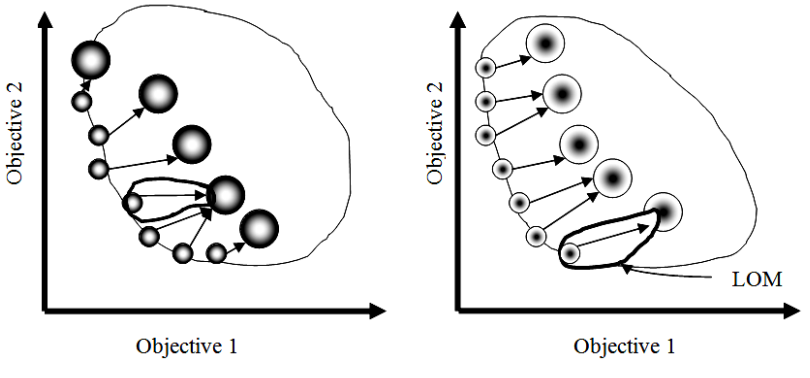


Fig. 5. The Loss of Optimality Measure assessment

with computational aspects rather than with the solutions performances aspects. The measures include the Waste of Resources Measure (WRM) and the Computational time measure. These measures have been shown to be important issues in comparing between the different possible solution approaches. In this chapter the focus is on performances, thus the LOM and not the other introduced measures is the focus. The interested reader is referred to [13] and [14] for details on comparisons based on the omitted measures.

2.5 Possible Approaches to Solve an m-MOOP

Currently there are several approaches for the solution of an m-MOOP. The approaches are depicted schematically in Fig. 6, and shortly related to, thereafter.

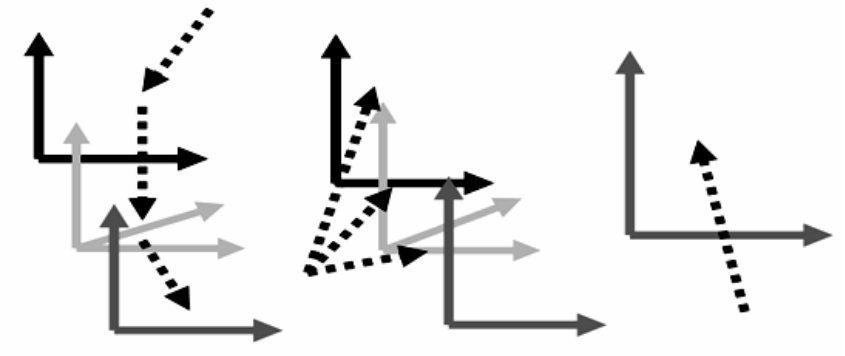


Fig. 6. Different approaches to the solution of an m-MOOP

1. Sequential approach: This approach has been introduced in [13] (see left most schematic description in Fig. 6). In this approach the MOPs of the m-MOOP are solved sequentially. The common components of the optimal set, which are found by solving the first MOP, are used as constants for the solution of the other MOPs. The method is explained and demonstrated in sections 2.5.1 and 3 respectively.

2. Simultaneous approach: This approach has been introduced in [14] (see second from left schematic description in Fig. 6). It is explained and demonstrated in sections 2.5.2 and 3 respectively.

3. Posed single objective: In this approach, which is introduced hereby, the MOPs' populations cooperate in order to reduce the loss of optimality (see sub-section 2.4). The m-MOOP is therefore posed as a single objective problem. Such posing dose not involves an aggregation function of the initial objective functions, but rather a newly imposed objective. This approach is introduced and initial results are reported upon, in sections 2.5.3 and 3 respectively.

2.5.1 The Sequential EMO Approach

The sequential approach is designed to solve the MOOPs by a stepwise approach. In such an approach, one of the MOOPs is solved, gaining its front. The optimal set for that

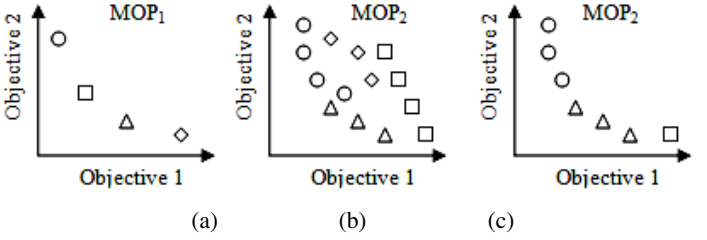


Fig. 7. Obtaining the m-MOOP front by a sequential approach

MOOP contain x^*. The search for the set x^* is then relaxed and the remaining MOPs are optimized using its values as constants. This means that the problem in equation 3 is decomposed to:

$$\begin{aligned} &\min(f(x,y)) \quad \text{if } m=1 \\ &\min_{x^*=con}(f(x,y)) \quad \text{if } m \neq 1 \end{aligned} \tag{5}$$

The EMO implementation is associated with solving sequentially n_{mop} MOPs by using a MOEA (e.g., NSGA-II) as detailed in the following pseudo algorithm.

Pseudo algorithm for the sequential m-EMO

a. Choose one of the MOPs and use an EMO to find $(x,y_1)^*$
b. For i=2,…,n_{mop} use $x\in(x,y_1)^*$ within an EMO search to find $(x,y_1)^*$ for all $i\neq1$
c. For i=2,…,n_{mop} perform non dominance sorting to $(x,y_1)^*$
d. If $\exists x\in(x,y_1)^* \wedge x\notin(x,y_1)^*$ y_i=2,…,n_{mop} eliminate this solution in MOP_1

To elucidate the sequential approach and its implementation within an m-MOOP refer to Fig. 7a-c. Fig. 7a depicts the front of a MOP, out of 2-MOOPs, which is solved first (step 'a' above). Each solution performances, is designated with a different symbol, emphasizing that each may be associated with different values for the communal parameters (the x parameters). Using each of the x values found by the solution of the first MOP, may result in a front when the second MOP is solved (step 'b' above). This means that for each point in the objective space of the first MOP there may be a set of solutions' performances in the second objective space. These are designated by corresponding symbols in Fig. 7b.

Following the demand for optimality of solutions to all MOPs, the fronts depicted in Fig. 7b are sorted for non-dominance to result in the front (step 'c' above), which is depicted in Fig. 7c. As a last step (step 'd' above), the solution which is associated with the diamond in Fig. 7a is removed (as it is not associated with optimal solutions in both MOPs). The solution to the m-MOOP is the two fronts, which are depicted in both Fig. 7a and 7c after eliminating the diamond solution in Fig. 7a.

From the study, which has been conducted and reported on in [13] the sequential approach seems to possess several drawbacks including: a) not all solutions on the front of one MOP have optimal corresponding solutions in other MOPs. This means that

there is an apparent waste of computational resources. b) The results are dependent on the sequential order in which the MOPs are selected and solved. This means that different fronts and therefore different solutions will be obtained for different sequential orders. c) The overall computation time is highly dependent on the order at which the sequential approach is implemented. d) Loss of optimality might be substantial in cases such as in Fig. 4c. e. All measures and results are highly depended on the selection of the sequential order at which the MOPs are solved. Thus if there is a clear preference to one of the products (MOP) it should be solved first.

The above drawbacks should be resolved mainly by taking a new approach, which solves the problem simultaneously. Such a simultaneous approach has been introduced in [14] and is explained in the following section.

2.5.2 Simultaneous EMO Approach

Solving the m-MOOP simultaneously by using evolution means that, the population should include individuals that may evolve all problems' solutions and are to share its resources. This approach may be viewed as a MOO problem that is built from the overlap of several component MOO problems and creates a MOO problem that is nearly decomposable. Such a view could be related to the approach of solving a composition of problems taken by Watson and Pollack [21]. Nonetheless in [21] the notion of divide and conquer is a fundamental one. That is, an individual build from two different ones survives if and only if it is better than the composing individuals. This is in contrast to the approach introduced in [14]. Taking a divide and conquer approach for the m-MOOP seems problematic as the commonalty dictates an inseparable coupling between the codes of the solutions for the different MOPs. Moreover, the situation described in Fig. 4c (where the solution involves a loss in all MOOPs), contradicts the basic understanding of coevolution by imposing an evolution where its solutions are worst than those obtained by evolving each population separately.

In nature, a composition may be an individual having more than one chromosome set from different species (e.g., [22]). Such a composition is termed allopolyploidy and is limited in the sense that it usually involves closely related species. Any structured GA ([11]) may be viewed as an allopolyploidy. In a structured GA the success of a particular set within an individual determines the probability of the individuals survival. Such individuals have been used for engineering design (e.g., [23]). An individual that initially holds the code of more than one concept, and can therefore be regarded as a specie, (e.g., [24]), has been used in several former investigations (e.g., [25, 26]). In those works, a compound individual holds a genetic code of different concepts. In [25], the decision on a concept that is to be represented by the individual has been a result of decoding the concepts' competition code. This means that an individuals survival from one generation to the next depends on the solution of one of the species, which are encoded within that individual. In [26], the survival of an individual depends on the worst case/cases of all the solutions decoded from that individual. This means that an individual may have a cluster of representations within the objective space and the survival of that individual depends on the success of a part or all its representatives in the objective space. In the approach presented in [14], all the individual's decoded solutions participate in the determination of the individuals probability for survival. Moreover the

success of the individual solutions is not measured within one multi-objective space but rather in several such spaces.

A Simultaneous Approach seems to be an appropriate approach to overcome the sequential approach deficiencies. This is mainly important within the context of this chapter, as related to deficiency '*d*', (see previous section). It is expected that because there is no preference to one MOP over the others, the pressure applied by such a search will be towards the fronts of all MOPs. Therefore it is assumed that there will not be a front/s of MOP/s that will contribute more than others to the loss of optimality and therefore a decrease in the loss of optimality is expected.

In the simultaneous approach, all MOPs are solved simultaneously while a search for communal components is conducted across the MOPs. The EMO approach suggested in [14] utilizes a structured individual and some unique procedures as explained in the following.

Algorithmic features. In this section the algorithmic features, which were introduced in [14] are described. This is followed by the description of the simultaneous EMMOO (see also [14]).

Multi problem individual: While searching for a solution to the m-MOOP, a pressure should be applied towards optimal solutions of all MOPs. The search is conducted by utilizing a Multi Problem Individual (MPI), which holds the code for all design parameters of the MOPs. In other words, an MPI codes the x vector and the y^m; $m=1,\cdots,n_{mop}$ (see section 2 for nomenclature). Such an MPI is depicted in Fig. 8.

The MPI holds the code of all design variables of all MOPs. The entire MPI's code is decoded and the resulting set of variables is utilized to compute the performances of the MPI related solutions for all the MOPs. A population of such MPIs is evolved within an EMO. It is noted that no overlap is assumed between variables of the different MOPs excluding the x variables.

Subjective Utility Rank (SUR)*:* In the m-MOOP, an MPI is associated with more than one solution (one for each MOP), and therefore may be related to different levels of

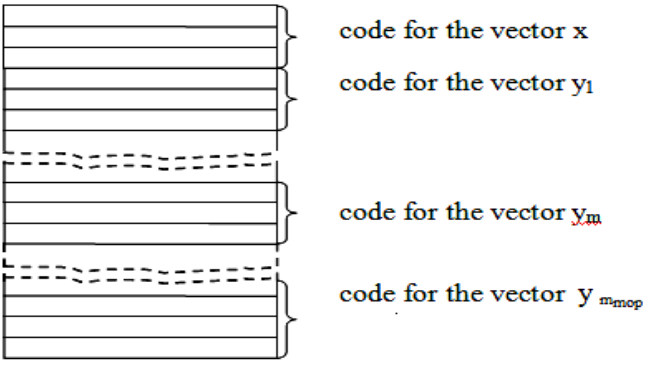


Fig. 8. The structure of the MPI

non-dominance. These levels are a result of the performances of the MPI related solutions, with respect to the other performances in all MOPs' objective spaces. The probability of an MPI to survive in the evolution should consider the optimality of the MPIs solutions in all of these spaces. This consideration leads to the need to find a measure that will represent the MPI's optimality in all MOPs. This is done as explained in the following.

Perform a non-dominated sorting (see [16]) for each of the n_{mop} problems and find for each problem the fronts, Fr_i^m, $\{i=1,, N_r^m; m=1,.., n_{mop}\}$ where N_r^{textm} is the number of fronts in the m-th MOP, in a generation. For the j-th MPI for all j=1,..,2N, find a vector of ranks such that $\overline{rank_j}=(rank_j^1,\ldots rank_j^m,\ldots rank_j^{n_{mop}}$. Assign each of the MPIs a SUR, such that the j-th individual is assigned a SUR:

$$SUR_j = \sum_{m=1}^{n_{mop}} rank_j^m + \frac{\sum_{m=1}^{n_{mop}} rank_j^m - rank_j^{low}}{n_{mop}-1} \tag{6}$$

where, $rank_j^m$, is the level of non-dominance of the j-th individual in the m-th MOP. $rank_j^{low}=\min(\overline{rank_j})$ is the lowest level of dominance associated with the j-th individual. The first expression on the right ensures that as the sum of all ranks of an MPI is lower, a lower rank is assigned to the MPI. This applies a pressure towards optimality of all MOPs solutions. The second expression on the right ensures a transverse pressure towards low loss of optimality. To further explain the SUR, suppose that one individual is associated with two ranks of 1 and 5 while another individual has solutions that are ranked 3 and 3. For both individuals the first expression value is 6, none the less the SURs are 9 and 6. This means that a pressure is applied towards solutions with low ranks in all MOPs. It is noted that this last expression is that makes the current introduced approach superior over the P-MOOP approach (see example for further explanations).

Subjective Crowding Distance: When considering crowding in the simultaneous m-MOOP, the crowding of each MPI in all MOPs has to be considered for the assignment of a crowding to the MPI. It is suggested to assign a crowding to each MPI as explained in the following.

For each of the MPIs compute the crowding distances associated with the nmop performances points in the objective spaces (see e.g., [16]). This means that the j-th MPI will have n_{mop} crowding distances d_j^m; m=1,, n_{mop}. It is noted that this distance is normalized with respect to each objective and therefore it is intrinsically normalized for all MOPs. Assign each of the MPIs a subjective crowding distance:

$$\begin{aligned} D_j &= \infty \text{if } \exists d_j^m = \infty \text{ for all } m = 1,\ldots,n_{mop} \\ D_j &= 2\cdots\max(d_j^m) - \frac{\sum_{m=1}^{n_{mop}} \max(d_j^m) - d_j^m}{n_{mop}-1} \end{aligned} \tag{7}$$

Equation 7 ensures that an MPI which codes a boundary solution at any of the MOPs will not disappear. If none of the solutions associated with the MPI are boundary solutions, then the distance assigned to it has two main effectors. One effect is of the first expression of equation 7. It assigns an initial distance according to the maximal distance that an MPI is associated with. This has the same motivation as before, namely to

prefer MPIs which have a high crowding distance at any of the MOPs. The second expression on the right penalizes the initial crowding, based on the other distances an MPI is associated with. The highest value an MPI can get is twice the maximal distance and the lowest is its maximal crowding over all MOPs. For example suppose that one MPI of a 4-MOOP is associated with the distances of 0.8, 0.8, 2, and 3, while the other has the distances of 1, 2, 2, 3. Using equation 7, both MPIs are assigned initially a value of 6. Nonetheless the final values of the distances are 4.2 and 4.66 respectively, giving the second MPI a higher probability for survival as it contributes more across the MOPs.

Simultaneous EMO for the m-MOOP. The EMO approach taken in [14] utilizes the MPI within an EMO algorithm. The algorithm is based on NSGA-II ([16]) with modifications, which allow a simultaneous solution of all the MOPs taking into consideration the need for commonality. The choice of NSGA-II is based on the need to allow a fair comparison with the sequential approach of [13], which utilized NSGA-II to search for the different MOPs fronts. The simultaneous approach algorithm and its explanation are hereby given (see further details in [14])

Simultaneous m-EMO

a. Initialize a parent population P_t, of MPIs, with a size n, n= $|P_t|$.
b. Duplicate P_t to produce an identical population Q_t (an artificial offspring population).
c. Combine parent and offspring populations to create $R_t = P_t \cup Q_t$.
d. Decode each of the MPIs' of R_t, to obtain a solution to each of the n_{mop} problems.
e. For each MPI and for each of its n_{mop} solutions find the performances in the objective space.
f. Find all MPIs' SURs (see equation 6).
g. For each SUR, find its set of MPIs, SR_r r=1,…, n_r.
h. Initialize a new parent population $P_{t+1} = \emptyset$. Set a SUR counter r=1. While $|P_{t+1}| + SR_r \leq n$, include the set SR_r in the new parent population: $|P_{t+1}| = |P_{t+1}| + SR_r$ and set r=r+1.
i. Complete the filling of P_{t+1} with the most widely spread n-$|P_{t+1}|$ MPIs using L_r where L_r=sort(D^r, >).
j. Create offspring population Q^*_{t+1} from P_{t+1} by Tournament Selection. This selection is done by choosing randomly n times, two MPIs from P_{t+1} and comparing them by tournament as follows:
   j1. If their SURs are different the MPI with the lower SUR is the winner and is placed in Q^*_{t+1}.
   j2. If their SURs are equal the MPI with the larger subjective crowding distance is the winner and is placed in Q^*_{t+1}.
k. Perform Crossover to obtain Q^{**}_{t+1}.
l. If stropping criteria is not met go to 'c'.
m. Present the MPIs, solutions'performances that are assigned with the lowest SUR.

In step 'a' and 'b', two populations of MPIs are initialized. The initialization of the second population is done by duplicating the first one. Such an initialization is done to save computations. In step 'c', as done in NSGA-II [16], the populations are merged

to find the combined population R. In step 'd' the combined population's MPIs are decoded (in a case of a binary code) to extract each of the MPIs solutions to all MOPs. These solutions are used in step 'e'to find the performances of each MPI in all MOPs. In step 'f' by using equation 1, the SUR for each MPI is found. In step 'g', the MPIs population is sorted into ranks according to the MPIs SURs. These ranks are associated with utility values and not with the original non-dominance levels. In step 'h', the elite population is filled with MPIs beginning with SR_1. This filling is commenced while an entire set SR_r may be added. If the room left for filling the elite population is smaller than the number of MPIs in the filling set the continuation of the process is altered. The alternation, which is described in step 'i', consists of adding to the elite population the most uncrowned individuals, which are sorted, based on the subjective crowding distance. In steps j-l, the algorithm is similar to that of NSGA-II [16], but using the MPIs' SURs and subjective crowding distances, instead of individual s' non-dominance levels and crowding distances. In step 'm', the fronts are presented. Here the presented individuals are those that have the lowest existing SUR. It is noted that this SUR=n_{mop}, for cases where the LOM=0. In other cases this is no longer valid, and higher SURs may be associated with the MOPs' resulting fronts.

The simultaneous approach possesses some drawbacks. These drawbacks were discussed in [14] and include: a. the need for increasing the dimension of the population as the number of MOPs involved with the m-MOOP increases, b. inconsistency of the results. Following the fact that the fitness is based on a utility function (see equation 6), there are cases where perturbations occur. Such perturbations cause the results not to converge to specific fronts. These expected perturbations, lead to a need to either attend and control these perturbations or otherwise avoid them by taking a new approach. Moreover, the need to sort the population n_{mop} times, impose a computationally expensive procedure (the non-dominance sorting). To avoid the above problematic issues a new approach is hereby suggested. It does not include sorting or a utility function.

2.5.3 A New Approach- Posed Single Objective Problem

The new approach resembles the simultaneous approach as far as its use of the same structured MPI. It is associated with the same purpose of that of the simultaneous approach, that is, to reduce the LOM. Here this demand is set as the only objective for the evolution. While striving to comply with such a demand, it may happen that optimality is not attained (see Fig. 4b, 4c).

In contrast to the simultaneous approach, which utilize the dominance relations within the m-MOOPs' associated MOPs in order to reduce the LOM, here the evolution is directly heading for reducing the LOM. This means that the m-MOOP is set as a single objective problem, with minimizing the LOM as the only objective. With respect to such posing it is noted that in order to prevent premature convergence, niching should be generally used. Considering niching in single objective problems, it seems that it may be important for the following two cases. In the first case there are a set of communal components, which are associated with a similar LOM. The second is a case where there is a preferred common component (with the lowest LOM) followed by less good solutions (with higher LOMs). The two cases and their correlation to a single objective problem are depicted in Fig. 9.

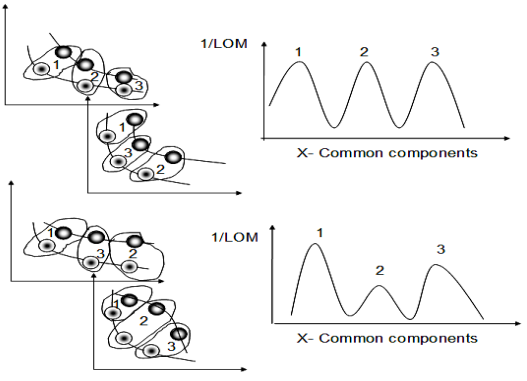


Fig. 9. Posing the m-MOOP as a single objective problem

In the upper depicted case all solutions are associated with the same LOM while in the lower depicted case the common component associated with the solutions '1' in both MOP has the lowest LOM (highest 1/LOM). The search procedure adapts a basic Genetic Algorithm with a small modification (restricted niching) and is described in the following:

Simultaneous m-EMO

a. Find all MOOPs of the m-MOOP fronts to serve as the optimal sets
b. Initialize a parent population P_t, of MPIs, with a size n, n= $|P_t|$.
c. Decode each of the MPIs' of P_t, to obtain a solution to each of the n_{mop} problems.
d. For each individuals j=1,…,n_p compute the maximal loss of optimality over all MOPs. $M_j=\max_m D^m_{OS^*_m \to j, AS_m}$
e. Assign fitness, Fit_i, to the i-th individual according to:
   $Fit_i=\frac{1}{M_j+\varepsilon}$,
   where ε is used to avois infinite fitness.
f. Penalize the fitness of all MPIs by implementing a restricted niching as follows:
   $fit^*_i=\frac{fit_i}{m_i}$
   where m_i is a sharing function:

$$m_i = sh_i(d^x_{i,j})$$

$$\text{where } sh_i(d^x_{i,j}) = \begin{cases} 1 - \frac{d^x_{i,j}}{\sigma}, & ifd^x_{i,j} < \sigma \end{cases}$$

g. Perform crossover
h. Perform mutation
i. Perform selection

2.6 Hybrid Approaches, Is It an Option?

Hybridization associated with evolutionary computation is a well recognized and developing paradigm. Hybridization might involve the incorporation of knowledge within

the evolutionary search. Such knowledge might be inserted as a result of an a-priory understanding of the problem (e.g., initial population) or during the search (e.g., adapted mutation rate). Hybridization has shown to improve the performances of EMOs by speeding the convergence to the Pareto front by allowing a reduction of the computational affords (i.e., reducing the number of function evaluations, see e.g., [27]). In the case of m-MOOPs, hybridization becomes more complicated. This declaration has several origins including;

a. One of the interesting situation associated with m-MOOPs (see sub-section 2.3) is that improving one MOP's performances may results in a deterioration of the performances associated with the other MOPs. Thus a local search within one MOP should be coupled with a local search in the other MOPs. Currently, such coupling seems like a challenge. It is noted that some of the situations considered in sub-section 2.3, makes the consideration of robustness of m-MOOPs' solutions a real pain. The scenarios of one of the MOPs' solutions may be associated with various behaviors in the other MOPs with no clear understanding of what is the worst case,

b. Excluding the sequential approach; the other approaches are based on the evolution of MPIs. This means that mutation is conducted across the entire code and not per MOP. Thus control of the EC process by altering mutation/other-operators is problematic as it has an effect across all MOPs. One way to overcome this deficiency is to consider separation of the populations into different sets and evolving simultaneously. Such a separation may lead to a new set of approaches to the solution of m-MOOPs.

2.7 Generic Nature of the Problem and Its Related Solution Approaches

Although the former two publications, [13] and [14], are utilizing an engineering example and sharing of mechanical components, it seems that the problem is generic. Apart from widening its use for other commonality aspects (e.g., common manufacturing processes etc), the m-MOOP may be related to other fields of interest. For example consider a problem, where two mobile robots' trajectories have to be planned. One robots' mission is to travel from a start point to a target point while minimizing the travel distance as well as minimizing the detection to an opponent located at the target point (see e.g., [28]). The other robot is associated with a Multi-objective traveler salesman problem. Suppose that these robots should meet at a certain point (for refueling etc). This means that the trajectories are bound to pass through a certain location in a mutual time step. It is clear that the problem is an m-MOOP. Other fields such as scheduling, and Component Based Software Development, CBSD, (e.g., [29]) might be also associated with the m-MOOP.

3 Examples

In this section bi-MOOP examples are given to demonstrate the two formerly introduced approaches and the new one. In all of the following examples NSGA-II with 20 individuals per MOP, 50% cross-over and 5% mutation is used over 300 generations. An 8 bit code is used for all design parameters. The simulations were done using MATLAB^{TM}.

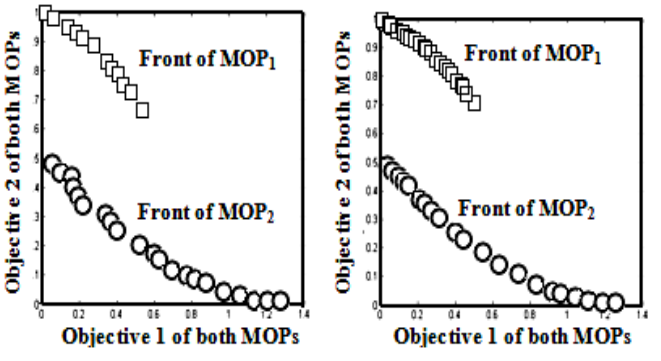
3.1 Academic Example1

In the first example the sequential and the simultaneous EMOs are utilized to solve a bi-MOOPs, which is associated with fronts of the MOPs:

$$\begin{aligned} \text{MOP}_1 : \min(f_1^1, f_2^1) \text{ where}, f_1^1 &= x \\ f_2^1 &= 1 + y_1^2 - x + 0.2\sin(\pi x) \\ \text{and:} -2.0 \le x \le 2.0, &-2.0 \le y_1 \le 2.0 \end{aligned}$$

$$\begin{aligned} \text{MOP}_2 : \min(f_1^2, f_2^2) \text{ where}, f_1^2 &= -10(e^{0.2(x^2+y_2^2)^{0.5}} + 1 \\ f_2^2 &= (x-0.5)^2 + (y_2 - 0.5)^2 \\ \text{and:} -2.0 \le x \le 2.0, &-2.0 \le y_1 \le 2.0 \end{aligned}$$

Comparing the results as obtained by the sequential approach (Fig. 10a) with those obtained by the simultaneous approach (Fig. 10b), is done by using the LOM (see section 2.4).



(a) (b)

Fig. 10. a: Sequential approach, b: Simultaneous approach

The values for the LOM computed for the sequential approach and the simultaneous approach are; LOM=0.015, LOM= 0.013, respectively. From these results it may be observed that the LOM is low and similar in both approaches. The small loss is a consequence of the fact that both MOPs fronts are associated with communal components (x values), which lie within the feasible searched interval. In [13] a mechatronic example has also been used. It showed similar results.

3.2 Academic Example2

The m-MOOP is associated with the following objectives:

$$\begin{aligned} \text{MOP}_1 : \min(f_1^1, f_2^1) \text{where}, f_1^1 &= x_1 \\ f_2^1 &= x_2 \end{aligned}$$

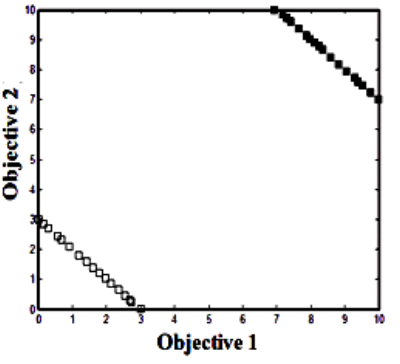


Fig. 11. Sequential development of the m-MOOP's solutions

with constraints that allow a development of a Pareto front: $g_1=x_1+ x_2-3\geq 0$, $g_2=x_2+ 30x_1-3\geq 0$, , $g_3=x_2+ 0.05x_1+0.15$ and: $0.01\leq x_1\leq 10$, $0.01\leq x_2\leq 10$

$$\text{MOP}_2 : \min(f_1^2, f_2^2) \text{where}, f_1^2 = x_1 + 2(y-5)$$
$$f_2^2 = x_2 + 2(y-5)$$

where x_1 and x_2 are the communal parameters and $5\leq y\leq 6$

In this example the equations have been chosen in such a way that when solutions become more optimal in one MOP their corresponding solutions in the other MOP are less optimal (the situation of Fig. 4c). The fronts of both MOPs are analytically identical although associated with different Pareto optimality sets. Solving each MOP separately, MOP_1 is associated with optimal solutions where $0.01\leq x_1 \leq 3$ and $0.01\leq x_2 \leq 3$ while MOP_2 is associated with $7\leq x_1 \leq 10$ and $7\leq x_2 \leq 10$. Using the sequential approach of [13] to solve this problem, results in the fronts, which are depicted in Fig. 11.

The front of one MOP is designated by blank squares whereas the related solutions of the second MOP (sequentially found) is designated by black squares. It can be seen from Fig. 4, that the loss of optimality (starting either with MOP_1 or with MOP_2) is 9.9 and that the LOM is also the minimal possible loss of optimality. It is inherent to the relation between the objectives of MOP_1 and MOP_2 that moving the front of one MOP backwards (from optimality) will result in the movement of the second MOPs' solutions towards more optimality. Therefore, as explained in section 3, it is expected that a simultaneous search of the MOPs will result in a pressure towards both fronts not preferring one MOP over the other. Fig. 12a, depicts the performances of the population R1 in MOP1 (which is associated with both P1 and Q1) designated by squares. Fig. 12b depicts the performances of the same population in MOP_2 designated by circles. In each of these figures the elite population (found according to step 'h' of the SA), P_2, performances are highlighted by filled symbols.

Inspection of figures Fig. 12a and 12b reveals that the elite population performances are concentrated at the middle of the entire set of performances in both MOPs. This means that the search pressure is not concentrated at optimal solutions for each MOP

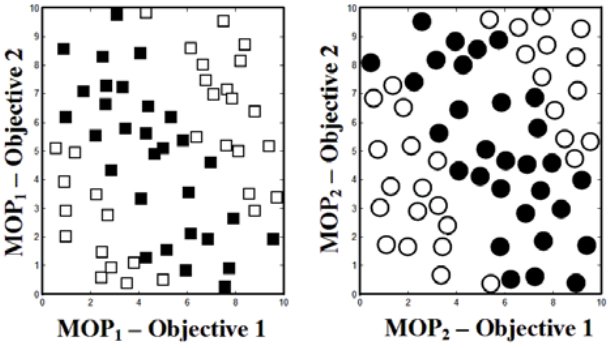


Fig. 12. MOP1 and MOP2 initial and elite populations

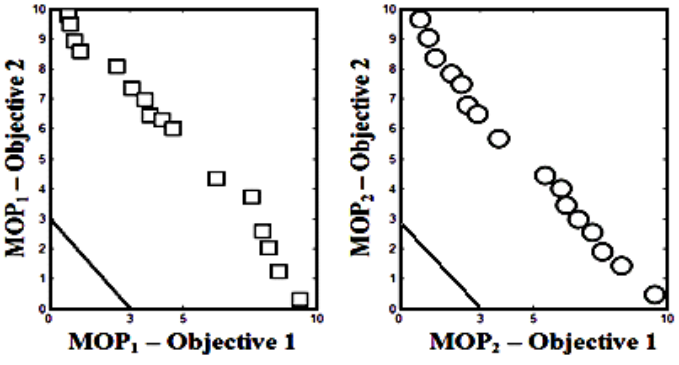


Fig. 13. The MOPs' fronts found by using the SA

but rather at solutions, which guaranty less loss of optimality in the MOPs. Running the simultaneous algorithm results in the fronts, which are depicted in Fig. 13, where the fronts of each MOP (found separately) are designated by continuous lines. It is depicted that the loss of optimality in the simultaneous approach is less than that is obtained by the sequential approach (7.2 instead of 9.9 respectively). Moreover, almost half of the solutions are associated with a loss of optimality of 4.95! (that is approximately half the loss obtained by using the sequential approach).

The posed single objective approach has been implemented for this problem. For $\sigma = 0.2$ the results are depicted in Fig. 14.

In Fig. 14, the results are shown for $\sigma = 0.2$. Running this problem again and again for different values of σ, showed two major influences of that parameter on the results. The first is the influence on the LOM and the second is the influence on the spread. These influences are depicted in Fig. 15. The left panel of Fig. 15 depicts the effect

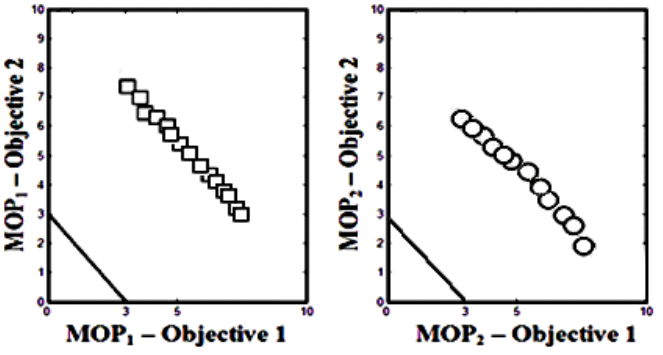


Fig. 14. The resulting front for the posed single objective, with $\sigma = 0.2$

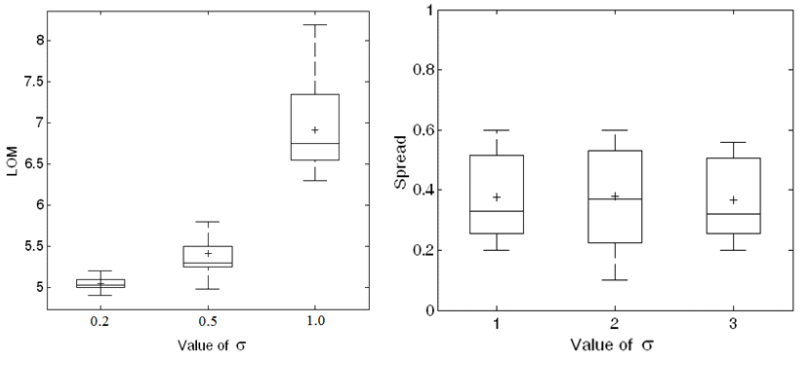


Fig. 15. Statistical results showing the effect of changing σ on the LOM (left panel) and on the spread (right panel)

of changing σ on the LOM. The results where based on the 30 individuals population and repeating the run for each σ ,30 times. It is depicted that at $\sigma = 0.2$ the LOM is the minimal (compared with the other tested values). More over most (95%) of the population is associated with a loss of optimality, which is less than 5.2. As grows the LOM is higher as well as the difference in the LOM between runs. The right panel of Fig. 15 depicts the effect of changing σ on the spread of solutions' performances. The spread has been assessed by calculating the percentage of different solutions' performances in the total population's performances. Therefore high percentage of different solutions together with a low LOM is desirable. When depicting Fig. 14 it is observed that such desirable result may be found, nevertheless based on the statistical results this is not always the case.

It is clear that there is a need to tune σ for the case in hand in order to improve the LOM. In any case the spread shows inconsistency. A possible approach to solve this issue is addressed in the following section.

4 Discussion and Future Work

In this chapter the previously introduced m-MOOP is discussed including two approaches to solve the problem, namely the sequential and the simultaneous approaches. A comparison between these approaches based on the loss of optimality measure shows the advantages of the latter over the former. Perturbations associated with the simultaneous approach, warrants the introduction of yet another approach. The newly introduced approach concerns the evolution of MPIs based on their success in reducing the LOM. The amalgamation of several MOPs into a single objective problem, which is not a utility function of some sort, is a totally new approach to solve m-MOOPs. Although the initial study has shown some promising results, further studies should be conducted especially in relation to the tunable parameter.

The statistical results obtained for the new approach warrants for an approach which will overcome both the need to tune a parameter and to improve the consistency of results. Such an approach that may solve these problems might be to use the sequential-optimization approach, which has been introduced in [30] combined with the posed single objective approach. In such a method, the m-MOOP is solved as a posed single objective problem with no niching mechanism. This is followed by running the optimization again with constraining the common components, which were found in the first run, preventing them from being the results again. This may be sequentially practiced until enough solutions are gained.

A different approach might be to consider a solution by taking game theory approaches. For example the m-MOOP might be posed as a multi objective game between players, where each tries to optimize the performances of a solution within its multi-objective space. The players are nevertheless bound to the game rules, which pose the restriction of commonality. This situation might be related to Nash equilibrium. From Wikipedia: *In game theory, the Nash equilibrium (named after John Forbes Nash, who proposed it) is a solution concept of a game involving two or more players, in which no player has anything to gain by changing only his or her own strategy unilaterally. If each player has chosen a strategy and no player can benefit by changing his or her strategy while the other players keep theirs unchanged, then the current set of strategy choices and the corresponding payoffs constitute a Nash equilibrium. Stated simply, you and I are in Nash equilibrium if I am making the best decision I can, taking into account your decision, and you are making the best decision you can, taking into account my decision. Likewise, many players are in Nash equilibrium if each one is making the best decision (s)he can, taking into account the decisions of the others. However, Nash equilibrium does not necessarily mean the best cumulative payoff for all the players involved;* ***in many cases all the players might improve their payoffs if they could somehow agree on strategies different from the Nash equilibrium.*** This means that if there was no demand for minimizing the LOM, a strategy where the MOPs do not interact could (and does) yield better results in several cases. The game theory approach, will possibly call for the need to use a separate population for each of the MOPs. Such a separation should advance hybridization as discussed in sub-section 2.6. Future work should also include a comparison between the introduced approaches for more than two coupled MOPs. Further studies should also test the approach for more compound industrial problems.

References

1. Fellini, R., Kokkolaras, M., Panos, P.Y., Perez-Duarte.: A Platform Selection Under Performance Loss Constraints in Optimal Design of Product Families. In: Proceedings of 2002 Design Engineering Technical Conferences and Computer and Information in Engineering Conference, Montreal, Canada, September 29-October 2 (2002)
2. Robertson, D., Ulrich, K.: Planning Product Platforms. Sloan Management Review 39(4), 19–31 (1998)
3. Lehnerd, A.P.: Revitalizing the Manufacture and Design of Mature Global Products. In: Guile, B.R., Brooks, H. (eds.) Technology and Global Industry: Companies and Nations in the World Economy, pp. 49–64. National Academy Press, Washington (1987)
4. Aboulafia, R.: Airbus Pulls Closer to Boeing. Aerospace America 38(4), 16–18 (2000)
5. Simpson, T.W.: Product Platform design and optimization: status and promise. In: Proceedings of 2003 Design Engineering Technical conferences and Computers and Information in Engineering Conference, Chicago, Illinois USA (2003)
6. Siddique, Z., Rosen, D.W., Wang, N.: On the Applicability of Product Variety Design Concepts to Automotive Platform Commonality. In: ASME Design Engineering Technical Conferences, Atlanta, GA, ASME, DETC/DTM-5661 (1998)
7. Chakravarty, A.K., Balakrishnan, N.: Achieving product variety though optimal choice of module variations. IIE Transactions 33, 587–598 (2001)
8. Gonzalez-Zugasti, J.P., Otto, K.N., Baker, J.D.: A Method for Architecting Product Platforms. Research in Engineering Design 12, 61–72 (2000)
9. Fisher, M.L., Ramdas, K., Ulrich, K.T.: Component Sharing in the Management of Product Variety: A Study of Automotive Braking Systems. Management Science 45(3), 297–315 (1999)
10. Rai, R., Allada, V.: Modular product family design: agent-based Pareto-optimization and quality loss function-based post-optimal analysis. International Journal of Production Research 41(17) (2003)
11. Simpson, T.W., DSouza, B.S.: Assessing variable levels of platform commonality within a product family using a multiobjective genetic algorithm. Journal of Concurrent Engineering: Research and Applications 12(2), 119–129 (2004)
12. Dasgupta, D., McGregor, D.R.: A more biologically motivated genetic algorithm: The model and some results. Cybernetics and Systems: An International Journal 25(3), 447–469 (1994)
13. Avigad, G.: Multi-multi objective optimization problem and its solution by a multi objective evolutionary algorithm. In: The proceedings of EMO 2007, Japan (2007)
14. Avigad, G.: Solving the multi-multi objective optimization problem by a simultaneous MOEA. In: The 2007 IEEE congress on evolutionary computation (2007)
15. Ponweiser, W., Vincze, M.: The multiple multi objective problem – definition, solution and evaluation. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 877–892. Springer, Heidelberg (2007)
16. 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)
17. Goldberg, D.E.: Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading (1989)
18. Fellini, R., Kokkolaras, M., Papalambros, P.Y., Perez-Duarte, A.: Platform Selection Under Performance Loss Constraints in Optimal Design of Product Families. In: Proceedings of 2002 Design Engineering Technical Conferences and Computer and Information in Engineering Conference Montreal, Canada, September 29-October 2 (2002)
19. Nelson, S.A., Parkinson II, M.B., Papalambros, P.Y.: Multicriteria Optimization in Product Platform Design. ASME Journal of Mechanical Design 123(2), 199–204 (2001)

20. Bosman, P.A.N., Thierens, D.: The balance between proximity and diversity in multiobjective evolutionary algorithms. IEEE Transactions on Evolutionary Computation 7(2), 174–188 (2003)
21. Watson, R.A., Pollack: Biosystems 69(2-3), 187–209 (May 2003) (special issue on Evolvability)
22. Werth, C.R., Guttman, S.I., Eshbaugh, W.H.: Recurring origins of allopolyploid species in Asplenium. Science 228, 731–733 (1985)
23. Parmee, I.C.: Evolutionary and adaptive strategies for efficient search across whole system engineering design hierarchies. Artificial Intelligence for Engineering Design, Analysis and Manufacturing Journal 12, 431–445 (1998)
24. Moshaiov, A.: Multi-objective design in nature and in the artificial. In: Meguid, S.A., Gomes, J.F.S. (eds.) Proceedings of the 5th International Conference on Mechanics and Materials in Design (July 2006)
25. Avigad, G., Moshaiov, A., Brauner, N.: Concept-based interactive brainstorming in engineering design. Journal of Advanced Computational Intelligence and Intelligent Informatics 8(5), 1–6 (2004)
26. Avigad, G., Moshaiov, A., Brauner, N.: MOEA-based approach to delayed decisions for robust conceptual design. In: Rothlauf, F., Branke, J., Cagnoni, S., Corne, D.W., Drechsler, R., Jin, Y., Machado, P., Marchiori, E., Romero, J., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2005, vol. 3449, pp. 584–589. Springer, Heidelberg (2005)
27. Jin, Y., Olhofer, M., Sendhof, B.: A Framework for Evolutionary Optimization with Approximate Fitness Functions. IEEE Transactions on Evolutionary Computations 6, 481–494 (2002)
28. Moshaiov, A., Avigad, G.: Interactive concept-based IEC for multi-objective search with robustness to human preference uncertainty. In: Proceedings of the, IEEE congress on evolutionary computation, Vancouver, Canada (2006)
29. Weyuker, E.J.: Testing Component-Based Software: A Cautionary Tale. IEEE Software 15(5), 54–59 (1998)
30. Avigad, G., Deb, K.: The Sequential Optimization-Constraint Multi-objective Problem and its Applications for Robust Planning of Robot Paths. In: The proceedings of the 2007 IEEE congress on evolutionary computation Singapore (2007)

Implementation of Multiobjective Memetic Algorithms for Combinatorial Optimization Problems: A Knapsack Problem Case Study

Hisao Ishibuchi, Yasuhiro Hitotsuyanagi, Noritaka Tsukamoto, and Yusuke Nojima

Department of Computer Science and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University
1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531, Japan
hisaoi@cs.osakafu-u.ac.jp, hitotsu@ci.cs.osakafu-u.ac.jp, nori@ci.cs.osakafu-u.ac.jp, nojima@cs.osakafu-u.ac.jp

In this chapter, we discuss various issues related to the implementation of multiobjective memetic algorithms (MOMAs) for combinatorial optimization problems. First we explain an outline of our MOMA, which is a hybrid algorithm of NSGA-II and local search. Our MOMA is a general framework where we can implement a number of variants. For example, we can use Pareto dominance as well as scalarizing functions such as a weighted sum in local search. Through computational experiments on multiobjective knapsack problems, we examine various issues in the implementation of our MOMA. More specifically, we examine the following issues: the frequency of local search, the choice of initial solutions for local search, the specification of an acceptance rule of local moves, the specification of a termination condition of local search, and the handling of infeasible solutions. We also examine the dynamic control of the balance between genetic operations (i.e., global search) and local search during the execution of our MOMA. Experimental results show that the hybridization with local search does not necessarily improve the performance of NSGA-II whereas local search with problem-specific knowledge leads to drastic performance improvement. Experimental results also show the importance of the balance between global search and local search. Finally we suggest some future research issues in the implementation of MOMAs.

1 Introduction

It has been demonstrated in the literature [1, 5, 7, 15, 26] that the search ability of evolutionary optimization algorithms can be improved by the hybridization with local search. Such a hybrid algorithm has often been referred to as memetic algorithms [27]. Implementation of memetic algorithms for single-objective optimization has been discussed in details [6, 23, 24, 25]. Self adaptation of search strategies in memetic algorithms has also been discussed for single-objective optimization under the name of multi-meme algorithms [23] and adaptive memetic algorithms [29, 30].

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 27–49.
springerlink.com

Whereas most memetic algorithms have been developed for single-objective optimization, real-world application problems usually involve multiple objectives. It is well-recognized that evolutionary algorithms are suitable for multiobjective optimization because a number of non-dominated solutions can be simultaneously obtained by their single run. Currently evolutionary multiobjective optimization (EMO) is one of the most active research areas in the field of evolutionary computation. Whereas a large number of multiobjective evolutionary algorithms (MOEAs) have been proposed in the literature [2], we do not have many studies on memetic algorithms for multiobjective optimization.

In single-objective memetic algorithms (SOMAs), their local search part is driven by the same objective function as in their evolutionary part. That is, the same objective function is used in global search and local search. Thus the hybridization of evolutionary algorithms with local search is straightforward in the design of SOMAs. This is not the case in the design of multiobjective memetic algorithms (MOMAs) because local search is basically a single-objective optimization technique for finding a single optimal solution (while global search in MOMAs are supposed to search for a large number of non-dominated solutions with respect to multiple objectives). Thus we need to implement a local search procedure that can handle multiple objectives for the implementation of MOMAs.

The first MOMA, which is called a multiobjective genetic local search (MOGLS) algorithm, was proposed by Ishibuchi and Murata in 1996 [13]. In MOGLS, a weighted sum fitness function is used in parent selection and local search while Pareto dominance is used for updating a secondary population (i.e., an archive population). In order to search for a large number of non-dominated solutions with a wide range of objective values, the weight vector in the weighted sum fitness function is randomly updated whenever a pair of parent solutions is chosen for crossover. The same weight vector is used in local search for an offspring solution generated by genetic operations from the chosen parents. The performance of MOGLS was examined for flowshop scheduling [14]. A variant of MOGLS with higher search ability was proposed by Jaszkiewicz [18]. It was demonstrated by Jaszkiewicz [19] that his MOGLS [18] outperformed other MOMAs (i.e., M-PAES [21] and the original MOGLS [13, 14]) and a well-known MOEA (i.e., SPEA [32]).

M-PAES (memetic Pareto archived evolution strategy) by Knowles and Corne [21] is an MOMA where Pareto dominance is used for comparing the current solution and its neighbor in local search. When they are non-dominated with each other, they are compared using a crowding measure based on a grid-type partition of the objective space. The performance of M-PAES was examined for multiobjective knapsack problems in [21] and degree-constrained multiobjective minimum-weight spanning tree problems in [22].

One advantage of local search over global search in MOMAs is its efficiency. That is, local search is much faster than global search in MOMAs. This is because local search is based on the comparison between the current solution and its neighbor while the fitness evaluation of an individual in evolutionary multiobjective optimization is based on its relation to the entire population. Moreover there exist efficient methods for evaluating local moves in many combinatorial optimization problems such as traveling salesman

problems. Thus MOMAs are usually much faster than MOEAs if their computation times are compared under the same number of examined solutions.

In this chapter, we discuss various issues related to the implementation of MOMAs through computational experiments on multiobjective 0-1 knapsack problems:

(a) Frequency of local search: How often should we use local search during the execution of MOMAs (e.g., every generation or every ten generations)?
(b) Choice of initial solutions for local search: Which solutions should we use as initial solutions for local search (e.g., all solutions or only a few solutions)?
(c) Choice of an acceptance rule for local moves: How should we compare the current solution with its neighbor (e.g., weighted sum or Pareto dominance)?
(d) Duration of local search: How many neighbors should we examine during local search from each initial solution (e.g., only a few neighbors or a large number of neighbors)?
(e) Constraint handling: How should we handle constraint conditions (e.g., repair or penalty)?
(f) Timing of local search: When should we use local search (e.g., only in the first or final generation, or in every generation)?
(g) Control of local search intensity: How should we control local search intensity (e.g., increase or decrease the local search application probability)?

As an MOMA, we use a hybrid algorithm of NSGA-II [3] and local search. Our MOMA is a general framework where we can implement various variants to examine the above issues. It is demonstrated through computational experiments that the performance of NSGA-II is not always improved by simply hybridizing it with local search. Its performance is, however, drastically improved by the hybridization when we use local search with a weighted sum-based repair scheme. That is, the use of problem-specific local search drastically improves the performance of NSGA-II. Experimental results also demonstrate the importance of the balance between local search and global search. For example, the performance of NSGA-II is severely degraded when we spend too much computation time on local search.

In this chapter, we first explain our MOMA in Section 2. Next we examine the above-mentioned issues through computational experiments in Section 3. Then we suggest some future research directions in Section 4. Finally we conclude this chapter in Section 5.

2 Multiobjective Memetic Algorithms

2.1 Basic Framework of Our Multiobjective Memetic Algorithm

Let us consider the following k-objective maximization problem:

$$\text{Maximize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_k(\mathbf{x})), \tag{1}$$

$$\text{subject to } \mathbf{x} \in \mathbf{X}, \tag{2}$$

where $\mathbf{f}(\mathbf{x})$ is a k-dimensional objective vector, $f_i(\mathbf{x})$ is its i-th element (i.e., i-th objective) to be maximized, $\mathbf{x}$ is a decision vector, and $\mathbf{X}$ is a feasible region of $\mathbf{x}$. When the following two conditions are satisfied, a feasible solution $\mathbf{x} \in \mathbf{X}$ is said to be dominated by another feasible solution $\mathbf{y} \in \mathbf{X}$ (i.e., $\mathbf{y}$ dominates $\mathbf{x}$: $\mathbf{y}$ is better than $\mathbf{x}$):

$$^{\forall} i, f_i(\mathbf{x}) \leq f_i(\mathbf{y}) \text{ and } ^{\exists} j, f_j(\mathbf{x}) < f_j(\mathbf{y}). \tag{3}$$

When this relation does not hold for a pair of solutions, they are non-dominated with each other. That is, $\mathbf{x}$ and $\mathbf{y}$ are non-dominated with each other when we can not say which is better between them using the Pareto dominance relation in (3). If there is no solution $\mathbf{y}$ in a solution set that dominates $\mathbf{x}$, $\mathbf{x}$ is a non-dominated solution in that solution set. When all solutions are non-dominated, the solution set is called a non-dominated solution set.

If there is no feasible solution $\mathbf{y}$ (in the feasible region $\mathbf{X}$ in (2)) that dominates $\mathbf{x}$, $\mathbf{x}$ is said to be a Pareto-optimal solution of the multiobjective optimization problem in (1)-(2). The set of all Pareto-optimal solutions is the Pareto-optimal solution set. Its projection on the objective space is called the Pareto front. Evolutionary multiobjective optimization is to find a set of non-dominated solutions that well approximates the Pareto-optimal solution set (i.e., the Pareto front in the objective space).

As an MOMA, we use a hybrid algorithm of NSGA-II and local search. Our MOMA is a general framework where we can implement various variants [16]. The generation update mechanism of our MOMA is illustrated in Fig. 1.

From the current population in Fig. 1, an offspring population is generated by genetic operations. Local search is applied to some offspring. An improved population is constructed by offspring improved by local search. Good individuals are selected from the current, offspring and improved populations to form the next population.

Let us denote the population size as N_{pop}. The size of the offspring population is the same as the current population. That is, N_{pop} offspring are generated by genetic operations. The size of the improved population depends on the number of offspring to which local search is applied (i.e., offspring used as initial solutions for local search). It also depends on the ability of local search to improve current solutions and its duration from each initial solution.

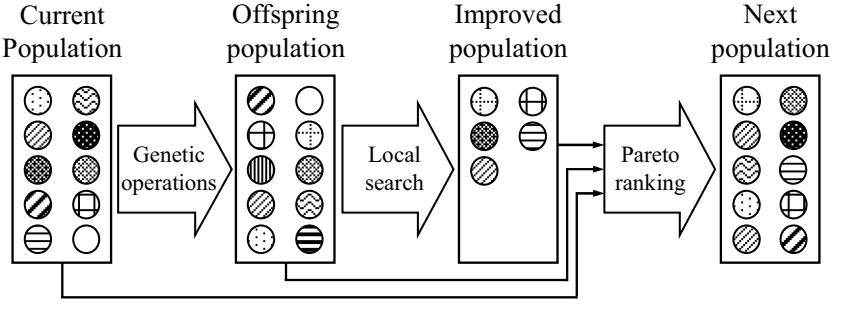


Fig. 1. Generation update in our multiobjective memetic algorithm (MOMA)

The outline of our MOMA is written as follows:

[MOMA]

Step 1: P = Initialize(P)
Step 2: While the stopping condition is not satisfied, do
Step 3: P' = Genetic Operations(P)
Step 4: P'' = Local Search(P')
Step 5: P = Generation Update($P \cup P' \cup P''$)
Step 6: End while
Step 7: Return Non-dominated(P)

First an initial population with N_{pop} solutions is randomly generated in Step 1. Then Steps 3-5 are iterated until a pre-specified stopping condition is satisfied. Finally non-dominated solutions in the final population are presented as the final result of the execution of our MOMA.

Step 3 of our MOMA is exactly the same as NSGA-II [3]. That is, each solution in the current population is evaluated by Pareto sorting in the following manner. First the best rank (i.e., Rank 1) is assigned to all the non-dominated solutions in the current population. Rank 1 solutions are tentatively removed from the current population. Next the second best rank (i.e., Rank 2) is assigned to all the non-dominated solutions in the remaining population. In this manner, ranks are assigned to all solutions in the current population. The rank of each solution is used as the primary criterion in parent selection. A crowding distance is used as the secondary criterion to differentiate between solutions with the same rank (for details, see [2, 3]). Using Pareto sorting and the crowding distance, N_{pop} pairs of parents are selected from the current population by binary tournament selection with replacement. An offspring solution is generated from each pair of parents by crossover and mutation to form an offspring population P' with N_{pop} offspring.

In Step 4, we use the following weighted sum fitness function for local search:

$$f(x) = \lambda_1 f_1(x) + \lambda_2 f_2(x) + \cdots + \lambda_k f_k(x), \tag{4}$$

where $\lambda = (\lambda_1, \lambda_2, ..., \lambda_k)$ is a non-negative weight vector. In this chapter, we first generate a set of uniformly distributed weight vectors using the following formulation as in Murata et al. (2001):

$$\lambda_1 + \lambda_2 + \cdots + \lambda_k = d, \tag{5}$$

$$\lambda_i \in \{0, 1, ..., d\} \text{ for } i = 1, 2, ..., k. \tag{6}$$

For example, we have six weight vectors (2, 0, 0), (1, 1, 0), (1, 0, 1), (0, 2, 0), (0, 1, 1), (0, 0, 2) when the value of d is specified as $d = 2$ for three-objective problems (i.e., $k = 3$). In our computational experiments, the value of d is specified as $d = 100$ for $k = 2$ (101 weight vectors), $d = 13$ for $k = 3$ (105 weight vectors), $d = 7$ for $k = 4$ (120 weight vectors) and $d = 6$ for $k = 6$ (462 weight vectors).

In Step 4, first a weight vector is randomly drawn from the weight vector set generated in the above-mentioned manner. Then an initial solution for local search is selected from the offspring population P' using tournament selection with replacement. In order

to choose a good initial solution for local search, we use a large tournament size (20 in our computational experiments). Local search is applied to the chosen initial solution with the local search application probability P_{LS}. The same weighted sum fitness function with the current weight vector is used for local search from the initial solution until local search is terminated. In local search, a neighbor is randomly generated from the current solution. When a better neighbor is found, the current solution is replaced with that neighbor. That is, we use the first move strategy where local search accepts the first improved neighbor instead of the best improved neighbor in the neighborhood of the current solution. After the replacement of the current solution, local search is applied to the updated current solution. The total number of examined neighbors in a series of local search from the initial solution is used as the termination condition in our MOMA. We denote this parameter as N_{LS} (i.e., N_{LS} is the total number of examined neighbors in a series of local search from each initial solution).

The selection of an initial solution and the probabilistic application of local search with the local search application probability P_{LS} are iterated N_{pop} times in each generation where N_{pop} is the population size.

It should be noted that a weight vector is randomly drawn whenever an initial solution for local search is to be selected. This means that local search from each initial solution is governed by the weighted sum fitness function with a different weight vector. This is to search for a variety of Pareto-optimal solutions with a wide range of objective values in the objective space.

2.2 Variants of Our Multiobjective Memetic Algorithm

Since our MOMA is a general framework of a hybrid algorithm of NSGA-II with local search, we can implement its various variants [16]. For example, Pareto dominance instead of the weighted sum fitness function can be used for comparing the current solution with its neighbor in local search. Local search does not have to be used in every generation. It can be periodically used (e.g., every two generations or every ten generations). In its extreme case, local search can be used in only the initial or final generation. Moreover, we can dynamically change the local search application probability P_{LS} during the execution of our MOMA. That is, we can use a different local search application probability in each generation of our MOMA.

Using these variants, we examine various issues related to the implementation of MOMAs in the next section. We also examine some problem-specific implementation issues such as constraint handling and repair.

3 Computational Experiments

3.1 Test Problems

As test problems, we use nine multiobjective knapsack problems of Zitzler and Thiele [32]. They are denoted as 2-250, 2-500, 2-750, 3-250, 3-500, 3-750, 4-250, 4-500 and 4-750 where "*k*-*n*" means a *k*-objective *n*-item problem. The *k*-*n* knapsack problem in [32] is written as follows:

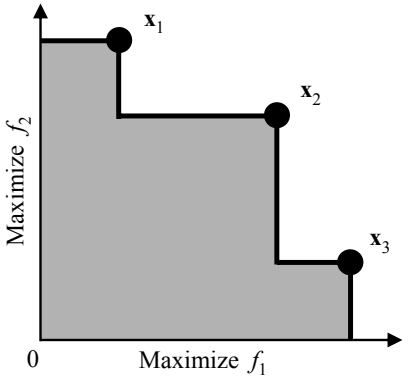


Fig. 2. Dominated region by a non-dominated solution set with three solutions

$$\text{Maximize } f(x) = (f_1(x), f_2(x), ..., f_k(x)), \tag{7}$$

$$\text{subject to} \sum_{j=1}^{n} w_{ij}x_j \leq c_i,\ i = 1,2,...,k, \tag{8}$$

$$x_j = 0 \text{ or } 1,\ j = 1,2,...,n, \tag{9}$$

where

$$f_i(x) = \sum_{j=1}^{n} p_{ij}x_j,\ i = 1,2,...,k. \tag{10}$$

In this formulation, x is an n-dimensional binary vector, p_{ij} is the profit of item j according to knapsack i, w_{ij} is the weight of item j according to knapsack i, and c_i is the capacity of knapsack i. Each solution x is handled as a binary string of length n.

Each variant of our MOMA is applied to each test problem 50 times. All experimental results reported in this chapter are average results over 50 runs.

3.2 Performance Measure

We use the hypervolume measure [31] to evaluate a non-dominated solution set obtained by each run of our MOMA. The hypervolume measure is the area (volume or hypervolume) of the dominated region by a non-dominated solution set. In Fig. 2, we show the dominated region by three non-dominated solutions x_1, x_2 and x_3 in a two-dimensional objective space. The hypervolume measure is the area of the shaded region in Fig. 2. As we can see from Fig. 2, we need a reference point for calculating the hypervolume measure. In Fig. 2, the origin of the objective space is used as the reference point. In our computational experiments, we also use the origin of the objective space as the reference point for the calculation of the hypervolume measure. For details of the calculation of the hypervolume measure, see the text book by Deb [2] on evolutionary multiobjective optimization.

For the two-objective test problems (i.e., 2-250, 2-500 and 2-750), we also use the concept of attainment surfaces [4] to visually demonstrate the average performance of each variant of our MOMA. The attainment surface is the boundary of the non-dominated region of a non-dominated solution set (e.g., bold lines in Fig. 2). Since a single non-dominated solution set is obtained by a single run of each variant of our MOMA, we obtain multiple attainment surfaces by its multiple runs. We use the 50% attainment surface as an average result over those runs. See the text book by Deb [2] for details of the calculation of the 50% attainment surface.

A number of performance measures have been proposed to evaluate the quality of non-dominated solution sets in the literature (e.g., see [2, 33]). No performance measures can simultaneously evaluate various aspects of non-dominated solution sets [33]. For example, the generational distance (GD) can only evaluate the proximity of non-dominated solution sets to the Pareto front:

$$\mathrm{GD}(S) = \frac{1}{|S|} \sum_{x \in S} \min\{d_{xy} \mid y \in S^*\}, \tag{11}$$

where S is a non-dominated solution set, S^* is the set of all Pareto-optimal solutions, and d_{xy} is the distance between a solution x and a Pareto-optimal solution y in the k-dimensional objective space:

$$d_{xy} = \sqrt{(f_1(x) - f_1(y))^2 + \cdots + (f_k(x) - f_k(y))^2}. \tag{12}$$

In some computational experiments in this chapter, we use this measure for the performance evaluation of our MOMA together with the hypervolume measure. We also use the following simple measure, which is called the range in this chapter, to evaluate the diversity of non-dominated solution sets in the k-dimensional objective space:

$$\mathrm{Range}(S) = \sum_{i=1}^{k} [\max_{x \in S}\{f_i(x)\} - \min_{x \in S}\{f_i(x)\}]. \tag{13}$$

We mainly use the hypervolume measure because it evaluates the quality of non-dominated solution sets by taking into account their proximity to the Pareto front as well as their diversity along the Pareto front. It also has other good properties as a performance measure (for characteristic features of various performance measures, see [33]). Recently the hypervolume measure has been used for performance evaluation in many studies in the EMO community.

3.3 Conditions of Computational Experiments

When we apply NSGA-II to our k-n test problem, we use the following parameter specifications:

Population size: 200 individuals,
Crossover probability: 0.8 (uniform crossover),
Mutation probability: $1/n$ (bit-flip mutation),
Termination condition: 2000 generations.

It should be noted that the mutation probability is specified by the number of items (i.e., n: string length). Various variants of our MOMA are executed under the same computation load as NSGA-II (i.e., the same total number of examined solutions as NSGA-II: 400,000 solutions). In local search, a neighboring solution is generated by applying the bit-flip operation to each bit of the current solution with the probability $4/n$.

In our MOMA, we examine the following parameter specifications:

Local search application probability: P_{LS}=0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1,
Termination condition of local search: N_{LS}=0, 1, 2, 5, 10, 20, 50, 100.

Local search is applied to each initial solution (selected from the current population) with the probability P_{LS}. When local search is applied to an initial solution, N_{LS} neighbors are examined in total in a series of local search from the initial solution. We examine all the 8×8 combinations of the above-mentioned eight values of P_{LS} and N_{LS}. It should be noted that our MOMA is exactly the same as NSGA-II when $P_{LS} = 0$ and/or $N_{LS} = 0$. This is because local search is not used when $P_{LS} = 0$ and/or $N_{LS} = 0$.

In NSGA-II and our MOMA, solutions in an initial population are randomly generated. Such a randomly generated solution does not always satisfy the constraint conditions in (8). Infeasible solutions are also generated by genetic operations (i.e., crossover and mutation) and local moves (i.e., bit-flip operations in local search). In order to transform an infeasible solution into a feasible one, we use a repair procedure based on a maximum profit/weight ratio as in Zitzler and Thiele [32]. That is, we remove items from an infeasible solution in ascending order of the following maximum profit/weight ratio until all the constraint conditions in (8) are satisfied:

$$q_j = \max\{p_{ij}/w_{ij} \,|\, i = 1,2,...,k\},\ j = 1,2,...,n. \tag{14}$$

The repair of infeasible solutions is implemented in the Lamarckian manner. That is, repaired strings are used in genetic operations in the next generation. For the comparison between the two repair schemes (i.e., Lamarckian and Baldwinian), see Ishibuchi et al. [12].

When we use the weighted sum fitness function in (4), we can utilize the information on the current weight vector $\lambda = (\lambda_1, \lambda_2, ..., \lambda_k)$ when repairing infeasible solution as in Jaszkiewicz [19, 20]. More specifically, we remove items from an infeasible solution in ascending order of the following weighted profit/weight ratio until the constraint conditions are satisfied:

$$q_j = \sum_{i=1}^{k} \lambda_i p_{ij} \Big/ \sum_{i=1}^{k} w_{ij},\ j = 1,2,...,n. \tag{15}$$

In this chapter, we refer to these two repair schemes as the maximum ratio repair and the weighted ratio repair. It should be noted that the weighted ratio repair is applicable to infeasible solutions only when we have the weight vector. Thus we always use the maximum ratio repair for infeasible solutions generated by genetic operations. We also use the maximum ratio repair when we use Pareto dominance in local search.

In Fig. 3, we demonstrate the difference between these two repair schemes using the 2-500 test problem. We applied the maximum ratio repair to an infeasible solution

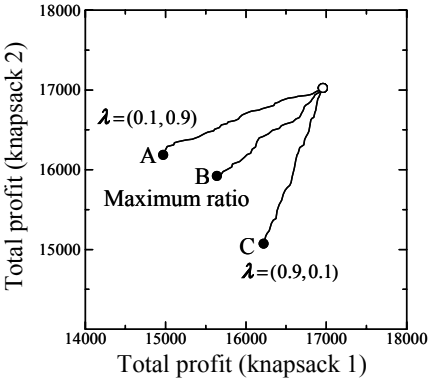


Fig. 3. Illustration of different repair schemes

denoted by an open circle in Fig. 3. Then we obtained a feasible solution B in Fig. 3. We also applied the weighted ratio repair to the same infeasible solution using two weight vectors: $\lambda = (0.9, 0.1)$ and $\lambda = (0.1, 0.9)$. When the weight for the first objective was large, we obtained a feasible solution C with a large first objective value. On the other hand, we obtained a feasible solution A with a large second objective value when the weight for the second objective was large. Similar feasible solutions were obtained from the weighted ratio repair with $\lambda = (0.5, 0.5)$ and the maximum ratio repair (the result with $\lambda = (0.5, 0.5)$ is not shown in Fig. 3).

For comparison, we also examine a penalty function method for constraint handling. In the penalty function method, we first combine the constraint condition for each knapsack in (8) into the corresponding objective function in (10) using a penalty parameter α as follows:

$$g_i(x) = f_i(x) - \alpha \cdot \max\left\{0, \sum_{j=1}^{n} w_{ij}x_j - c_i\right\}, \; i = 1, 2, ..., k. \tag{16}$$

Then the k-objective knapsack problem with the k-constraint conditions in (8)-(10) is handled as the following k-objective knapsack problem with no constraints (except for the 0-1 conditions):

$$\text{Maximize } g(x) = (g_1(x), g_2(x), ..., g_k(x)), \tag{17}$$

$$\text{subject to } x_j = 0 \text{ or } 1, \; j = 1, 2, ..., n. \tag{18}$$

It should be noted that we do not need any repair mechanisms in (17)-(18). In our computational experiments, we examine various values of α. The modified objective vector $g(x)$ instead of the original $f(x)$ is used in our MOMA (i.e., for the multiobjective fitness evaluation in NSGA-II and for local search) in the case of the penalty function method. For example, the following weighted sum fitness function is used in local search:

$$g(x) = \lambda_1 g_1(x) + \lambda_2 g_2(x) + \cdots + \lambda_k g_k(x). \tag{19}$$

It should be noted that the performance of our MOMA with the penalty function method is evaluated with respect to the original objective vector $f(x)$ using only feasible solutions in the final population.

It has been demonstrated that the performance of NSGA-II on knapsack problems strongly depends on the choice of a repair scheme [11, 12]. This is also the case in our computational experiments in this chapter. In the following subsections, we first report experimental results with the maximum ratio repair where the same repair scheme is used for global search and local search. Then we demonstrate the effect of using the weighted ratio repair for local search on the performance of our MOMA. The penalty function method is also compared with the two repair schemes.

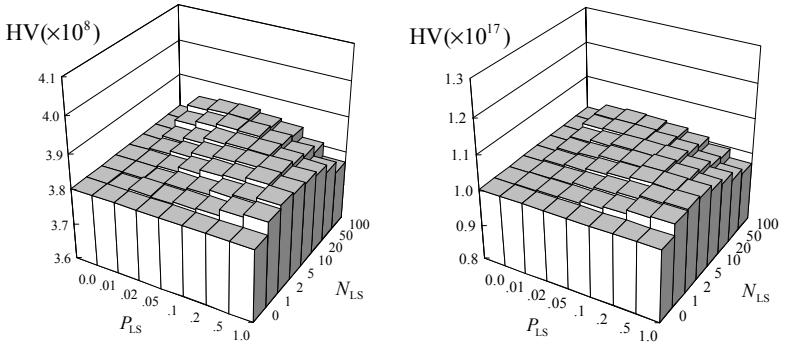
3.4 Experimental Results with Maximum Ratio Repair

In this subsection, we use the maximum ratio repair for global search and local search. In Fig. 4, we show experimental results on the 2-500 and 4-500 test problems by our MOMA with the weighted sum-based local search. The height of each bar in Fig. 4 shows the average value of the hypervolume measure for the corresponding combination of the local search application probability P_{LS} and the termination condition of local search N_{LS}. It should be noted that our MOMA is exactly the same as NSGA-II when $P_{\mathrm{LS}} = 0$ and/or $N_{\mathrm{LS}} = 0$. That is, the left-most column and the bottom row in each plot in Fig. 4 can be viewed as experimental results of NSGA-II.

In Fig. 4, we can observe performance improvement of NSGA-II by the hybridization with local search when the two parameters P_{LS} and N_{LS} are appropriately specified. We can also observe severe performance deterioration when P_{LS} and N_{LS} are too large. This is because almost all computation time was spent by local search around the top-right corner of each plot in Fig. 4. For example, when $P_{\mathrm{LS}} = 1$ and $N_{\mathrm{LS}} = 100$, local search examines 20,000 neighbors in each generation while global search generates 200 offspring. These observations in Fig. 4 show the importance of the global-local search balance in the implementation of MOMAs [17].

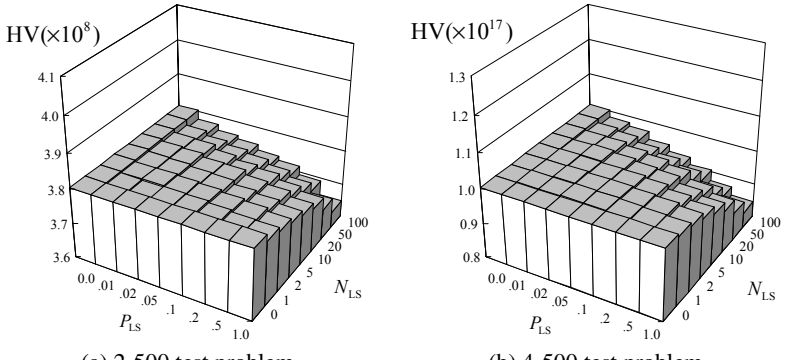
As shown in Fig. 4, good results were obtained when $P_{\mathrm{LS}} \cdot N_{\mathrm{LS}}$ was not too large or too small. More specifically, good results were obtained around the band-like region satisfying $1 \leq P_{\mathrm{LS}} \cdot N_{\mathrm{LS}} \leq 2$ in each plot of Fig. 4. In each generation, N_{pop} offspring were generated by genetic operations while $N_{\mathrm{pop}} \cdot P_{\mathrm{LS}} \cdot N_{\mathrm{LS}}$ neighbors were examined in local search of our MOMA. Thus $P_{\mathrm{LS}} \cdot N_{\mathrm{LS}}$ can be viewed as the ratio of local search over global search. When this ratio was too large (i.e., around the top-right corner in each figure in Fig. 4), global search of NSGA-II was not fully utilized in our MOMA because almost all computation time was spent by local search. As a result, good results were not obtained. On the other hand, when this ratio was too small (i.e., around the bottom-left corner), local search was not fully utilized in our MOMA.

In Fig. 5, we show experimental results by our MOMA with the Pareto dominance-based local search. From the comparison between Fig. 4 and Fig. 5, we can see that better results were obtained by the weighted sum-based local search in Fig. 4 than the Pareto dominance-based local search in Fig. 5. In Fig. 5, we can not observe any improvement in the performance of NSGA-II ($P_{\mathrm{LS}} = 0$ and/or $N_{\mathrm{LS}} = 0$) by the hybridization with local search (i.e., $P_{\mathrm{LS}} > 0$ and $N_{\mathrm{LS}} > 0$).



(a) 2-500 test problem (b) 4-500 test problem

Fig. 4. Hypervolume measure by our MOMA with the weighted sum-based local search where the maximum ratio repair scheme was used in global search and local search



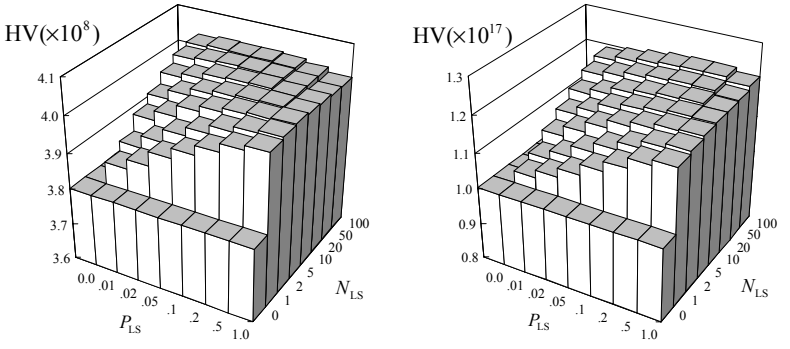
(a) 2-500 test problem (b) 4-500 test problem

Fig. 5. Hypervolume measure by our MOMA with the Pareto dominance-based local search where the maximum ratio repair scheme was used in global search and local search

Whereas we can see from Fig. 4 that the performance of NSGA-II was improved by the hybridization with local search, the performance improvement is not impressive in each plot of Fig. 4. That is, the simple hybridization with local search did not drastically improve the performance of NSGA-II.

3.5 Experimental Results with Weighted Ratio Repair

In this subsection, we use the weighted ratio repair in the weighted sum-based local search. In Fig. 6, we show experimental results on the 2-500 and 4-500 test problems by our MOMA with the weighted sum-based local search. We can observe clear performance improvement of NSGA-II (with $P_{\mathrm{LS}} = 0$ and/or $N_{\mathrm{LS}} = 0$ in Fig. 6) by the hybridization with local search (with $P_{\mathrm{LS}} > 0$ and $N_{\mathrm{LS}} > 0$).



(a) 2-500 test problem (b) 4-500 test problem

Fig. 6. Hypervolume measure by our MOMA with the weighted sum-based local search where the weighted ratio repair was used for local search

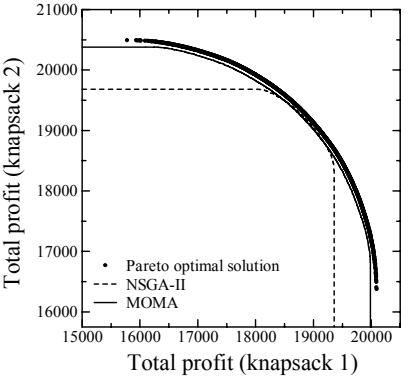
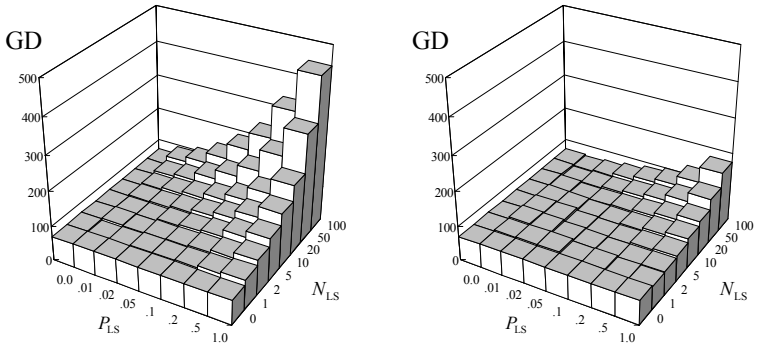


Fig. 7. 50% attainment surfaces by NSGA-II and our MOMA on the 2-500 problem

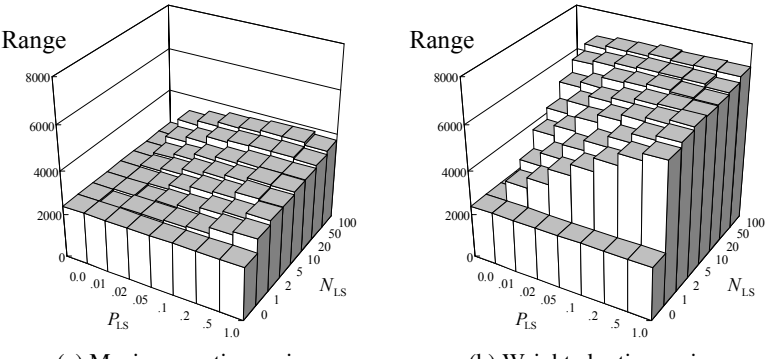
In order to visually demonstrate the improvement by the hybridization with local search, we show the 50% attainment surfaces by NSGA-II and our MOMA with $P_{\mathrm{LS}} = 0.02$ and $N_{\mathrm{LS}} = 100$ for the 2-500 test problem in Fig. 7 together with the true Pareto-optimal solutions. We can see that the hybridization of local search drastically improved the performance of NSGA-II with respect to the diversity of solutions along the Pareto front.

The two repair schemes are compared using the generational distance for the 2-500 test problem in Fig. 8. It should be noted that the smaller values of this measure mean the better proximity of solution sets to the Pareto front. It should be also noted that the experimental results are exactly the same between the two plots in Fig. 8 in the case of $N_{\mathrm{LS}} = 0$ and/or $P_{\mathrm{LS}} = 0$. In this case, our MOMA is actually the same as NSGA-II with no local search. We can see from Fig. 8 that the hybridization with local search did not improve the convergence property of NSGA-II (i.e., it did not decrease the generational distance). On the contrary, the use of too much computation time on local search (i.e.,



(a) Maximum ratio repair (b) Weighted ratio repair

Fig. 8. Comparison between the two repair schemes in our MOMA using the generational distance for the 2-500 test problem



(a) Maximum ratio repair (b) Weighted ratio repair

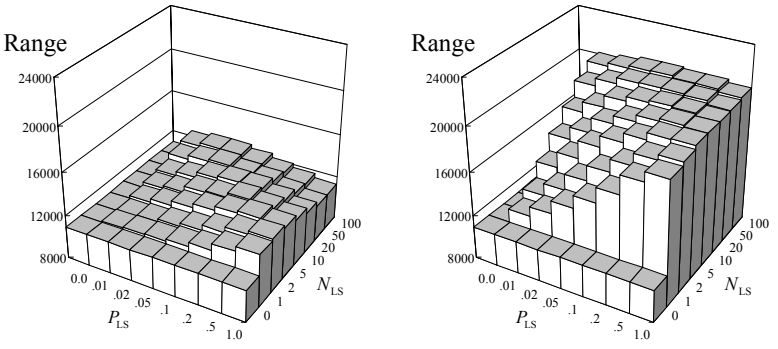
Fig. 9. Comparison between the two repair schemes in our MOMA using the range measure for the 2-500 test problem

around the top-right corner in each plot of Fig. 8) degraded the convergence (especially in the case of the maximum ratio repair).

In the same manner as Fig. 8, the two repair schemes are compared using the range measure in Fig. 9. We can observe in Fig. 9 that the use of the weighted ratio repair significantly increased the diversity of solutions in our MOMA in comparison with the maximum ratio repair. We obtained the same observation for the other test problems. For example, we show experimental results on the 4-500 test problem in Fig. 10. In Fig. 10 (a), we can also observe a negative effect of too much local search on the diversity of solutions (around the top-right corner).

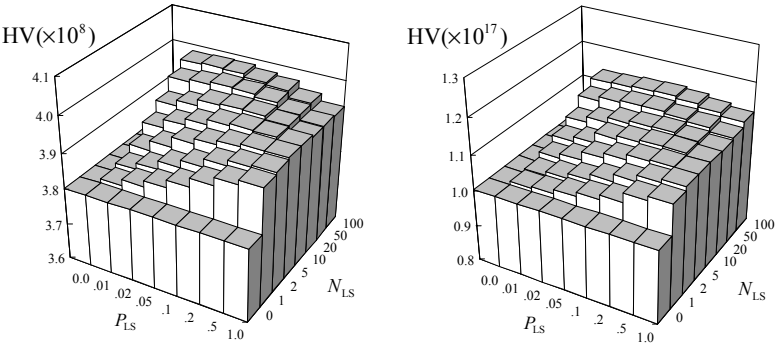
3.6 Choice of Initial Solutions for Local Search

We examined other implementation issues using our MOMA with the weighted ratio repair in the weighted sum-based local search. First, let us discuss the choice of



(a) Maximum ratio repair (b) Weighted ratio repair

Fig. 10. Comparison between the two repair schemes in our MOMA using the range measure for the 4-500 test problem

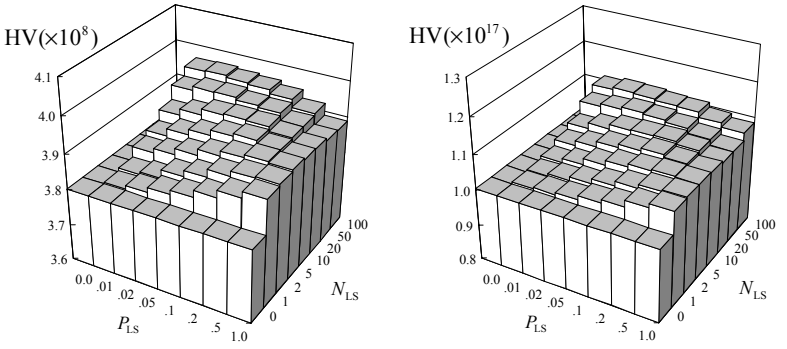


(a) 2-500 test problem (b) 4-500 test problem

Fig. 11. Experimental results by our MOMA where the selection pressure in the selection of initial solutions for local search was weaker than Fig. 6. The tournament size was 20 in Fig. 6 while it was 2 in this figure.

initial solutions for local search. In the above-mentioned computational experiments in Fig. 6, we used the weighted sum-based tournament selection with tournament size 20 in the selection of initial solutions for local search. We also performed computational experiments using binary tournament selection. Experimental results are summarized in Fig. 11. From the comparison between Fig. 6 and Fig. 11, we can see that the performance of our MOMA was deteriorated by decreasing the tournament size in the selection of initial solutions for local search.

We also examined the case of random selection of initial solutions for local search. In this case, initial solutions were randomly chosen for local search from the offspring population in each generation. Experimental results are summarized in Fig. 12. From the comparison of Fig. 12 with Fig. 6 and Fig. 11, we can see that the performance of



(a) 2-500 test problem (b) 4-500 test problem

Fig. 12. Experimental results by our MOMA where initial solutions for local search were randomly chosen from the offspring population in each generation

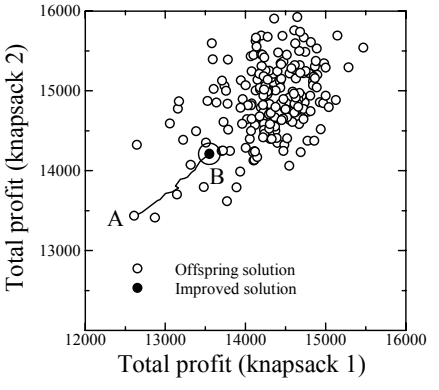


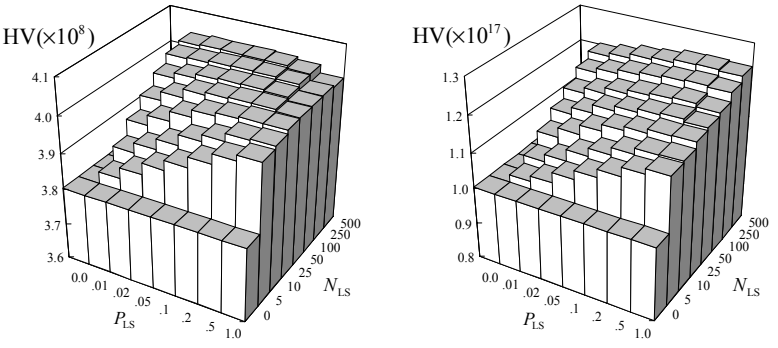
Fig. 13. Illustration of the waste of computation time by local search. The initial solution A was improved by local search to the final solution B denoted by the closed circle.

our MOMA was further deteriorated by the random choice of initial solutions for local search. This means that the choice of good solutions is essential in our MOMA.

The deterioration in the performance of our MOMA by the random choice of initial solutions for local search can be explained by the waste of time for improving poor solutions. This is illustrated in Fig. 13 where local search is applied to solution A. While the initial solution A is significantly improved by local search, the final solution B denoted by the closed circle in Fig. 13 is still a poor solution if compared with other solutions.

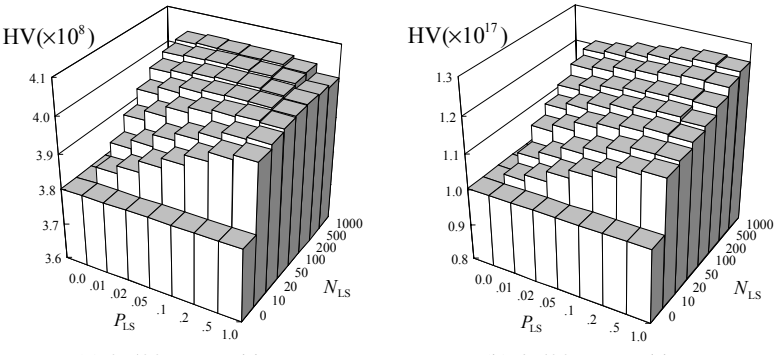
3.7 Frequency of Local Search

Next we discuss the frequency of local search. Whereas we used local search in every generation of our MOMA in the previous computational experiments, it is possible to use local search less frequently. We examined two specifications of the frequency



(a) 2-500 test problem (b) 4-500 test problem

Fig. 14. Experimental results by our MOMA where local search was employed every five generations



(a) 2-500 test problem (b) 4-500 test problem

Fig. 15. Experimental results by our MOMA where local search was employed every ten generations

of local search: every five generations and every ten generations. In order to maintain the same global-local search balance, we used a five (ten) times larger value as the termination condition N_{LS} of local search than the case of every generation when local search was employed every five (ten) generations. Experimental results are summarized in Fig. 14 and Fig. 15. From the comparison of Fig. 6 with Fig. 14 and Fig. 15, we can see that similar results were obtained from the three specifications of the local search frequency (i.e., every generation, every five generations, and every ten generations). This is because we used the same global-local search balance in the three cases by adjusting the local search termination condition N_{LS}.

3.8 Constraint Handling

In this subsection, we compare the penalty function method with the two repair schemes: maximum ratio repair and weighted ratio repair. In our computational experiments, we

Table 1. Comparison between the penalty function method and the two repair schemes. The best result is highlighted by bold face for each test problem.

Test	Repair		Penalty function method				
Problem	Maximum	Weighted	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$	$\alpha = 50$
2-250	1.000	**1.038**	0.939	0.967	0.999	0.999	0.995
2-500	1.000	**1.047**	0.962	0.979	0.999	0.998	0.993
2-750	1.000	**1.053**	0.965	0.983	0.994	0.993	0.987
3-250	1.000	**1.075**	0.838	0.936	0.986	0.986	0.987
3-500	1.000	**1.101**	0.890	0.961	0.989	0.988	0.985
3-750	1.000	**1.115**	0.902	0.966	0.989	0.987	0.984
4-250	1.000	**1.106**	0.786	0.881	0.895	0.886	0.864
4-500	1.000	**1.157**	0.831	0.905	0.919	0.901	0.876
4-750	1.000	**1.179**	0.858	0.917	0.921	0.910	0.888

examined five specifications of the penalty parameter: $\alpha = 0.5, 1, 5, 10, 50$. Only feasible solutions in the final population were used in the calculation of the hypervolume measure. The local search application probability P_{LS} and the termination condition of local search N_{LS} were specified as $P_{\mathrm{LS}} = 0.1$ and $N_{\mathrm{LS}} = 20$ in all computational experiments in this subsection. We chose these values of P_{LS} and N_{LS} since good results were obtained in Fig. 6 from this combination.

Experimental results are summarized in Table 1 where all results (i.e., hypervolume values) are normalized using those by the maximum ratio repair. For each test problem, the best result is highlighted by bold face. From this table, we can see that much better results were obtained by the weighted ratio repair than the penalty function method. In all cases, the penalty function method was inferior to even the maximum ratio repair in Table 1.

3.9 Dynamic Control of Global-Local Search Balance

In this subsection, we discuss the effect of dynamically changing the local search application probability P_{LS}. As in the previous subsection, we specified the termination condition of local search N_{LS} as $N_{\mathrm{LS}} = 20$ in all computational experiments in this subsection. When the local search application probability P_{LS} was constant, we specified it as $P_{\mathrm{LS}} = 0.1$. We examined the linear increase of P_{LS} from 0 to 0.2 and its linear decrease from 0.2 to 0. We also examined two extreme cases where local search was employed only in the initial or final generation. In these two extreme cases, P_{LS} was specified as $P_{\mathrm{LS}} = 1$ only in the first or final generation. The value of N_{LS} was specified as $N_{\mathrm{LS}} = 1320$ so that the overall global-local search balance was the same in all computational experiments in this subsection.

Experimental results are summarized in Table 2 where all results (i.e., hypervolume values) are normalized using those by the case of the constant value of the local search application probability P_{LS} (i.e., $P_{\mathrm{LS}} = 0.1$). For each test problem, the best result is highlighted by bold face. In Table 2, the worst results were obtained for almost all test problems (especially for the four-objective test problems) when we used local search

Table 2. Effect of dynamically changing the local search application probability. The best result is highlighted by bold face for each test problem.

Test	Constant	Two extreme cases		Linear change	
Problem	(0.1)	Initial	Final	Increase	Decrease
2-250	1.0000	0.9994	0.9978	**1.0001**	0.9998
2-500	**1.0000**	0.9918	0.9944	0.9998	0.9986
2-750	**1.0000**	0.9815	0.9889	0.9987	0.9973
3-250	1.0000	1.0009	**1.0113**	1.0040	1.0000
3-500	1.0000	0.9924	1.0029	**1.0030**	0.9974
3-750	1.0000	0.9823	0.9994	**1.0021**	0.9947
4-250	1.0000	0.9954	**1.0453**	1.0113	0.9964
4-500	1.0000	0.9824	**1.0438**	1.0124	0.9930
4-750	1.0000	0.9698	**1.0455**	1.0086	0.9880

Table 3. Experimental results with $P_{\mathrm{LS}} = 0.05$ (constant). Except for the local search application probability (i.e., the overall balance between global and local search), we used the same settings in Table 2 and Table 3.

Test	Constant	Two extreme cases		Linear change	
Problem	(0.05)	Initial	Final	Increase	Decrease
2-250	1.0000	0.9929	0.9926	**1.0014**	1.0012
2-500	1.0000	0.9745	0.9875	**1.0006**	0.9994
2-750	1.0000	0.9590	0.9801	**1.0003**	0.9989
3-250	1.0000	0.9946	0.9984	**1.0063**	1.0022
3-500	1.0000	0.9743	0.9859	**1.0064**	1.0008
3-750	1.0000	0.9466	0.9762	**1.0074**	1.0000
4-250	1.0000	0.9887	**1.0265**	1.0191	1.0041
4-500	1.0000	0.9583	1.0096	**1.0210**	1.0014
4-750	1.0000	0.9332	1.0020	**1.0186**	0.9977

only in the first generation. Similar performance was obtained from three cases: constant, linear increase and linear decrease. Difference in the average hypervolume values in these three cases is less than 1% for almost all test problems. For the four-objective test problems, the best results were obtained by the use of local search only in the final generation. This may be because NSGA-II worked well for preparing a wide variety of initial solutions for local search. Since NSGA-II did not work well on many-objective problems, the use of local search only in the initial solution is not a good idea. In this case, good solutions found in the first generation by local search can be lost by global search in NSGA-II.

Since the application probability of local search and its termination condition were specified as $P_{\mathrm{LS}} = 0.1$ and $N_{\mathrm{LS}} = 20$ in Table 2, the balance between global and local search was $1 : 2$ (it is calculated as $1 : P_{\mathrm{LS}} \cdot N_{\mathrm{LS}}$). When we modified it as $1 : 1$ by specifying the constant local search application probability as 0.05 (instead of 0.1 in Table 2), we obtained experimental results in Table 3. From Table 3, we can see that the

best results were obtained from the linear increase of the local search application probability for almost all test problems. We can also see that better results were obtained for several test problems by the linear decrease than the constant specification. On the other hand, we can observe a clear deterioration in the performance of our MOMA in the two extreme cases (except for the use of local search in the final generation for the four-objective test problems). We also examined other settings of computational experiments. Then we obtained different observations from those in Table 2 and Table 3 (e.g., see [10]). These experimental results suggest that the choice of an appropriate control strategy of the local search application probability strongly depends on the problem and the parameter specifications. Thus the self-adaptation of the local search application probability, which is a future research issue for multiobjective optimization, seems to be a better idea than the use of a pre-specified control strategy.

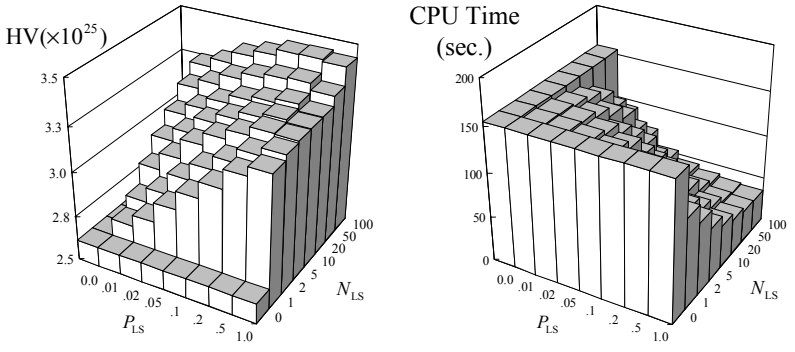
4 Future Research Issues

It has been pointed out by some studies [8, 9, 20] that Pareto dominance-based MOEAs such as NSGA-II [3] and SPEA [32] do not work well on many-objective problems. This is because almost all solutions in each generation become non-dominated with each other when we apply MOEAs to many-objective problems. This means that Pareto-dominance can not generate a strong selection pressure toward the Pareto front. Thus the convergence of solutions to the Pareto front in Pareto dominance-based MOEAs is severely deteriorated by the increase in the number of objectives. From the same reason, Pareto dominance-based local search does not work well for many-objective problems [28].

In Fig. 16, we show experimental results of our MOMA on a six-objective 500-item test problem. We used the weighted sum-based local search with the weighted ratio repair in Fig. 16. We generated the 6-500 test problem in the same manner as in [32]. The performance of our MOMA was examined by the hypervolume measure in Fig. 16 (a) and the CPU time in Fig. 16 (b).

As shown in Fig. 16 (a), the performance of NSGA-II (with $P_{\mathrm{LS}} = 0$ and/or $N_{\mathrm{LS}} = 0$) was drastically improved by the hybridization with local search ($P_{\mathrm{LS}} > 0$ and $N_{\mathrm{LS}} > 0$). This is because the selection pressure toward the Pareto front was introduced by the use of the weighted sum-based local search. At the same time, the CPU time of NSGA-II was drastically decreased by the hybridization. NSGA-II, which found solution sets with the smallest average value of the hypervolume measure in Fig. 16 (a), spent the longest average CPU time in Fig. 16 (b). This is because the multiobjective fitness evaluation in NSGA-II is more time-consuming especially in the case of many-objective problems than the weighted sum-based fitness evaluation.

As shown in Fig. 16, our MOMA worked on many-objective problems much better than NSGA-II. The design of MOMAs for many-objective problems and their performance evaluation are promising future research issues. This is because Pareto dominance-based MOEAs do not work well on many-objective problems. That is, a new framework of evolutionary multiobjective optimizers is needed for the handling of many objectives.



(a) Hypervolume measure (b) CPU time

Fig. 16. Experimental results on the 6-500 test problem by our MOMA with the weighted ratio repair in the weighted sum-based local search

Another promising future research issue is the self-adaptation of local search strategies in MOMAs. Whereas this issue has been discussed in a number of studies for single-objective optimization in SOMAs [23, 29, 30], almost all existing MOMAs use fixed local search strategies.

The hybridization of local search with other population-based search algorithms such as particle swarm optimization (PSO) may be a promising future research area in MOMAs. Performance comparison among different frameworks of MOMAs (e.g., GA-based, PSO-based, etc.) seems to be an interesting future research issue.

It is also required to develop problem-specific MOMAs for real-world applications. As we have already demonstrated in this chapter, the use of domain knowledge often drastically improves the performance of MOMAs. The success of MOMAs on real-world applications will help the development of the field of MOMAs. Thus real-world applications of MOMAs are required for further development of MOMAs.

5 Conclusions

In this chapter, we demonstrated that the performance of NSGA-II can be significantly improved by the hybridization with local search. One important issue in the implementation of such a hybrid algorithm for multiobjective optimization (i.e., MOMA) is the balance between global search and local search. The performance of NSGA-II was often degraded by the hybridization when this balance was inappropriate. Another important issue is the use of problem-specific knowledge in local search. When we used the same operator as mutation for local search (i.e., bit-flip operation with the maximum profit ratio repair), the performance improvement of NSGA-II by the hybridization was not impressive. On the other hand, the use of the weighted ratio repair in the weighted sum-based local search drastically improved the performance of our MOMA. That is, the performance of NSGA-II was drastically improved by the hybridization with local search when we used problem-specific knowledge in local search. We also demonstrated that the use of local search only in the initial generation was not a good idea.

Good results were obtained when we used local search throughout the execution of our MOMA or only in the final generation.

This work was partially supported by Grant-in-Aid for Scientific Research (B): KAKENHI (17300075).

References

1. Bersini, H., Dorigo, M., Langerman, S., Seront, G., Gambardella, L.: Results of the First International Contest on Evolutionary Optimization. In: Proc. of 1996 IEEE International Conference on Evolutionary Computation, pp. 611–615 (1996)
2. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation 6, 182–197 (2002)
4. Fonseca, C.M., Fleming, P.J.: On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers. PPSN IV. LNCS, vol. 114, pp. 584–593. Springer, Heidelberg (1996)
5. Freisleben, B., Merz, P.: A Genetic Local Search Algorithm for Solving Symmetric and Asymetric Traveling Salesman Problems. In: Proc. of 1996 IEEE International Conference on Evolutionary Computation, pp. 616–621 (1996)
6. Hart, W.E.: Adaptive Global Optimization with Local Search. Ph. D. Thesis, University of California, San Diego (1994)
7. Hart, W.E., Krasnogor, N., Smith, J.E. (eds.): Recent Advances in Memetic Algorithms, pp. 3–27. Springer, Berlin (2005)
8. Hughes, E.J.: Evolutionary Many-Objective Optimization: Many Once or One Many? In: Proc. of 2005 Congress on Evolutionary Computation, pp. 222–227 (2005)
9. Ishibuchi, H., Doi, T., Nojima, Y.: Incorporation of scalarizing fitness functions into evolutionary multiobjective optimization algorithms. 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. 493–502. Springer, Heidelberg (2006)
10. Ishibuchi, H., Hitotsuyanagi, Y., Nojima, Y.: An Empirical Study on the Specification of the Local Search Application Probability in Multiobjective Memetic Algorithms. In: Proc. of 2007 IEEE Congress on Evolutionary Computation, pp. 2788–2795 (2007)
11. Ishibuchi, H., Kaige, S.: Effects of Repair Procedures on the Performance of EMO Algorithms for Multiobjective 0/1 Knapsack Problems. In: Proc. of 2003 Congress on Evolutionary Computation, pp. 2254–2261 (2003)
12. Ishibuchi, H., Kaige, S., Narukawa, K.: Comparison between lamarckian and baldwinian repair on multiobjective 0/1 knapsack problems. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005, vol. 3410, pp. 370–385. Springer, Heidelberg (2005)
13. Ishibuchi, H., Murata, T.: Multi-Objective Genetic Local Search Algorithm. In: Proc. of 1996 IEEE International Conference on Evolutionary Computation, pp. 119–124 (1996)
14. Ishibuchi, H., Murata, T.: A Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling. IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews 28, 392–403 (1998)
15. Ishibuchi, H., Murata, T., Tomioka, S.: Effectiveness of Genetic Local Search Algorithms. In: Proc. of 7th International Conference on Genetic Algorithms, pp. 505–512 (1997)
16. Ishibuchi, H., Narukawa, K.: Some issues on the implementation of local search in evolutionary multiobjective optimization. In: Deb, K., et al. (eds.) GECCO 2004, vol. 3102, pp. 1246–1258. Springer, Heidelberg (2004)

17. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling. IEEE Trans. on Evolutionary Computation 7, 204–223 (2003)
18. Jaszkiewicz, A.: Genetic Local Search for Multi-Objective Combinatorial Optimization. European Journal of Operational Research 137, 50–71 (2002)
19. Jaszkiewicz, A.: On the Performance of Multiple-Objective Genetic Local Search on the 0/1 Knapsack Problem - A Comparative Experiment. IEEE Trans. on Evolutionary Computation 6, 402–412 (2002)
20. Jaszkiewicz, A.: On the Computational Efficiency of Multiple Objective Metaheuristics: The Knapsack Problem Case Study. European Journal of Operational Research 158, 418–433 (2004)
21. Knowles, J.D., Corne, D.W.: M-PAES: A Memetic Algorithm for Multiobjective Optimization. In: Proc. of 2000 Congress on Evolutionary Computation, pp. 325–332 (2000)
22. Knowles, J.D., Corne, D.W.: A Comparison of Diverse Approaches to Memetic Multiobjective Combinatorial Optimization. In: Proc. of 2000 Genetic and Evolutionary Computation Conference Workshop Program: WOMA I, pp. 103–108 (2000)
23. Krasnogor, N.: Studies on the Theory and Design Space of Memetic Algorithms. Ph. D. Thesis, University of the West of England, Bristol (2002)
24. Land, M.W.S.: Evolutionary Algorithms with Local Search for Combinatorial Optimization. Ph. D. Thesis, University of California, San Diego (1998)
25. Merz, P.: Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscape and Effective Search Strategy. Ph. D. Thesis, University of Siegen, Siegen (2000)
26. Merz, P., Freisleben, B.: Fitness Landscape Analysis and Memetic Algorithms for the Quadratic Assignment Problem. IEEE Trans. on Evolutionary Computation 4, 337–352 (2000)
27. Moscato, P.: Memetic Algorithms: A Short Introduction. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 219–234. McGraw-Hill, London (1999)
28. Murata, T., Kaige, S., Ishibuchi, H.: Generalization of Dominance Relation-Based Replacement Rules for Memetic EMO Algorithms. In: GECCO 2003. Lecture Notes in Computer Sciences, vol. 2723, pp. 1234–1245. Springer, Berlin (2003)
29. Ong, Y.S., Keane, A.J.: Meta-Lamarckian Learning in Memetic Algorithms. IEEE Trans. on Evolutionary Computation 8, 99–110 (2004)
30. Ong, Y.S., Lim, M.H., Zhu, N., Wong, K.W.: Classification of Adaptive Memetic Algorithms: A Comparative Study. IEEE Trans. on Systems, Man, and Cybernetics: Part B - Cybernetics 36, 141–152 (2006)
31. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms - A comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998, vol. 1498, pp. 292–301. Springer, Heidelberg (1998)
32. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Trans. on Evolutionary Computation 3, 257–271 (1999)
33. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Trans. on Evolutionary Computation 7, 117–132 (2003)

Part II

Knowledge Infused in Design of Problem-Specific Operators

Solving Time-Tabling Problems Using Evolutionary Algorithms and Heuristics Search

Dipti Srinivasan and Zhang Hua

Department of Electrical & Computer Engineering, National University of Singapore,
10 Kent Ridge Crescent, Singapore 119260
dipti@nus.edu.sg

The university time-tabling problem deals with scheduling classes into available timeslots without violating any constraints of time, venue and personnel. This problem is considered to be of complexity NP and therefore takes a lot of time to solve manually. In this chapter, a new approach to solve these by using a multi-layered-algorithm combining evolutionary algorithms and heuristic search has been attempted. Instead of considering all the constraints equally and in a concurrent manner, different types of constraints are handled by different techniques in separate layers. An evolutionary algorithm first generates sequences of classes, and a heuristic function is then applied to estimate the cost (in terms of number of timeslots needed) to satisfy all the constraints, which is then used by the evolutionary algorithm to rate its individuals. The heuristic function has the advantage of giving results quite quickly. The implementation of this algorithm on actual data obtained from a large university department has been very successful in solving complicated scheduling problems. Indeed, it takes less than thirty seconds to give multiple feasible solutions to this complex real-life time-tabling problem with vast search space (around 10180 possibilities).

1 Introduction

The design of timetables for academic institutions is a challenging problem due to a large number of constraints and a vast amount of classes, students and lectures. Furthermore, there exists no standard solution to this problem as different academic institutions face different variations of this problem each semester. Solving the timetable scheduling has been shown to be a very complex and time-consuming problem. It can take up to a few weeks for university staff to find manually a feasible solution which might not be optimal. Generated timetables are considered feasible if the set of so-called hard constraints are all respected.

Time-tabling problems can be defined as Constraint Satisfaction Problems (CSP). A CSP is defined by a set of variables, a domain for each variable, and a set of constraints [1, 2]. The possible states of a CSP are defined by the values that the variables are assigned to on their domain. The relations between values are defined by constraints. A solution to the CSP is a state in which none of the constraints are violated. In the time-tabling problems different type of constraints are present, such as clashes in time

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 53–69.
springerlink.com

(classes must be scheduled into available timeslots), space (lectures must be held in rooms with sufficient capacity) and personnel (lecturers can not hold two classes at the same time). These constraints can be further divided into two types.

- Hard constraints - constraints that directly determine the feasibility of solutions, a solution to the problem cannot violate any of these constraints. For example, two lectures can not be held in the same room at the same time.
- Soft constraints - constraints that determine the quality of solutions, but not the feasibility of solutions, violations to soft constraints should be avoided

CSP are classified as NP problems [7]. However, checking whether a solution is feasible for CSP can be reduced to polynomial time. A discrete CSP can always be solved by brute force. For this, all the possible combinations of assigned variable values are generated and tested to see if all the constraints are satisfied. Such algorithms can take a very long (but finite) amount of time when the search space is vast. These algorithms waste a large portion of the running time on looking through state trees which already contains constraints violation. That is, if the values of any two of the variables are clashing, then whatever the values of the other variables, a feasible solution can not be found. In order to solve this problem, many different techniques have been tested. The first techniques used mimicked the way humans would solve this problem [11]. Advanced computational methods have also been tried, such as genetic algorithms [10], simulated annealing [4], and neural networks [5]. A modified evolutionary algorithm with multiple context reasoning was also found effective in solving a mid-scale timetabling problem but not able to solve a more complicated problem. [8] Algorithms with have also been presented. Compared to all of the above techniques, advanced software engineering [3] and rule-based language [6] techniques have been found to be superior in terms of the capability of incorporating different types of constraints.

The discussions are organized as follows. Section 2 defines the timetabling problem, section 3 details the algorithm proposed. Section 4 shows the comparative results between the proposed algorithm and conventional evolutionary techniques. Finally, conclusions are drawn in Section 5.

2 The Time-Tabling Problem

2.1 General Form

A typical time-tabling problem can be represented as shown below.

- A set of P classes {C1, C2, C3, , Cp}
- A set of Q available timeslots {T1, T2, T3, TQ}, each of which has its own set of available classrooms
- A set of hard constraints which determines the validity of a solution
- A set of soft constraints which determines the quality of a solution

Given the number of classes and available timeslots, since there are up to Q possible ways to schedule every single class, the number of possible states can be written as

$$N_{states} = Q^P \tag{1}$$

Examples of constraints are given in Table 1.

Table 1. Illustration of constraints

Constraint Domain	illustration	Examples
Time	Time clashing between classes	Affiliated classes must not be held at the same time, since some students need to attend all of these classes
Venue	Venue clashing between classes	No more than one class should be held in any lecture theatre at the same time
Personnel	Personnel (students or teachers) clashing between classes	- None of the students or teachers should be required to attend more than one class at a time -None of the teachers should be required to teach for more than three consecutive hours

2.2 The NUS ECE Time-Tabling Problem

The time-tabling problem in Department of Electrical and Computer Engineering (ECE) at the National University of Singapore was used to test the algorithm developed. The data taken from the year 2001/2002 involves 68 modules, 114 classes and 90 lecturers. All classes have to be scheduled in a total of 65 timeslots.

The ECE time-tabling problem involves the scheduling of three different levels of classes: ECE2 (Year 2), ECE3/4 and M Sc classes (The first year is done in another department). There are 20 ECE2 classes to be scheduled, and the rooms available are only one large lecture theatre and some small tutorial rooms. There are altogether 50 available timeslots, (10 timeslots a day, 5 days a week) some of which are reserved for laboratory sessions. Constraints involved in ECE2 time-tabling problem are mainly hard constraints which include time, venue and teacher constraints.

The scheduling of ECE3/4 classes is more complicated. There are 72 classes to be scheduled into 50 timeslots with limited room resources. As the students have more freedom in choosing their modules, there are more constraints than in the ECE2 timetable. A number of MSc classes are to be scheduled in evening timeslots. 22 3-hour-classes have to be scheduled into the 15 evening timeslots (3 hours, 5 days a week). The MSc. Timetable does not pose much problem on its own, but clashes must be avoided when combining it with the previous timetables. Some of the constraints that are specific to the department are highlighted below.

- Timeslots for lectures shall be assigned with even hours, i.e. 8am, 10am, 12pm, etc
- Any lecturer should not be teaching for more than 3 consecutive hours
- Any lecturer should not be teaching for more than 4 hours in one day

It should be noted that the scheduling processes of different levels of are inter-dependent, since they must share available resources such as lecturer theatres and lecturers. The complexity of the ECE Time-tabling problem can be estimated as follows. Since there

are 20 ECE2 classes to be scheduled into 50 timeslots, 66 ECE3/4 classes to be scheduled into 50 timeslots, and 22 MSc classes to be scheduled into 15 timeslots, the total number of possible timetables is

$$N_{timetables} = 50^{20} \times 50^{66} \times 15^{22} \approx 10^{172} \quad (2)$$

This number is extremely high (as a comparison, only 1018 seconds have passed since the Big-bang) and brute search methods would take billions of years, hence the need for more refined techniques.

A previous research done in 2001 successfully presented a modified evolutionary algorithm that partially solved the ECE time-tabling problem [8]. However it could only successfully schedule all the ECE2 classes and failed to give a feasible solution to the entire ECE time-tabling problem.

3 Proposed Algorithm

3.1 Main Idea

The principle behind the working of the algorithm is to divide the problem into different layers, each layer being solved in a different manner. Furthermore, the problem can be divided into four steps.

1. Generate a time-table of ECE2 classes
2. Based on the result from 1, generate a time-table of ECE3/4 classes
3. Based on the result from 1 and 2, generate a time-table of MSc classes
4. Combine the results from 1, 2 and 3, generate the time-table for all ECE classes

For each of these steps the algorithm uses two layers to find a solution. The first layer uses evolutionary computation to rearrange the set of classes, and the second layer uses a heuristic function to find the best mapping of the rearranged set of classes onto the set of timeslots, as can be seen in Fig 1.

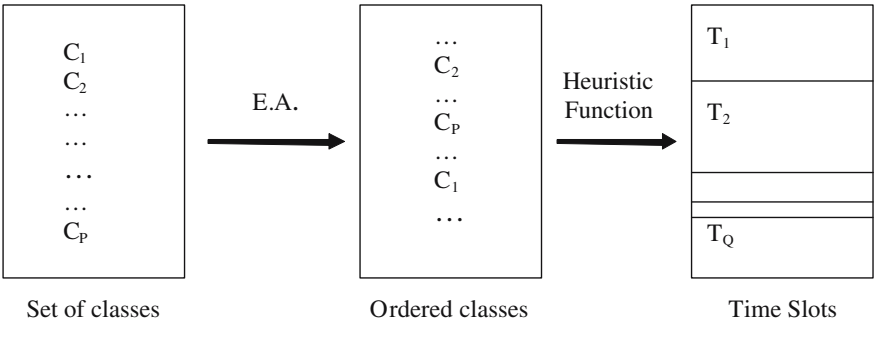


Fig. 1. Architecture of the two-layer algorithm

3.2 The Time-Tabling Process

The time-table scheduling is done as a two step process.

- Generate a ordered sequence of classes
- Break the sequence into ordered segments

The first segment will be scheduled in the first timeslot; classes in the second segment will be scheduled in timeslot two, etc. The heuristic function in step 2, finds the best possible timetable using the order of classes created in step 1. Then, the number of time slots needed, is used as the fitness value for the evolutionary algorithm in step 1. Step 2 is a form of brute search, as all the possible states are tested. However, since the order of the classes is fixed, there are not that many states. In a similar manner, the search space for the evolutionary algorithm is limited to the possibilities of ordering P elements, the size of the search space can be calculated as

$$S_1 = P! \tag{3}$$

The size of the search space for step 2 is the number of possibilities to divide the P classes in Q slots. For this, Q-1 'cuts' are made into P+1 elements (as the first timeslots may be unoccupied). The resulting search space can be given by

$$S_2 = C_{P+1}^{Q-1} \tag{4}$$

3.3 Evolutionary Algorithm

Evolutionary algorithm is used to generate the ordered sequence of classes. Permutation encoding has been chosen as the encoding scheme: a chromosome is represented by a sequence of integers corresponding to the index of the class. The chromosome is directly the order of the classes in the ordered sequence. Selection of parents is done using rank selection, so that even the worst individuals have a (small) chance of being selected.

Mutation is done by swapping a couple of values, and single-point cross-over is used. Since both the mutation rate and crossover rate are chosen to be strictly inferior to 1, some of the parents are replaced intact into the next generation. This is done to ensure that there are always some good individuals in the population.

3.4 First Approach to Heuristics Function

The goal of the heuristic function is to place 'cuts' in the ordered classes which are then mapped sequentially into the timeslots. The sequence of classes can not be modified at this stage: all the classes before the first cut are to be scheduled in timeslot 1; all the classes in between the first and the second cut are to be scheduled in timeslot 2, and so on. At the same time, the heuristics function must ensure that the resulted segmentation must not violate any constraints. Finally, the function returns the minimum number of segments to obtain a constraint-violation-free solution, which will be then used as the fitness for the evolutionary algorithm. The flowchart of the heuristics function is shown in Fig. 2.

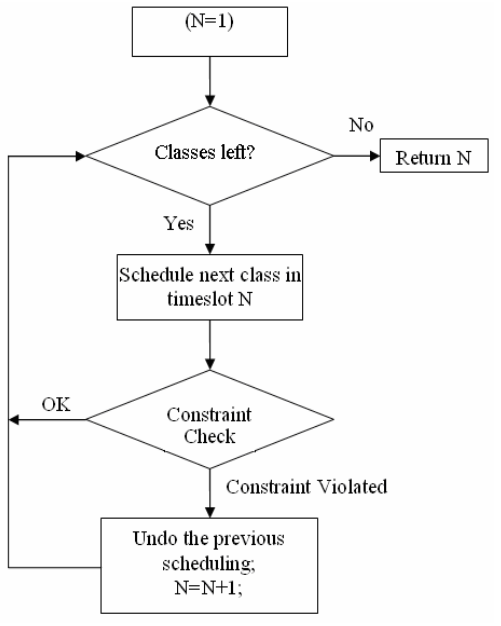


Fig. 2. Flowchart of the heuristics function

The heuristics function always tries to schedule the next class into the current timeslot. A check is then conducted to see whether any constraint is violated. If all the constraints are satisfied, the algorithm will carry on with next class. If any violation of constraints is detected, the algorithm will undo the last scheduling (which caused the violation) and try again with the next timeslot. The whole process ends when there are no more classes to be scheduled. The returned value is the total number of timeslot used.

3.5 Improved Heuristics Function

As previously mentioned, the heuristics function quickly estimates the quality of solutions that can be induced from the input. The heuristics function could not alter the order of the classes; however, although this greatly accelerates the process, it also restricts the behavior of the heuristics function and prevents it from doing the scheduling in more intelligent ways. To further improve the behavior of the heuristics function, the rule set has been revised. The new heuristics function still schedules the classes in the order they are given by the evolutionary algorithm, however, instead of starting the search from the previous time slot the algorithm restart from the first timeslot at each new class. This can be seen in Fig. 3. The resulted heuristics function performs a variation of greedy search, which always follows the way that seems to maximize the outcome at the current stage. Greedy algorithms often perform quite well. They tend to find solutions quickly, although they do not always find the optimal solutions [7].

The new heuristics function will always try to schedule the next class from the sequence into an existing timeslot (non-empty timeslot, with other classes already

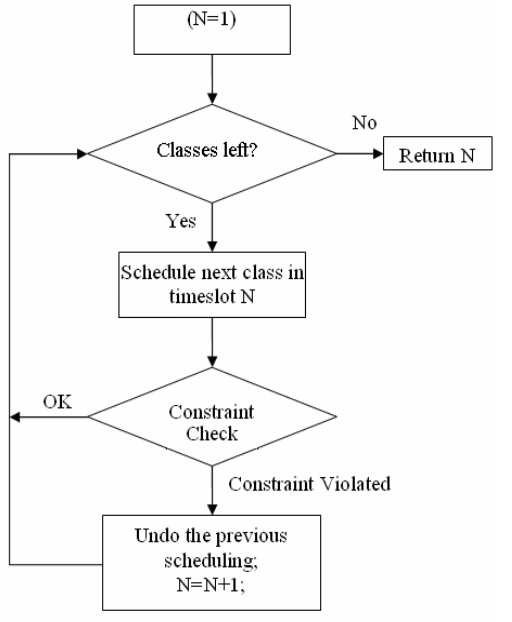


Fig. 3. Flowchart of the improved heuristics function

scheduled). Only when it fails to schedule the next class into any of the existing timeslots, it starts trying empty timeslots. It should be noticed that in the worst case, the improved heuristics function behaves in the same manner as the original heuristics function: the worst case happens when the algorithm fails to schedule the current class into any of the used timeslots; and has to try unused timeslots. This is indeed what happens in the original heuristics where the algorithm only tries to schedule the current class in either the current timeslot or unused timeslots (without trying to schedule the current class in previous timeslots). Therefore, the improved heuristics function is always better than (in the worst case, as good as) the original one. Of course, the improved heuristic function can be more time consuming.

In some situations, the improved heuristics function may give a solution to a random input which is a good as the global optimum. However, this is not generally true because in a complicated time-tabling problem that involves a huge number of classes and constraints, the quality of the solution given by this algorithm is strongly correlated to the input sequence. Therefore, evolutionary algorithm techniques still need to be applied on top of the heuristics function.

4 Application and Simulation Results

4.1 Overview

The proposed algorithm has been tested with an actual time-tabling problem, the scheduling for the ECE department of NUS, where it has shown its capability of

handling complicated real-life instances of time-tabling problems. Simulations done on the two-layered-algorithm with the original heuristics function has shown that it was not efficiently enough to handle the NUS ECE time-tabling problem. However, simulation results have shown that the improved heuristics function performs much better than the original heuristics function.

4.2 Test Methods

The algorithm was implemented using Java, and Microsoft Access (to manage the database of classes, timeslots and constraints) and run on a Windows-based PC with an Intel Pentium4 CPU running at 2.13GHz. Without the knowledge about the global optimal solution, the evolution process is allowed to go until the generation number reaches a preset value in all the test runs. The smallest value of chromosome cost in the last generation was then compared with the number of available timeslots. As defined in the cost function (the heuristics function), the cost of chromosomes represents the number of timeslots needed to schedule all classes. Hence, any cost value less than the number of available timeslots means that this chromosome could be mapped to a valid solution.

The program tries first to do the scheduling of ECE2 classes with the corresponding constraint settings. If a feasible solution is found, the program continues to schedule rest of the classes (ECE3/4 classes) using corresponding constraint settings. Testing on scheduling of MSc. classes are not to be conducted at this stage.

The parameter set (for the evolutionary algorithm process) chosen in this test stage is determined on a trial and error basis. The dataset used is the actual dataset of academic year 2001/2002.

4.3 Scheduling of the ECE2 Classes Using the Original Heuristics Function

The ECE2 time-tabling problem involves 10 modules, each of which have both 2 hours of lecture (conducted in consecutive timeslots) and 1 hour of tutorial, therefore 20 classes are to be scheduled. There are altogether 50 available timeslots. Parameters used in the evolutionary algorithm are given below.

- Population Size = 200
- Maximum Generation = 100
- Crossover Rate = 0.95
- Mutation Rate = 0.005

Five test runs are conducted. The program takes about 3 seconds to complete the entire evolution process in each run. In all test runs, the algorithm was able to successfully find feasible solutions. In fact, all test runs found a solution that needed 36 timeslots to schedule all of the ECE2 classes, which is much smaller than the actual number of available timeslots which in out case is 50.

The following figures are obtained from one of the test runs. Fig. 4 shows the performance of the algorithm in terms of minimal cost of chromosomes across generations. As the best cost converges, the percentage of chromosomes with the best value starts to converge rapidly to 100% as can be seen in Fig. 5.

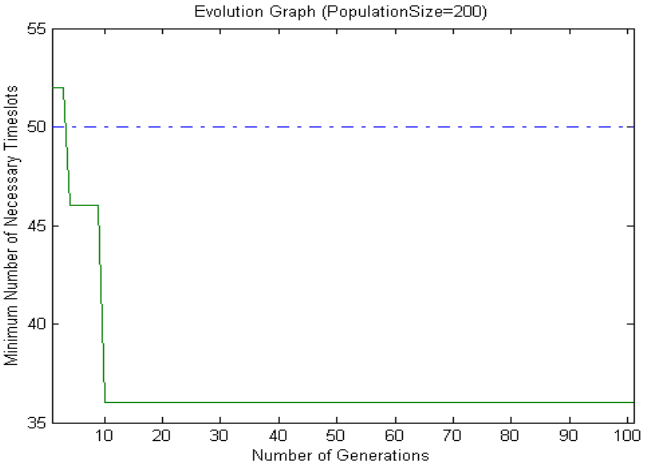


Fig. 4. Performance plot in scheduling of ECE2 classes - Minimal cost of chromosomes versus Number of generations

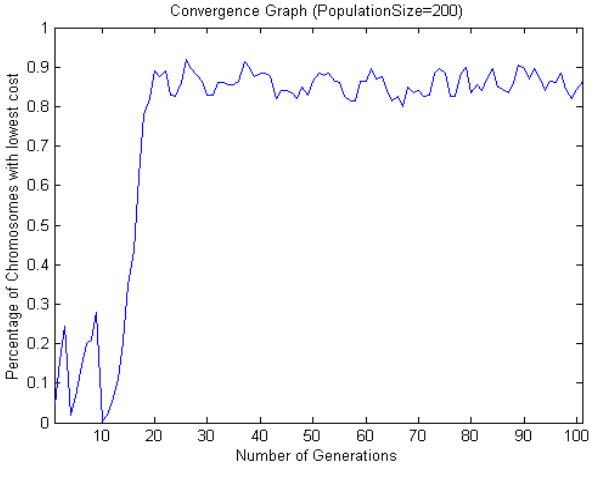


Fig. 5. Convergence plot in scheduling of ECE2 classes Percentage of chromosomes with smallest cost versus Number of generations

4.4 Scheduling of ECE3/4 Classes

Compared with the scheduling of ECE2 classes, the scheduling of ECE3/4 classes is more complicated. The scheduling involves 33 modules, each of which has both 2 hours of lecture and 1 hour of tutorial to be scheduled resulting in 66 classes. There are altogether 50 available timeslots. Furthermore, there are more constraints than in the ECE2 set. Parameters in the evolutionary algorithm are modified as shown below.

- Population Size = 500
- Maximum Generation = 300
- Crossover Rate = 0.95
- Mutation Rate = 0.005

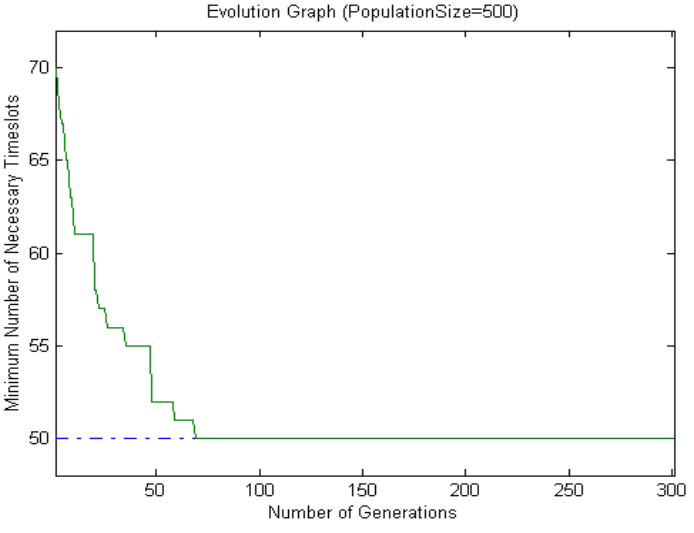


Fig. 6. Performance plot in scheduling of ECE3/4 classes - Minimal cost of chromosomes versus Number of generations

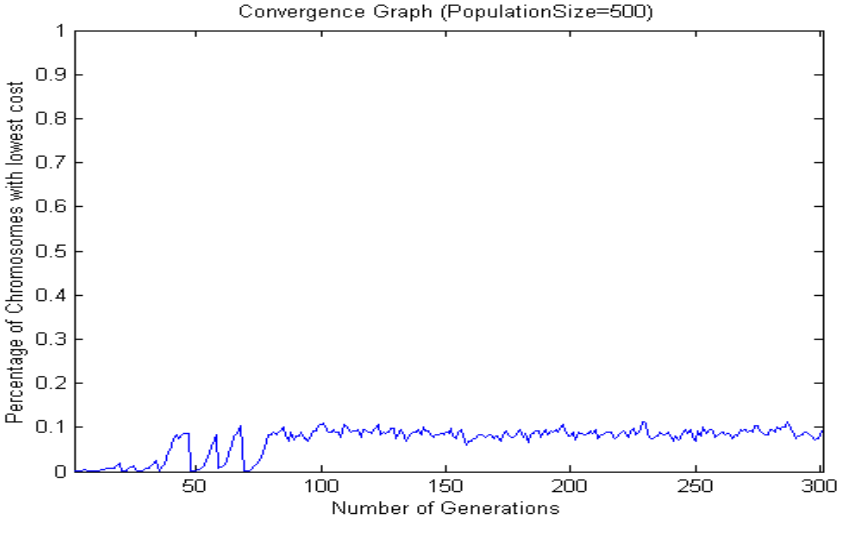


Fig. 7. Performance plot in scheduling of ECE3/4 classes - Minimal cost of chromosomes versus Number of generations

Five test runs are conducted. The program takes about 50 seconds to complete each test run. In all the test runs, the program was able to present feasible solutions, which fits all of the ECE3/4 classes into the 50 available timeslots. Fig. 6 shows the performance of the program in one of the successful test run.

Compare to the plot in the ECE2 test run (Fig. 5), a much smaller portion (around 10%) of the chromosomes actually reach the smallest chromosome cost as can be seen on Fig. 7.

The algorithm using the original heuristic function is efficient enough to do the scheduling of ECE2 classes in a very short time (only a few seconds for a run). The algorithm has also shown its ability to do the scheduling of ECE3/4 classes. In most of test runs, the algorithm was able to find a feasible solution. However, the running time is too long (nearly one minute). It should be foreseen that the algorithm may fail to give feasible solutions in practical time, when facing more complicated problems. Therefore, the overall efficiency of the algorithm should be further improved, and the use of the improved heuristic function is justified.

4.5 Simulation Results with the Improved Heuristics Function

The simulations were repeated with the improved heuristics function. All simulations were conducted on the same platform, using the same dataset. In the scheduling of ECE2 classes, the following parameters (applied in the evolutionary algorithm) were used.

- Population Size = 50
- Maximum Generation = 20
- Crossover Rate = 0.95
- Mutation Rate = 0.005

Five test runs were conducted. On the test platform, the running time of the algorithm was less than 1 second. In all the test runs, the algorithm successfully found feasible solutions that needed 36 timeslots to schedule all of the classes. Simulation results have shown that the improved heuristics function has a very good response to random inputs. In fact, even from the first generation of chromosomes, it successfully finds a solution that is as good as the final solution. This can raise the question of the necessity of the evolutionary algorithm. Moreover, the evolutionary algorithm is very successful in improving the convergence of chromosome cost values. Most chromosomes' cost value quickly converges to the smallest value during the evolution.

A feasible solution was retained, and the test was continued with the scheduling of MSc classes, in which the following parameters were used.

- Population Size = 10
- Maximum Generation = 5
- Crossover Rate = 0.95
- Mutation Rate = 0.005

A total of five test runs were conducted, each taking less than one second to complete. In all test runs, the program successfully scheduled 20 MSc classes into 12 timeslots

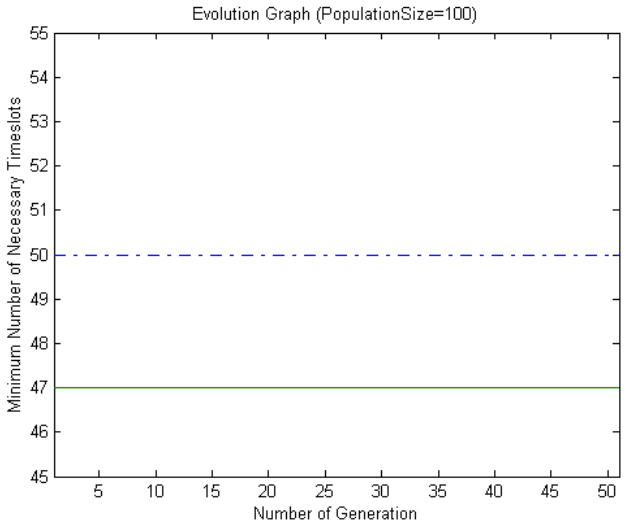


Fig. 8. Performance plot in scheduling of ECE3/4 classes - Minimal cost of chromosomes versus Number of generations

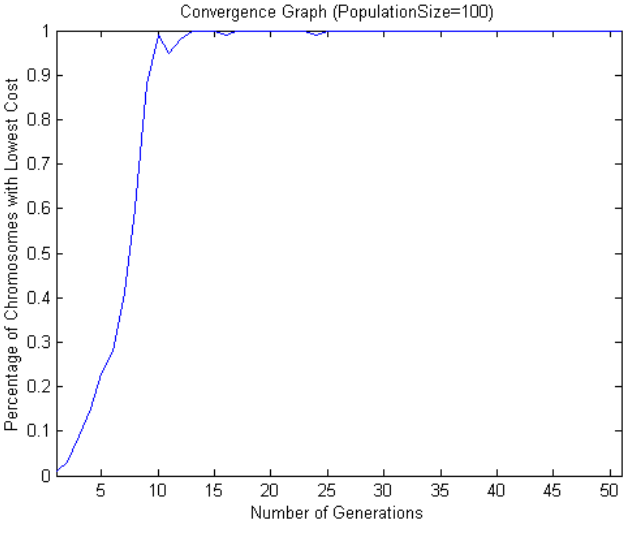


Fig. 9. Convergence plot in scheduling of ECE2 classes Percentage of chromosomes with smallest cost versus Number of generations

without violation of any constraint. The experimental results shows that, similar to the scheduling of ECE2 classes, the algorithm was able to find feasible solutions which are as good as the final solution from the first generation onwards. In the last generation of evolution, all chromosomes' cost converges to the best value. Once more, the necessity of the evolutionary algorithm is doubtful.

Testing on scheduling of ECE3/4 was conducted, based on the combined result of a feasible schedule of MSc classes, and the schedule of ECE2 classes found before. In the scheduling of ECE3/4 classes, the following parameters were used.

- Population Size = 100
- Maximum Generation = 50
- Crossover Rate = 0.95
- Mutation Rate = 0.005

A total of 5 test runs were conducted, each taking about 10 seconds to complete. In all of the test runs, the program successfully found feasible solutions that needed 47 timeslots to schedule all classes. Performance plots of one of the test runs are shown in Fig. 8 and 9.

The improved heuristics function performs satisfactorily even with randomly generated input sequences. On top of it, the evolutionary algorithm successfully helps in increasing the overall fitness of the whole population. More than 90% of the chromosomes have their cost value quickly converge to the lowest value. This behavior can be helpful as it can give the user different timetables of equal quality, and the user can then chose that best fits his needs.

4.5.1 Performance of the Improved Heuristics Function without Evolutionary Algorithm

It has been shown in previous sections that the improved heuristics function has a very good response towards random inputs (without the help of evolutionary algorithm). In this section, we shall consider this characteristic in more detail, by adding more complexity to the problem, so random search is made more difficult. In previous simulations, ECE2 and ECE3/4 classes were scheduled separately by the algorithm (though the process is inter-dependent). However, in this section, the algorithm performs the

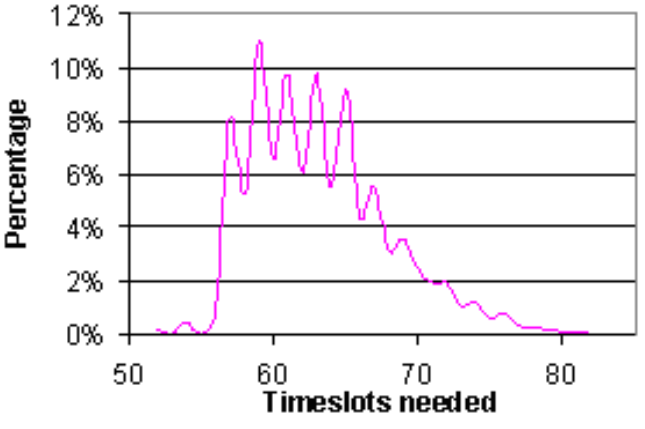


Fig. 10. Performance of the improved heuristics function with random inputs

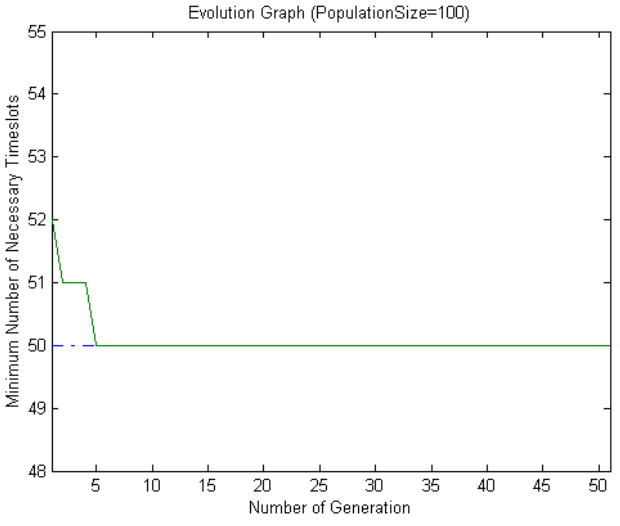


Fig. 11. Performance plot in scheduling of ECE2 and ECE3/4 classes - Minimal cost of chromosomes versus Number of generations

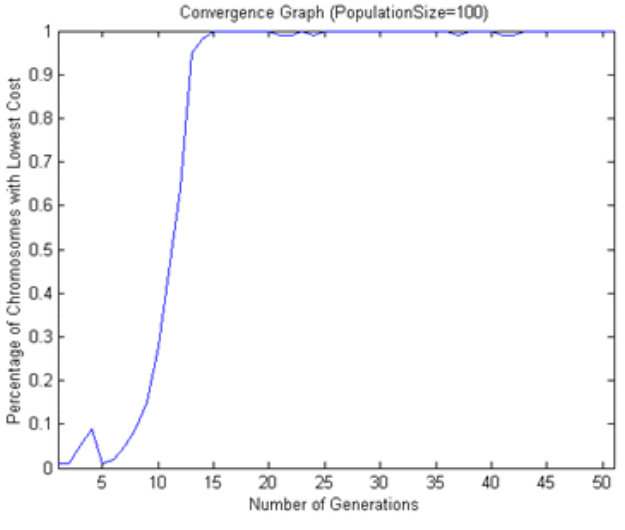


Fig. 12. Convergence plot in scheduling of ECE2 andECE3/4 classes Percentage of chromosomes with smallest cost versus Number of generations

scheduling of both ECE2 and ECE3/4 classes together. It is easy to see that such an arrangement makes the scheduling process much more complicated than previous ones. Moreover, one more constraint has been added into the time-tabling problem to make it more complicated and follow real-world scenario. Timeslots between 1400hrs and

1700hrs on Fridays are reserved (Friday afternoons). No classes are to be scheduled into these timeslots.

To consider the performance of the improved heuristics function on its own, the evolutionary algorithm process was removed from the overall algorithm. In the simulation run, 10000 random sequences were used as inputs for the improved heuristics function. Fig. 10 shows the simulation results.

From the graph, we can conclude that when the target problem gets more complicated, the heuristics function by itself is not effective enough to solve the problem. The heuristics function could only manage to schedule these classes into more timeslots (ranging from 51 to 76) than the actual number of available timeslots (50), which is not feasible. In practice, it failed to schedule all ECE2 and ECE3/4 classes into available timeslots after trying all of the 10000 random inputs. Now, the improved heuristics function is tested with the evolutionary algorithm giving the input sequences. Working together, the problem was solved easily. The following parameters were used in the simulation.

- Population Size = 100
- Maximum Generation = 50
- Crossover Rate = 0.95
- Mutation Rate = 0.005

The algorithm took 16 seconds to complete the process and gave feasible solutions that needed 50 timeslots to schedule all ECE2 and ECE3/4 classes. As can been seen from Fig. 11, a feasible solution (50 timeslots) was found in 5 generation, so only 500 individuals were needed, whereas with random inputs, no solution was found even with 20 times as more inputs (10000 random inputs).

4.5.2 Performance Comparison between Heuristic Functions

Table 2 shows the effectiveness of the improved heuristics function when compared with the original heuristic function. Obviously, the multi-layered-algorithm performs much better with the improved heuristics function than with the original one.

Table 2. Comparison between algorithms with different heuristics function

Heuristics function used with EA	Original heuristics function	Improved heuristics function
Scheduling of ECE2 classes	Passed	Passed
Time needed for ECE2 classes	3 seconds	$<$ 1 second
Timeslots needed for ECE2 classes	36	36
Scheduling of ECE3/4 classes	Passed	Passed
Time needed for ECE3/4 classes	50 seconds	10 seconds
Timeslots needed for ECE3/4 classes	50	47
Scheduling of MSc classes	N/A (not conducted)	Passed
Scheduling of ECE3/4 classes in one step	N/A (not conducted)	Passed

Table 3. Comparison of results

Heuristics function used with EA	Original heuristics function	Improved heuristics function
Scheduling of ECE2 classes	Passed	Passed
Time needed for ECE2 classes	60 seconds	$<$ 1 second
Timeslots needed for ECE2 classes	45	36
Scheduling of ECE3/4 classes	Failed	Passed
Time needed for ECE3/4 classes	N/A (Failed)	10 seconds
Timeslots needed for ECE3/4 classes	N/A (Failed)	47
Scheduling of MSc classes	N/A (not conducted)	Passed

4.5.3 Comparison of Different Approaches

A previous research on automating the time-tabling process for ECE department was done using the same data (ECE classes) to test it [8]. This approach used only a modified evolutionary algorithm (without any heuristics). Since the same data set is used, a comparison can be done with the algorithm detailed above, and the results are shown in Table 3. Both algorithms successfully performed the scheduling of the ECE2 classes. However, the pure evolutionary approach failed to schedule the ECE3/4 classes into available timeslots and hence failed to give a solution to the entire ECE time-tabling problem.

The algorithm presented with improved heuristic approach is superior to the previous approach when facing an identical problem dataset.

5 Conclusion

In this chapter, the possibility of solving university time-tabling problems with an algorithm that combines evolutionary computational techniques and heuristics search has been presented. Two implementations with different heuristics functions have been detailed and tested. Experimental results on these algorithms with real data from the NUS ECE time-tabling problem have shown their effectiveness in solving a practical time-tabling problem. The combine use of evolutionary algorithms and heuristic functions has shown to be much more successful than evolutionary algorithms alone or heuristic functions alone (using random inputs). The algorithm has been tested against a previous algorithm on the same dataset and its superiority both in speed and quality of the solutions has been shown.

Future development may focus on applying the similar algorithm to other scheduling problems, such as cargo scheduling in ports or train scheduling. In this algorithm, all constrains were treated as hard constraints, so all the solutions found were of high quality. However, end-users might want to specify many soft constraints which might saturate this algorithm. A multi-objective approach could then be taken, and soft constraints could then be conditional (constraint A or B has to be respected).

References

1. Fellini, R., Kokkolaras, M., Panos, P.Y., Perez-Duarte.: A Platform Selection Under Performance Loss Constraints in Optimal Design of Product Families. In: Proceedings of 2002 Design Engineering Technical Conferences and Computer and Information in Engineering Conference, Montreal, Canada, September 29-October 2 (2002)
2. Robertson, D., Ulrich, K.: Planning Product Platforms. Sloan Management Review 39(4), 19–31 (1998)
3. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Englewood Cliffs (1995)
4. Torrens, M.: Constraint Satisfaction Problems (2002), `http://liawww.epfl.ch/~torrens/Project/project/node10.html`
5. Lee, J., Fanjiang, Y.-Y., Lai, L.F.: A Software Engineering Approach to University Timetabling. In: Proceedings International Symposium on Multimedia Software Engineering, pp. 124–131 (2000)
6. Sheung, J., Fan, A., Tang, A.: Time tabling using genetic algorithm and simulated annealing. In: TENCON 1993. Proceedings of IEEE Region 10 Conference on Computer, Communication, Control and Power Engineering, vol. 1, pp. 448–451 (October 1993)
7. Yu, T.L.: Time-table scheduling using Neural Network Algorithms, Neural Networks. In: 1990 IJCNN International Joint Conference on 1990, vol.1, pp. 279 –284 (June 1990)
8. Solotorevsky, G., Gudes, E., Meisels, A.: RAPS: a rule-based language for specifying resource allocation and time-tabling problems. IEEE Transactions on Knowledge and Data Engineering 6(5), 681–697 (1994)
9. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. MIT Press, Cambridge (1990)
10. Srinivasan, D., Seow, T.H., Xu, J.X.: Automated time table generation using multiple context reasoning for university modules. In: Proceedings of the 2002 Congress on Evolutionary Computation, vol. 2 (2002)
11. Schaerf, A.: Local search techniques for large high-school timetabling problems. IEEE Transactions on Systems, Man, and Cybernetics- Part A: Systems and Humans 29(4), 368–377 (1999)
12. Colorni, A., Dorigo, M., Maniezzo, V.: Metaheuristics for high school timetabling. Computational Optimization and Applications 9(3), 275–298 (1998)
13. Schmidt, G., Strohlein, T.: Timetable construction-An annotated bibliography. Computer Journal 23(4), 307–316 (1979)

An Efficient Genetic Algorithm with Uniform Crossover for the Multi-Objective Airport Gate Assignment Problem

Xiao-Bing Hu[1] and Ezequiel Di Paolo[2]

[1] Centre for Computational Neuroscience and Robotics, University of Sussex
xiaobing.hu@sussex.ac.uk
[2] Centre for Computational Neuroscience and Robotics, University of Sussex
ezequiel@sussex.ac.uk

Genetic Algorithms (GAs) have a good potential of solving the Gate Assignment Problem (GAP) at airport terminals, and the design of feasible and efficient evolutionary operators, particularly, the crossover operator, is crucial to successful implementations. This paper reports an application of GAs to the multi-objective GAP. The relative positions between aircraft rather than their absolute positions in the queues to gates are used to construct chromosomes in a novel encoding scheme, and a new uniform crossover operator, free of feasibility problems, is then proposed, which is effective and efficient to identify, inherit and protect useful common sub-queues to gates during evolution. Extensive simulation studies illustrate the advantages of the proposed GA scheme with uniform crossover operator.

1 Introduction

As a major issue in Air Traffic Control (ATC) operations, the Gate Assignment Problem (GAP) at airport terminals aims to assign aircraft to terminal gates to meet operational requirements while minimizing both inconveniences to passengers and operating costs of airports and airlines. The term gate is used to designate not only the facility through which passengers pass to board or leave an aircraft but also the parking positions used for servicing a single aircraft. These station operations usually account for a smaller part of the overall cost of an airlines operations than the flight operations themselves. However, they can have a major impact on the efficiency with which the flight schedules are maintained and on the level of passenger satisfaction with the service [1], [2].

Most airline companies create monthly or quarterly Master Flight Schedules (MFSs) containing flight numbers and along with the corresponding arrival and departing times. The ground controllers use the MFSs to examine the capacity of gates to accommodate proposed schedules. There are several considerations that can bear on the decisions, such as aircraft size and servicing requirements, idle time of gates, flight crew and aircraft rotation, passenger walking distance, baggage transport distance, ramp congestion, aircraft waiting time, and use of remote parking stands. In the past few decades, many optimization methods have been reported to improve the gate assignment operation at

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 71–89.
springerlink.com

airport terminals by focusing on one or two of the above considerations. For instance, passenger walking distance has been widely studied in the GAP research, and methods such as branch-and-bound algorithms [2], [3], integer programming [4], linear programming [5], expert systems [6], [7], heuristic methods [1], tabu search algorithms [8] and various hybrid methods [9], [10] were reported to minimize this distance. Baggage transport distance has been relatively less discussed in the GAP literature [1], [11]-[13], but the algorithms developed to solve the minimum passenger walking distance GAP can be easily extended to the case where baggage transport distance needs to be considered [1]. During the peak hours, it often happens that, particularly at hub airports, the number of aircraft waiting to dwell exceeds the number of available gates. In this case, aircraft waiting time on the apron should also be minimized [9], [14], [15]. The gate idle time is a criterion often used to assess the efficiency of using gate capacity [16]. However, the multi-objective GAP is relatively less discussed in literature. Reference [17] reported some interesting results, where passenger walking distance and passenger waiting time were both considered, the GAP was modelled as a zero-one integer program, and a hybrid method was developed based on the weighting method, the column approach, the simplex method and the branch-and-bound technique.

As large-scale parallel stochastic search and optimization algorithms, GAs have a good potential for solving NP-hard problems such as the GAP. For instance, reference [15] developed a GA to minimize the delayed time during the gates reassignment process, but the walking distance was not included. Reference [16] proposed a unified framework to specifically treat idle time of gates in the previous GAP models, and then developed a problem-specific knowledge-based GA. This paper aims to shed a little more light on how to design efficient GAs for the multi-objective GAP (MOGAP), where passenger walking distance, baggage transport distance, and aircraft waiting time on the apron need to be considered simultaneously. The design of highly efficient evolutionary operators, i.e., mutation and crossover, is crucial to successful applications of GAs to the GAP. Basically, mutation can increase the diversity of chromosomes in GAs to exploit the solution space, while crossover, in order to help GAs to converge to optima, needs to identify, inherit and protect good common genes shared by chromosomes, and at the same time to recombine non-common genes. Due to the stochastic nature of mutation and crossover, it is not an easy task to design efficient evolutionary operators free of the feasibility problem. For instance, infeasible solutions were particularly discussed in [16], where the mutation operator, rather than introducing diversity, was used to repair infeasible chromosomes generated by a conventional one-point split crossover operator.

This paper attempts to develop an infeasibility-free GA for the multi-objective GAP. Since crossover is often a main source of infeasible chromosomes, effort is particularly put on the design of a novel uniform crossover operator free of the feasibility problem. To this end, the relative positions between aircraft rather than the absolute positions of aircraft in the queues to gates is used to construct chromosomes in the new GA. As a result of the new uniform crossover operator, the design of mutation operator can concentrate on the original purpose of diversifying chromosomes.

2 Problem Formulation of the MOGAP

As mentioned before, there are quite a few different considerations in the GAP, and the MOGAP in this paper will focus on three of them: passenger walking distance, baggage transport distance, and aircraft waiting time on the apron. Passenger walking distance has a direct impact on the customer satisfaction. The typical walking distances in airports considered are: (I) the distance from check-in to gates for embarking or originating passengers, (II) the distance from gates to baggage claim areas (check-out) for disembarking or destination passengers, and (III) the distances from gate to gate for transfer or connecting passengers. Baggage transport distance occurs when baggage is transferred between aircraft and baggage claim areas. Basically, these distances can be reduced by improving the method by which scheduled flights are assigned to the airport terminal gates. Aircraft waiting time on the apron is the difference between the planned entering time to gates and the allocated entering time to gates. Due to the shortage of gates at peak hours, some scheduled aircraft have to wait extra time on the apron, which could end up with delayed departure and even cause passengers miss connection flights. Although this kind of ground delay is more tolerable than airborne delay in terms of safety and costs, it largely affects the customer satisfaction. Besides, aircraft waiting time can help address another big issue in the GAP: the efficiency of using gate capacity, which is often represented by how even the distribution of idle times is. In the minimum distance GAP, some special constraints have to be included in order to avoid most aircraft being assigned to a same single gate, which however can automatically be ensured by minimizing aircraft waiting time. Therefore, in the MOGAP, we will construct an objective function by combining the above three considerations. A simple way to conduct gate assignment is the first-come-first-served (FCFS) principle according to the planned entering time to gates, but the result is usually not optimal or even not near-optimal, because the FCFS principle does not take into account the layout of airport terminals. Even for a queue at a single gate, the FCFS principle is not the first option, mainly because different aircraft may have different ground time and different number of passengers. Obviously, putting ahead an aircraft with more passengers and less ground time could bring benefits, even if its planned entering time is later. Fig.1 gives a simple illustration of the MOGAP.

Suppose N_{AC} aircraft need to be assigned to N_G gates during a given time period $[T_S,T_E]$. Let P_i and G_i denote the planned entering time to gates and the ground time of the ith aircraft in the original set of aircraft under consideration, respectively. Assume P_i and G_i to be known in advance. In this paper, the planned entering time to gates for arrival aircraft is assumed to be the scheduled arrival time to the airport (A_i), and the planned entering time for departing aircraft is the scheduled departure time (D_i) minus the ground time, i.e., $P_i = D_i - G_i$. Let Q_g denote the queue at gate g, $Q_g(j)$ is the jth aircraft in Q_g, $g = 1,\ldots,N_G, j = 1,\ldots,H_g$, and H_g is the number of aircraft in Q_g satisfying

$$\sum_{g=1}^{N_G} H_g = N_{AC} \tag{1}$$

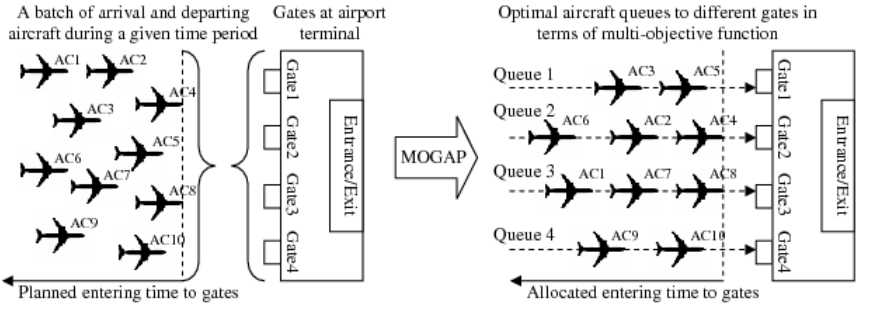


Fig. 1. Illustration of the MOGAP

$Q_g(j) = i$ means the ith aircraft in the original set is assigned as the jth aircraft to dwell at gate g. The allocated entering time to gates (E_i) for the ith aircraft in the original set can then be calculated as

$$E_{Q_g(j)} = \begin{cases} P_{Q_g(j)}, & j = 1 \\ max(P_{Q_g(j)}, E_{Q_g(j-1)} + G_{Q_g(j-1)}), & j > 1 \end{cases} j = 1,\ldots,H_g, g = 1,\ldots,N_G \qquad (2)$$

The waiting time on the apron for the ith aircraft in the original set is

$$W_i = E_i - P_i, i = 1,\ldots,N_{AC}. \qquad (3)$$

For the sake of simplicity of modeling, besides the N_G real gates, the entrance/exit of the airport terminal is usually considered as a dummy gate (e.g., see [1]), and we call it gate $N_G + 1$ in this paper. Associated with this dummy gate $N_G + 1$, we introduce a dummy aircraft $N_{AC} + 1$. Of course there is no real aircraft queue for this dummy gate, except the dummy aircraft which dwells at the dummy gate all time.

Three data matrices, $M_p \in R^{(N_{AC}+1)\times(N_{AC}+1)}$, $M_{PWD} \in R^{(N_G+1)\times(N_G+1)}$, and $M_{BTD} \in R^{(N_G+1)\times(N_G+1)}$, are used to record the number of passengers transferred between aircraft, passenger walking distances between gates, and baggage transport distances between gates, respectively. Given $i \le N_{AC}$ and $j \le N_{AC}$, the value of $M_P(i,j)$ is the number of passengers transferred from aircraft i to aircraft j, $M_P(i,N_A+1)$ records the number of arriving passengers from aircraft i to exit, i.e., the dummy aircraft $N_A + 1$, and $M_P(N_A+1,j)$ the number of departing passengers from entrance to aircraft j. For those passengers who just pass by the airport with a long-haul aircraft, we assume they do not leave the aircraft when the aircraft stops at the airport. Therefore, we always have $M_P(i,i) = 0$ for $i = 1,\ldots,N_{AC}+1$. $M_{PWD}(i,j)$ are the passengers walking distance from gate i to gate j, and $M_{BTD}(i,j)$ the baggage transport distance from gate i to gate j. Although $M_{PWD}(N_G+1,N_G+1) = 0$, we do not have $M_{PWD}(i,i) \neq 0, i = 1,\ldots,N_G$, because, even though passengers transfer between two aircraft which are successively assigned to the same gate, they still need to leave the first aircraft and wait in a certain terminal lounge before they can board the second aircraft. Similarly, for $M_{BTD}(i,i)$ one has $M_{BTD}(N_G+1,N_G+1) = 0$, but $M_{BTD}(i,i) \neq 0$. Besides these three matrices, we still need a data vector: $V_G = [v_1,\ldots,v_{N_{AC}+1}]$, where $1 \le v_i \le N_G + 1$ indicates that the

ith aircraft in the original set is assigned to gate v_i, and $v_{N_{AC}+1} \equiv N_G + 1$ means the dummy aircraft $N_{AC}+1$ is always assigned to the dummy gate $N_G + 1$.

Now we can calculate the total passenger walking distance (TPWD), the total baggage transferring distance (TBTD), and the total passenger waiting time (TPWT) as

$$J_{TPWD} = \sum_{g=1}^{N_G+1} \sum_{j=1}^{H_g} \sum_{i=1}^{N_{AC}+1} M_P(Q_g(j), i) M_{PWD}(g, v_i), \tag{4}$$

$$J_{TBTD} = \sum_{g=1}^{N_G+1} \sum_{j=1}^{H_g} \sum_{i=1}^{N_{AC}+1} M_P(Q_g(j), i) M_{BTD}(g, v_i), \tag{5}$$

$$J_{TPWT} = \sum_{i=1}^{N_{AC}} W_i \sum_{j=1}^{N_{AC}+1} (M_P(i, j) + M_P(j, i)), \tag{6}$$

respectively. In the MOGAP of this paper, the following weighted objective function is used to cover these three aspects:

$$J_{MOGAP} = \alpha J_{TPWD} + \beta J_{TBTD} + (1 - \alpha - \beta)\phi J_{TPWT}, \tag{7}$$

where α and β are tuneable weights to adjust the contributions of TPWD, TBTD and TPWT,

$$\alpha + \beta \leq 1, 0 \leq \alpha \leq 1, 0 \leq \beta \leq 1, \tag{8}$$

and ϕ is a system parameter to make the waiting time comparable to the distances. In this paper, the distances are measured in meters, and the times measured in minutes. Assuming the average passenger walking speed is 3km/h, then 1 minute waiting time for a passenger can be considered as 50 meters extra walking distance for him/her. In this paper, we take the half, i.e., set $\phi = 25$ because we assume that for passengers walking is physically more uncomfortable than waiting.

The MOGAP can now be mathematically formulated as a minimization problem:

$$min_{Q_1,\ldots,Q_{N_G}} J_{MOGAP}, \tag{9}$$

subject to (1) to (8). Clearly, how to assign aircraft to different gates to form N_G queues and how to organize the order of aircraft in each queue compose a solution, i.e., $Q_1, \ldots, Q_{N_G}$, to the minimization problem (9). Unlike other existing GAP models, the above formulation of the MOGAP needs no binary variables due to the usage of $Q_1, \ldots, Q_{N_G}$.

3 A GA with Uniform Crossover for the MOGAP

In this section we will report a new GA with uniform crossover for the MOGAP (The proposed new GA will be denoted as GAUC hereafter). For comparative purposes, we will also discuss two other GAs, particularly to compare their chromosome structures and crossover operators.

3.1 New Chromosome Structure

Basically, grouping aircraft according to gates, i.e., assigning aircraft to different gates, is one of the most important steps in the GAP, because it has direct influence on both walking distances and idle times of gates. If aircraft waiting time on the apron is not under consideration, then optimal grouping plus the FCFS principle can produce the best way of utilizing the gates at airport terminals. The GA proposed in [16] was designed based on aircraft grouping information. The structure of its chromosome is illustrated in Fig.2.(b), where a gene $C(i) = g$ means the ith aircraft in the original set of aircraft is assigned to dwell at gate g, in other words, gate g is assigned to aircraft i. Hereafter, we call aircraft grouping as gate assignment. Clearly, the GA based on gate assignment is not concerned about the order of aircraft in the queue to each gate, which is crucial to the minimization of aircraft waiting time on the apron.

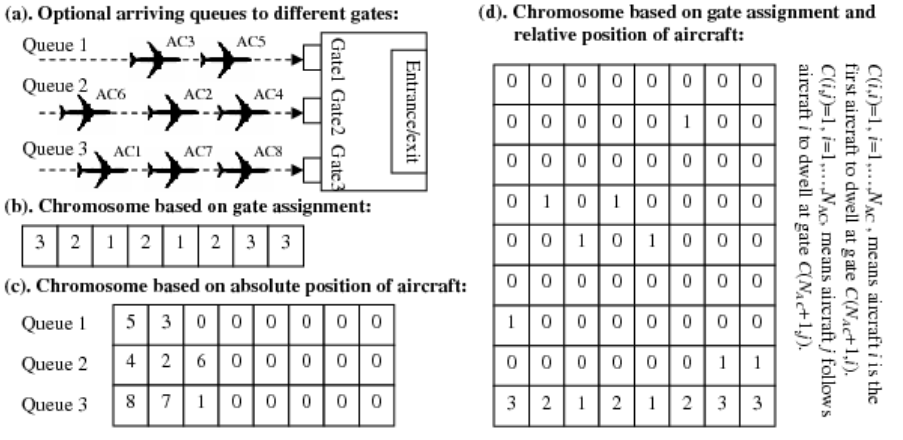


Fig. 2. Chromosome structures (see text)

As discussed before, different aircraft may have different number of passengers and different ground time, and therefore, switching the positions of some aircraft in a FCFS-principle-based queue could reduce the total passenger waiting time, which is another criterion to assess the level of customer satisfaction with the service. The GA proposed for the arriving sequencing and scheduling problem in [18] can be modified and extended to handle the position switching in the GAP. The chromosome structure is illustrated in Fig.2.(c), where one can see the absolute positions of aircraft in queues to gates are used to construct chromosomes, i.e., a gene $C(g, j) = i$ means the ith aircraft in the original set of aircraft is assigned as the jth aircraft to dwell to gate g. Apparently, the underlying physical meaning of a chromosome, i.e., queues to gates, is expressed in a straightforward way by the absolute-position-based structure. However, it is difficult to carry out genetic operations on common genes, i.e., to identify, to inherit and to protect them, in these chromosomes based on absolute position of aircraft.

Basically, common genes should be defined as those sub-structures or sections which are shared by some chromosomes and play an important role in evolving the fitness of

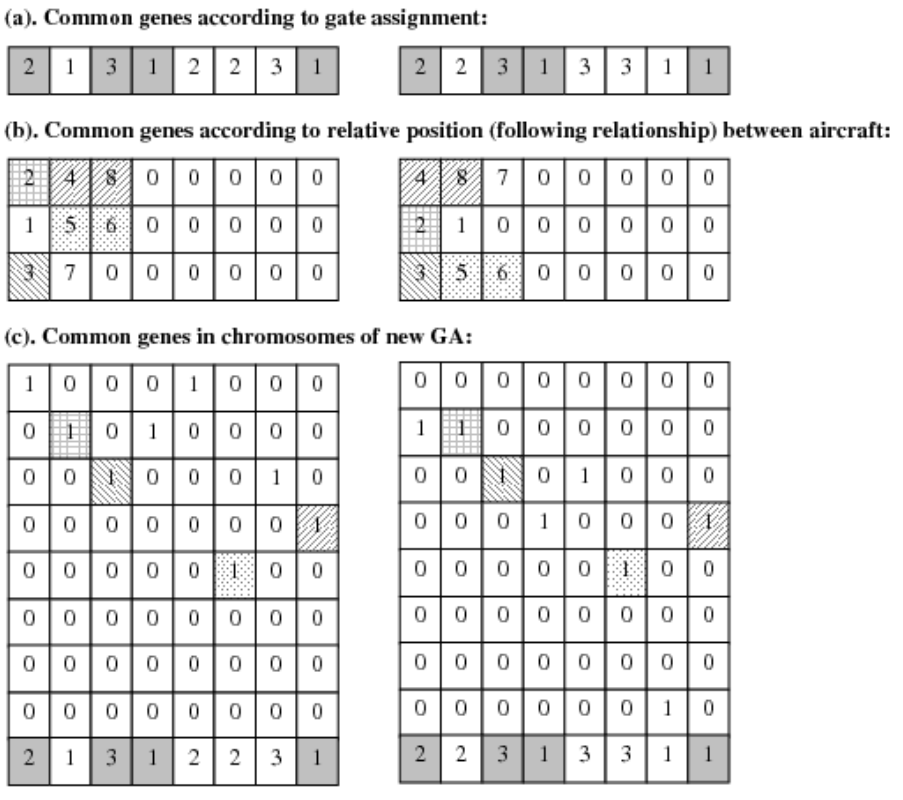


Fig. 3. Two definitions of common genes in the MOGAP

chromosomes. In the MOGAP, walking distances are sensitive to gate assignment, while relative position between aircraft in queues affects aircraft waiting time. In the evolutionary process, if many fit chromosomes assign a same gate to a same aircraft, and/or apply the same order to a same pair of successive aircraft in a queue, it is likely that this gate assignment and/or this relative position will also appear in the fittest chromosomes. Therefore, the same gate assignment and the same relative position are used to define common genes in the MOGAP. Fig.3.(a) and Fig.3.(b) illustrate these two definitions. From Fig.3.(a) one can see that the gate assignment based chromosome structure makes it very easy to identify common genes, i.e., if $C_1(i) = C_2(i)$, then these two genes are common genes. However, this structure has no information for identifying common relative position between aircraft in queues. An absolute position based chromosome structure has sufficient information for identifying both common gate assignment and common relative position, but unfortunately extra computationally expensive procedures are required.

The GAUC introduced in this paper will use the information of both gate assignment and relative position, not absolute position, to construct chromosomes. As illustrated in Fig.2.(d), a chromosome in the GAUC is a matrix with a dimension of $(N_{AC}+1) \times N_{AC}$,

where the first $N_{AC} \times N_{AC}$ genes, i.e., $C(i,j), i = 1,\ldots,N_{AC}, j = 1,\ldots,N_{AC}$, record relative positions between aircraft in queues, and the last N_{AC} genes, i.e., $C(N_{AC}+1,j), j = 1,\ldots,N_{AC}$, record gate assignments. If $C(i,i) = 1$ and $C(N_{AC}+1,i) = g$, this means the ith aircraft in the original set of aircraft is assigned as the first aircraft to dwell at gate g; If $C(i,j) = 1$ and $C(N_{AC}+1,j) = g$, this means aircraft j is assigned to follow aircraft i to dwell at gate g. As illustrated in Fig.3.(c), with this new structure, common genes under both definitions can be easily identified: If $C_1(i,j)\&C_2(i,j) = 1$, then they are common relative position; If $C_1(N_{AC}+1,j) = C_2(N_{AC}+1,j)$, then they are common gate assignment.

Feasibility is a crucial issue in the design of a chromosome structure. For the structure based on gate assignment, if aircraft waiting time is allowed, which is the case in the MOGAP, then there is no feasibility problem as long as $1 \le C(i) \le N_G$ for any $i = 1,\ldots,N_{AC}$. For another two structures, some special constraints must be satisfied. The feasibility of chromosomes based on absolute position of aircraft is defined by two constraints: (I) each aircraft appears once and only once in a chromosome, and (II) if $C(g,j) > 0$, then $C(g,h) > 0$ for all $1 \le h < j$. For the GAUC proposed in this paper, a feasible chromosome must satisfy the following constraints according to the underlying physical meaning in the MOGAP:

$$\sum_{i=1}^{N_{AC}} \sum_{j=1}^{N_{AC}} C(i,j) = N_{AC}, \tag{10}$$

$$\sum_{j=1}^{N_{AC}} C(i,j) \begin{cases} \le 2, & C(i,i) > 0 \\ \le 1, & C(i,i) = 0 \end{cases} \tag{11}$$

$$\sum_{i=1}^{N_{AC}} C(i,j) = 1, \tag{12}$$

$$1 \le \sum_{i=1}^{N_{AC}} C(i,i) = \bar{N_G} \le N_G, \tag{13}$$

$$\sum_{C(N_{AC}+1,j)=g, j=1,\ldots,N_{AC}} C(j,j) = 1 \text{for any} g \in \bar{\Phi}_G, \tag{14}$$

where, without losing the generality, it is assumed that only $\bar{N_G}$ gates in all N_G gates are assigned to aircraft and $\bar{\Phi}_G$ denotes the set of assigned gates. Constraints (10) to (14) are actually a new version of the two feasibility constraints for chromosomes based on absolute position. From constraints (10) to (11), one can derive that there may often be some empty rows, no more than $\bar{N_G}$ empty rows, in the matrix. If the ith row is empty, then it means aircraft i is the last aircraft to dwell to gate $C(N_{AC}+1,i)$.

Actually, constraints (10) to (14) will rarely be used in the GAUC. In the initialization of a chromosome, the following procedure can efficiently generate a feasible chromosome only with a need to check against constraint (13):

Step 1: Create a $(N_{AC}+1) \times N_{AC}$ matrix with all entries set as 0. Let $U = \{1,\ldots,N_{AC}\}$ represent the original set of aircraft, and let $\Phi_G = \{1,\ldots,N_G\}$ be the set of gates.

Step 2: While $U \neq \emptyset$, do

Step 2.1: For $1 \leq i \leq N_{AC}$, if one more new $C(i,i) = 1$ will violate Constraint (13), then randomly choose an existing $C(i,i) = 1$, let $g = C(N_{AC}+1,i)$, and go to Step 2.2. Otherwise, choose $i \in U$ and $g \in \Phi_G$ randomly, set $C(i,i) = 1$, $C(N_{AC}+1,i) = g$, and remove i from U and g from Φ_G, i.e., let $U = U - \{i\}$ and $\Phi_G = \Phi_G - \{g\}$.

Step 2.2: Regardless of $C(i,i)$, if there is no non-zero entry in row i, then randomly choose $j \in U$, set $C(i,j) = 1$, $C(N_{AC}+1,j) = g$, and remove j from U, i.e., let $U = U - \{j\}$. Otherwise, find the $C(i,j) = 1$, let $i = j$, and repeat Step 2.2.

3.2 Mutation Operator

Mutation is used by GAs to diversify chromosomes in order to exploit solution space as widely as possible. In the case of MOGAP, the mutation operation should be able to reassign an aircraft to any gate at any order. Therefore, we need two mutation operators: (I) one to shift randomly the positions of two successive aircraft in a same queue, and (II) the other to swap randomly aircraft in two different queues, or to remove an aircraft from one queue, and then append it to the end of another queue. The chromosome structure based on gate assignment only supports the second mutation. The structure based on absolute position supports both as denoted as follows:

Mutation I: $C(g,j) \leftrightarrow C(g,j+1), j = 1,\ldots,H_g - 1, g = 1,\ldots,N_G$.

Mutation II: $C(g_1,j) \leftrightarrow C(g_2,k), j = 1,\ldots,H_{g_1}$ and $k = 1,\ldots,H_{g_2}+1, g_1 \neq g_2, g_1 = 1,\ldots,N_G, g_2 = 1,\ldots,N_G$.

In the GAUC, the above two mutation operators need to be re-designed as the following in order to fit in the chromosome structure based on both gate assignment and relative position between aircraft in queues:

Mutation III: Randomly choose a non-zero gene, say, $C(i,j) = 1$. If there exist a $C(j,m) = 1$ (and maybe further a $C(m,h) = 1$), then change the values of some genes by following the instructions given in Fig.4.(a).

Mutation IV: Randomly choose two non-zero genes, say, $C(i,j) = 1$ and $C(h,x) = 1$ (there may be $C(j,m) = 1$ and/or $C(x,y) = 1$), which have different assigned gates, i.e., $C(N_{AC}+1,j) \neq C(N_{AC}+1,x)$. Then reset the values of some genes by following the instructions given in Fig.4.(b).

The mutation operations given in Fig.4 automatically guarantee the feasibility of resulting chromosomes as long as the original chromosomes are feasible.

3.3 Crossover Operator

There has long been a strong debate about the usefulness of crossover, and some people consider crossover as a special case of mutation, which is true in many designs of GAs [22]. However, we believe the fundamental role of crossover is different from that of mutation. Mutation aims to diversify chromosomes, while crossover can converge them by identifying, inheriting and protecting their common genes. As it is well known, it

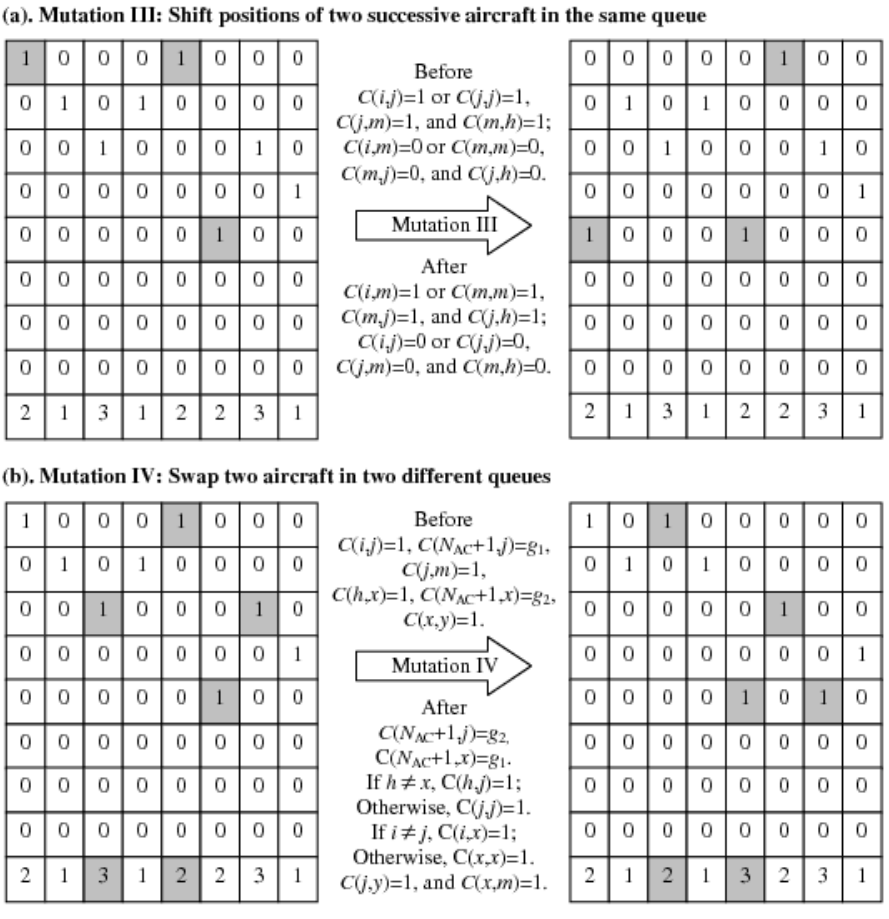


Fig. 4. Mutation operators in GAUC

is crucial for GAs to keep a good balance between diversity and convergence in the evolutionary process, which is really a challenging task in the design of GAs. The difficulties in the design of highly efficient crossover operators do not and also should not mean that crossover is useless. Otherwise, we should expect to see a natural world dominated by single gender species, which however we all know is not true. Therefore, as a nature-inspired algorithm, GAs should have some primary tasks, to identify, to inherit and to protect good/useful common genes in chromosomes, only or at least mainly for crossover.

Uniform crossover is probably the most wildly used crossover operator because of its efficiency in not only identifying, inheriting and protecting common genes, but also re-combining non-common genes [19]-[21]. Fig.5, using the chromosome structure based on gate assignment, compares uniform crossover with another also wildly used crossover operator: one position split crossover. From Fig.5 one can see that the

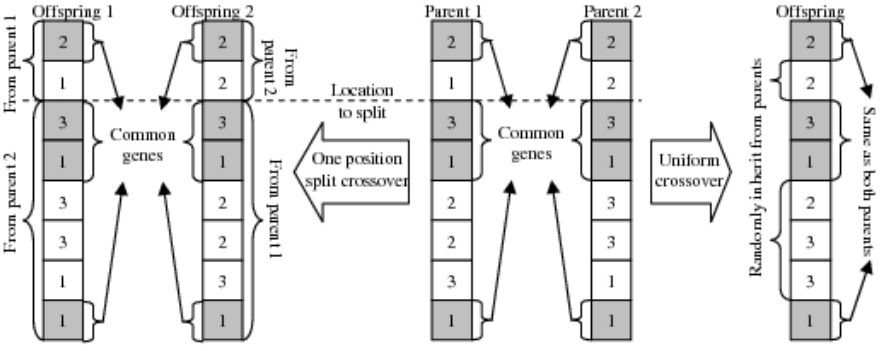


Fig. 5. Uniform crossover vs. on position split crossover

uniform crossover is actually a $(N_{AC}-1)$-position-split crossover, which is obviously much more powerful than the one position split crossover in terms of exploiting all possibilities of recombining non-common genes.

To design an effective and efficient uniform crossover operator to handle common relative position between aircraft in queues is a major objective of this paper. Although the chromosome structure based on absolute position of aircraft contains the information of relative position, due to the feasibility issue in chromosomes, it is very difficult to design an effective crossover operator to identify, inherit and protect common relative positions. However, for comparative purposes, here we still manage to design a crossover operator for the absolute position based chromosome structure:

$$\begin{aligned} &\text{If} \quad \{C_1(1,j),\ldots,C_1(N_G,j)\} = \{C_2(1,k),\ldots,C_2(N_G,k)\} \neq \{0,\ldots,0\} \\ &\text{then} \quad C_1(.,j) \leftrightarrow C_2(.,k) \end{aligned} \tag{15}$$

Equation (15) requires the set of all jth aircraft in C_1 to be the same as the set of all kth aircraft in C_2. This crossover does not often cause feasibility problems. Actually, it has no feasibility problem if the following constraint is added

$$C_1(g,j) > 0, C_2(g,k) > 0 \quad \text{for all} \quad g = 1,\ldots,N_G \tag{16}$$

However, what this crossover operator does is not what a crossover operator is expected to do, supposing we want to identify, inherit and protect those common genes defined by either gate assignment or relative position between aircraft. Actually the above crossover operator can be considered as a combination of Mutation I and II.

A focus of this paper is to design an effective and efficient crossover operator which can identify, inherit and protect both common gate assignment and common relative position between aircraft in queues to gates, and at the same time which can exploit all possibilities of re-combining non-common genes. The feasibility issue of the new crossover operator should be addressed in a computationally cheap way. With the new chromosome structure illustrated in Fig.2.(d) and the definitions of common genes shown in Fig.3.(c), we propose a novel uniform crossover operator described as the following procedure, which is further illustrated by Fig.6:

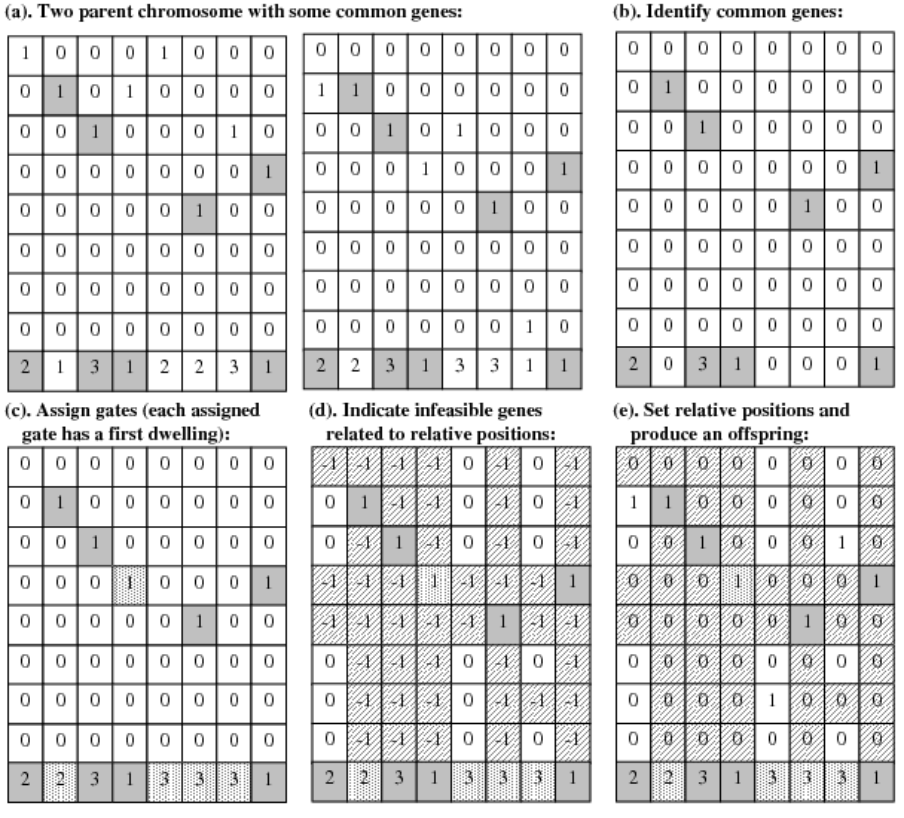


Fig. 6. Uniform crossover operation in GAUC

Step 1: Given two parent chromosomes C_1 and C_2, calculate C_3 to locate common genes

$$C_3(i,j) = C_1(i,j) \& C_2(i,j),$$

$$C_3(N_{AC}+1,j) = \begin{cases} C_1(N_{AC}+1,j), & C_1(N_{AC}+1,j) = C_2(N_{AC}+1,j) \\ 0 & C_1(N_{AC}+1,j) \neq C_2(N_{AC}+1,j) \end{cases} \quad (17)$$

$i = 1,\ldots,N_{AC}, j = 1,\ldots,N_{AC},$

i.e., $C_3(i,j) = 1$ or $C_3(N_{AC}+1,j) > 0$ means this location has a common gene shared by C_1 and C_2.

Step 2: Assign gates to C_3 by referring to C_1 and C_2. Basically, $C_3(N_{AC}+1,j)$ is set as $C_1(N_{AC}+1,j)$ or $C_2(N_{AC}+1,j)$, and $C_3(j,j)$ is set as $C_1(j,j)$ or $C_2(j,j)$, $j = 1,,N_{AC}$, at a half-and-half chance, subject to Constraint (14). Let $C_4 = C_3$.

Step 3: Indicate infeasible genes related to relative positions in C_4: Set $C_4(i,i) = -1$ for $i = 1,\ldots,N_{AC}$; If $C_3(i,j) = 1$ for $i \neq j$, then set $C_4(m,j) = -1$ and $C_4(i,m) = -1$

for $m = 1,\ldots,N_{AC}$; if $C_3(i,i) = 1$, then set $C_4(m,i) = -1$ for $m = 1,\ldots,N_{AC}$. $C_4(i,j) \neq 0$ means this location will not be considered when a new relative position between aircraft needs to be set up.

Step 4: While $\sum C_3(i,j) < N_{AC}, i = 1,\ldots,N_{AC}$, and $j = 1,\ldots,N_{AC}$, do

Step 4.1: Randomly choose j, such that $\sum C_3(i,j) = 0, i = 1,\ldots,N_{AC}$.

Step 4.2: Suppose $C_1(i_1,j) = 1$ and $C_2(i_2,j) = 1$, $i_1 = 1,\ldots,N_{AC}$, and $i_2 = 1,\ldots,N_{AC}$. If $C_3(N_{AC}+1,i_1) = C_3(N_{AC}+1,i_2) = C_3(N_{AC}+1,j)$ and $C_4(i_1,j) = C_4(i_2,j) = 0$, then set $i_3 = i_1$ or $i_3 = i_2$ at a half-and-half chance; Else if $C_3(N_{AC}+1,i_n) = C_3(N_{AC}+1,j)$ and $C_4(i_n,j) = 0$, $n = 1$ or 2, then set $i_3 = i_n$; Otherwise, randomly choose i_3 such that $C_3(N_{AC}+1,i_3) = C_3(N_{AC}+1,j)$ and $C_4(i_3,j) = 0$.

Step 4.3: Set $C_3(i_3,j) = 1, C_4(i_3,j) = 1, C_4(m,i_3) = -1$ and $C_4(i_3,m) = -1$ for $m = 1,\ldots,N_{AC}$.

Clearly, with the above crossover procedure, all common genes are efficiently identified, inherited and protected, and all possibilities of feasibly re-combining non-common genes can be exploited. As will be proved later, this uniform crossover is a very powerful search operator in the proposed GA.

3.4 Heuristic Rules

To further improve the performance of GA, e.g., to stimulate necessary and/or potentially useful local search, the following problem-specific heuristic rules are introduced:

- To help the algorithm to converge fast, not all of the new chromosomes are initialized randomly, but some are generated according to the FCFS principle. This is because, according to the real-world GAP operation, an FCFS-based solution is fairly reasonable, and an optimal or sub-optimal solution is often not far away from such an FCFS-based solution. Therefore, initializing some chromosomes according to the FCFS principle can effectively stimulate the local search around the FCFS-based solution.
- When initializing a chromosome randomly, we still follow the FCFS principle but in a loose way, i.e., an aircraft with an earlier P_i is more likely to be assigned to the front of a queue. This rule is also used to increase the chance of local search around the FCFS-based solution.
- If two aircraft have a same P_i, or their P_is are within a specified narrow time window, then the one with more passengers stands a better chance to be allowed to dwell first. This rule may help to reduce the total passenger waiting time significantly.
- For the sake of diversity, in each generation, a certain proportion of worst chromosomes are replaced by totally new ones.
- Like in [18], the population in a generation, $N_{Population}$, and the maximum number of generations in the evolutionary process, $N_{Generation}$, are adjusted according to N_{AC} in order to roughly keep the level of solution quality

$$N_{Population} = 30 + 10(round(max(0, N_{AC} - 10)/5)), \tag{18}$$

$$N_{Generation} = 40 + 15(round(max(0, N_{AC} - 10)/5)). \tag{19}$$

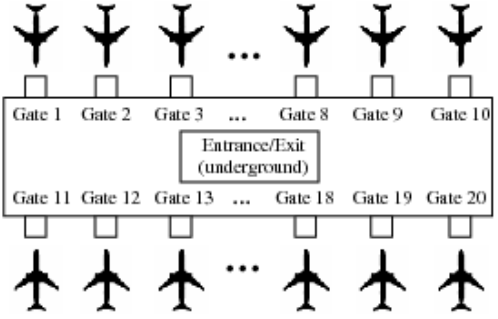


Fig. 7. Two-sided parking terminal layout

4 Simulation Results

The terminal layout has a big influence on the cost-efficiency of daily airport operations [23]. In our study, a typical terminal layout, two-sided parking terminal, is used, as illustrated in Fig.7. The terminal is assumed to have 20 gates. The data matrix M_{PWD} and M_{BTD}, i.e., distances for passenger walking and baggage transporting, are generated according to (20) to (23)

$$M_{PWD}(n,m) = M_{PWD}(m,n) = d_1 + d_2|nrem(n) - nrem(m)| \tag{20}$$

$$M_{PWD}(n,N_G+1) = M_{PWD}(N_G+1,n) = d_3 + d_2|nrem(n) - 5.5| \tag{21}$$

$$M_{BTD}(n,m) = M_{BTD}(m,n) = d_4 + d_5|nrem(n) - nrem(m)| \tag{22}$$

$$M_{BTD}(n,N_G+1) = M_{BTD}(N_G+1,n) = d_6 + d_5|nrem(n) - 5.5| \tag{23}$$

where $n = 1,\ldots,N_G$, $m = 1,\ldots,N_G$, and d_1 to d_6 are constant coefficients which can roughly determine the terminal size and the gate locations,

$$nrem(n) = \begin{cases} rem(n,11), & n < 11 \\ rem(n-10,11), & n \geq 11 \end{cases} \tag{24}$$

and *rem* is a function that calculate the remainder after division.

Traffic and passenger data are generated randomly under the assumption that the capacity of an aircraft varies between 50 and 300, the ground time span at a gate is between 30 and 60 minutes, and all aircraft are planned to arrive or depart within a period of one-hour time window. The congestion condition is indicated by N_{AC}. For comparative purposes, the GA reported in [18] is extended to solve the MOGAP. As discussed in Section 3, this extended GA employs the chromosome structure based on absolute position, and its crossover is actually a more complex mutation operator. Therefore, it is denoted as GACMO, in order to distinguish from the proposed GAUC

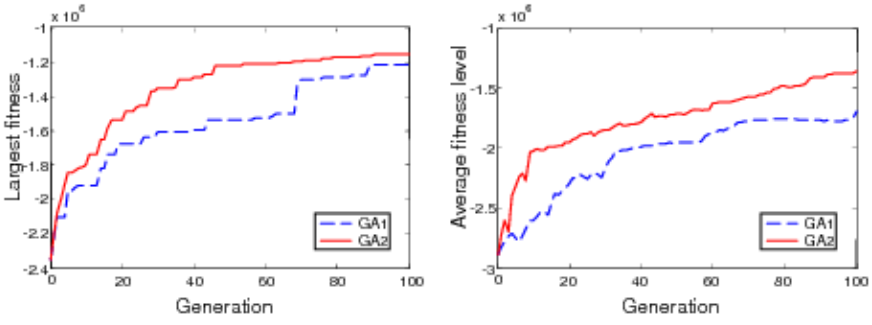


Fig. 8. Fitness levels in a test

with uniform crossover. Due to limited space, here we only give in Table 1 the results of a relatively simple case study, in order to illustrate how different GAs optimize gate assignment. In this test, J_{MOGAP} under GAUC is about 5% smaller than that of GACMO, and Fig.8 shows how the fitness, i.e. $-J_{MOGAP}$, changes in the evolutionary processes of GACMO and GAUC. From Fig.8.(a), one can see that the largest fitness of a generation in GAUC increases more quickly than that in GACMO, which means GAUC, which uses the new chromosome structure and uniform crossover, has a faster convergence speed than GACMO, which is designed based on absolute position of aircraft and has a crossover only equivalent to a combination of Mutation I and II. Actually, on average in this test, it takes GAUC 2.7778 generations to make a breakthrough in the largest fitness, while for GACMO, it takes 4.7619 generations. From Fig.8.(b) one can see that the average fitness of a generation in GAUC increases faster and stays larger than that in GACMO. This implies GAUC can effectively improve the overall fitness level, which is probably because the new uniform crossover proposed in this paper really works well in identifying, inheriting and protecting good common genes.

However, to get general conclusions about different GAs, we need to conduct extensive simulation tests, where N_{AC} is set as 30, 60 or 90 to simulate the situation of under-congestion, congestion, or over-congestion, and one of the single-objective functions in (4) to (6) or the multi-objective function in (7) is used. For each N_{AC} and objective function, 100 simulation runs are conducted under each GA, and the average results are given in Table 2 to Table 5, from which we have the following observations:

- Overall, GAUC is about $3\% \sim 10\%$ better than GACMO in terms of the specific objective function, which illustrates the advantages of the new chromosome structure based on relative position of aircraft in queues to gates and of the proposed uniform crossover operator based on the new structure.
- In the cases of single-objective GAP, as given in Table 2 to Table 4, GAUC achieves a better performance at the cost of other non-objective criteria. For instance, in Table 2, GAUC gets a smaller TPWD by sacrificing TAWT (total aircraft waiting time). TPWD and TBTD share a similar trend of change, i.e., if GAUC has a smaller/larger TPWD than GACMO, then it also has a smaller/larger TBTD. This is probably because both TPWD and TBTD, in a similar way, are determined largely by the terminal layout.

Table 1. Result of gate assignment in a single test

AC Code	$P_i(min)$	$G_i(min)$	GACMO $E_i(min)$	GACMO Gate	GAUC $E_i(min)$	GAUC Gate
1	28	40	28	3	28	3
2	26	50	26	16	26	16
3	5	40	5	9	5	9
4	12	45	12	7	12	7
5	43	50	43	19	43	19
6	27	45	27	4	27	4
7	34	40	34	2	34	2
8	10	35	10	12	10	12
9	48	30	48	12	48	12
10	56	35	56	14	56	14
11	25	35	25	15	25	15
12	52	35	53	13	53	13
13	7	50	7	11	7	8
14	56	40	63	8	57	8
15	56	60	60	15	60	15
16	49	35	49	20	49	5
17	39	30	39	1	39	1
18	47	40	47	9	47	9
19	13	40	13	13	13	13
20	52	35	57	11	63	11
21	25	50	25	5	25	20
22	35	40	35	18	35	18
23	16	50	16	6	16	6
24	20	35	20	14	20	14
25	56	45	66	6	66	6
26	4	40	4	10	4	10
27	8	55	8	8	8	11
28	54	50	57	7	57	7
29	45	35	45	10	45	10
30	28	45	28	17	28	17

Table 2. J_{TPWD} is used as objective function

$(\times 10^5)$		J_{TPWD}	TPWD(m)	TBTD(m)	TAWT(min)	MaxQL	MinQL
$N_{AC} = 30$	GACMO	7.2656	7.2656	18.9600	43.4807	29.5	0.2
	GAUC	7.0330	7.0330	18.4091	44.7622	29.6	0.2
$N_{AC} = 60$	GACMO	14.0606	14.0606	38.5475	201.1071	59.1	0.3
	GAUC	13.2538	13.2538	37.2206	209.5881	59.3	0.3
$N_{AC} = 90$	GACMO	19.7178	19.7178	56.4425	442.9681	88.6	0.6
	GAUC	18.8373	18.8373	55.0299	455.4340	88.5	0.5

Table 3. J_{TBTD} is used as objective function

$(\times 10^5)$		J_{TBTD}	TPWD(m)	TBTD(m)	TAWT(min)	MaxQL	MinQL
$N_{AC} = 30$	GACMO	18.5846	7.2739	18.5846	43.6277	29.5	0.3
	GAUC	17.8939	7.1005	17.8939	45.1086	29.6	0.2
$N_{AC} = 60$	GACMO	38.1412	14.2805	38.1412	202.0039	59.3	0.3
	GAUC	36.9374	13.1288	36.9374	210.1956	59.5	0.3
$N_{AC} = 90$	GACMO	55.8907	20.1136	55.8907	440.7336	88.8	0.7
	GAUC	54.0407	18.9287	54.0407	451.1360	89.0	0.6

Table 4. J_{TPWT} is used as objective function

$(\times 10^5)$		J_{TPWT}	TPWD(m)	TBTD(m)	TAWT(min)	MaxQL	MinQL
$N_{AC} = 30$	GACMO	1.5273	18.8120	24.8046	0.0611	2.2	0.9
	GAUC	1.4595	19.0023	24.9367	0.0583	2.2	0.9
$N_{AC} = 60$	GACMO	71.2180	36.9188	50.4188	2.8487	3.9	2.3
	GAUC	64.2053	37.3578	51.9046	2.5774	3.8	2.3
$N_{AC} = 90$	GACMO	219.8557	51.6150	73.1843	8.7942	5.3	3.9
	GAUC	208.5154	53.0487	75.5549	8.3508	5.3	4.0

Table 5. J_{MOGAP} is used as objective function

$(\times 10^5)$		J_{MOGAP}	TPWD(m)	TBTD(m)	TAWT(min)	MaxQL	MinQL
$N_{AC} = 30$	GACMO	11.9457	16.1300	23.5272	0.1528	2.0	0.9
	GAUC	11.4672	15.5086	23.0442	0.1477	2.1	1.0
$N_{AC} = 60$	GACMO	53.0853	35.9684	49.8836	3.0112	3.9	2.2
	GAUC	49.5900	34.1606	48.7724	2.8031	4.0	2.1
$N_{AC} = 90$	GACMO	120.2156	49.7772	72.3854	8.8088	5.3	4.0
	GAUC	115.7206	47.8941	72.2129	8.4692	5.2	4.0

- In the case of MOGAP, if the weights in the objective function are properly tuned ($\alpha = 0.5$ and $\beta = 0.1$ in the associated tests), GAUC is better than GACMO not only in terms of the multi-objective function adopted, but also in terms of each single-objective function not adopted.
- In the minimum distance (passenger walking distance or baggage transporting distance) GAP, as shown in Table 2 and Table 3, we use no extra constraints to enforce assigning gates evenly to aircraft. As a result, the gap between the maximum queue length (MaxQL) and the minimum queue length (MinQL) is huge, which implies many aircraft are assigned to a certain gate. While in the minimum waiting time GAP, as given in Table 4, the gap between MaxQL and MinQL is very small, which means evenly using gates is automatically guaranteed during the minimization of

waiting time. Therefore, since waiting time is considered in the MOGAP, the gap between MaxQL and MinQL is also very small, as shown in Table 5.

- Basically, in a more congested case, i.e., with a larger N_{AC}, the operation of gate assignment is more expensive. Roughly speaking, the distances increase linearly in terms of N_{AC}, while the waiting time goes up exponentially, mainly because of the heavy delay applied to aircraft during a congested period. This might suggest, in a more congested case, waiting time should be given a larger weight.

5 Conclusion

Uniform crossover is usually efficient in identifying, inheriting and protecting common genes in GAs, but it could be difficult to design or apply when chromosomes are not properly constructed. This paper aims to design an efficient GA with uniform crossover to tackle the multi-objective gate assignment problem (MOGAP) at airport terminals. Instead of the absolute position of aircraft in queues to gates, which is widely used in existing GAs for the GAP, the relative position between aircraft is used to construct chromosomes in the new GA. A highly efficient uniform crossover operator is then designed, which is effective to keep a good balance between diversity and convergence in the evolutionary process. The advantages of the new GA are demonstrated in extensive simulation tests. Further research will be conducted in order to extend the reported work from static air traffic situation to dynamical environment based on real traffic data which need to be collected and analyzed.

Acknowledgements

This work was supported by the EPSRC Grant EP/C51632X/1. A previous version of this paper was presented at The 2007 IEEE Congress on Evolutionary Computation (CEC2007), 25-28 Sep 2007, Singapore.

References

1. Haghani, A., Chen, M.C.: Optimizing gate assignments at airport terminals. Transportation Research A 32, 437–454 (1998)
2. Bolat, A.: Procedures for providing robust gate assignments for arriving aircraft. European Journal of Operations Research 120, 63–80 (2000)
3. Babic, O., Teodorovic, D., Tosic, V.: Aircraft stand assignment to minimize walking distance. Journal of Transportation Engineering 110, 55–66 (1984)
4. Mangoubi, R.S., Mathaisel, D.F.X.: Optimizing gate assignments at airport terminals. Transportation Science 19, 173–188 (1985)
5. Bihr, R.: A conceptual solution to the aircraft gate assignment problem using 0,1 linear programming. Computers & Industrial Engineering 19, 280–284 (1990)
6. Gosling, G.D.: Design of an expert system for aircraft gate assignment. Transportation Research A 24, 59–69 (1990)
7. Srihari, K., Muthukrishnan, R.: An expert system methodology for an aircraft-gate assignment. Computers & Industrial Engineering 21, 101–105 (1991)

8. Xu, J., Bailey, G.: Optimizing gate assignments problem: Mathematical model and a tabu search algorithm. In: Proceedings of the 34th Hawaii International Conference on System Sciences. Island of Maui, Hawaii, USA (2001)
9. Ding, H., Lim, A., Rodrigues, B., Zhu, Y.: The over-constrained airport gate assignment problem. Computers & Operations Research 32, 1867–1880 (2005)
10. Ding, H., Lim, A., Rodrigues, B., Zhu, Y.: New heuristics for the over-constrained flight to gate assignments. Journal of the Operational Research Society 55, 760–768 (2004)
11. Robuste, F.: Analysis of baggage handling operations at airports. PhD thesis, University of California, Berkeley, USA (1988)
12. Chang, C.: Flight sequencing and gate assignment in airport hubs. PhD thesis, University of Maryland at College Park, USA (1994)
13. Robuste, F., Daganzo, C.F.: Analysis of baggage sorting schemes for containerized aircraft. Transportation Research A 26, 75–92 (1992)
14. Wirasinghe, S.C., Bandara, S.: Airport gate position estimation for minimum total costs-approximate closed form solution. Transportation Research B 24, 287–297 (1990)
15. Gu, Y., Chung, C.A.: Genetic algorithm approach to aircraft gate reassignment problem. Journal of Transportation Engineering 125, 384–389 (1999)
16. Bolat, A.: Models and a genetic algorithm for static aircraft-gate assignment problem. Journal of the Operational Research Society 52, 1107–1120 (2001)
17. Yan, S., Huo, C.M.: Optimization of multiple objective gate assignments. Transportation Research A 35, 413–432 (2001)
18. Hu, X.B., Chen, W.H.: Genetic Algorithm Based on Receding Horizon Control for Arrival Sequencing and Scheduling. Engineering Applications of Artificial Intelligence 18, 633–642 (2005)
19. Sywerda, G.: Uniform crossover in genetic algorithms. In: Proceedings of the 3rd International Conference on Genetic Algorithms, USA (1989)
20. Page, J., Poli, P., Langdon, W.B.: Smooth uniform crossover with smooth point mutation in genetic programming: A preliminary study, Genetic Programming. In: Langdon, W.B., Fogarty, T.C., Nordin, P., Poli, R. (eds.) EuroGP 1999. LNCS, vol. 1598, p. 39. Springer, Heidelberg (1999)
21. Falkenauer, E.: The worth of uniform crossover. In: Proceedings of the 1999 Congress on Evolutionary Computation, USA (1999)
22. Eiben, A.E., Schoenauer, M.: Evolutionary computing. Information Processing Letters 82, 1–6 (2002)
23. Bandara, S., Wirasinghe, S.C.: Walking distance minimization for airport terminal configurations. Transportation Research A 26, 59–74 (1992)

Application of Evolutionary Algorithms for Solving Multi-Objective Simulation Optimization Problems

Lee Loo Hay, Chew Ek Peng, Teng suyan, and Li juxin

Department of Industrial and Systems Engineering, National University of Singapore,
10 Kent Ridge Crescent, Singapore 119260
iseleelh@nus.edu.sg

1 Introduction

Most optimization problems associated with real-world complex systems are too difficult to be modeled analytically. The difficulty may come from the uncertainties involved in the input and output of the system, the complex (often nonlinear) relationships between the system decision variables and the system performances, and possible multiple, conflicting performance measures to be considered when choosing the best design. In light of this, discrete event simulation has become one of the most popular tools for the analysis and design of complex systems due to its flexibility, its ability to model systems unable to be modeled through analytical methods, and its ability to model the time dynamic behavior of systems [1]. However, simulation can only evaluate system performances for a given set of values of the system decision variables, i.e., it lacks the ability of searching for optimal values which would optimize one or several responses of the system. This explains the increasing popularity of research in integrating both simulation and optimization, known as simulation optimization: the process of finding the best values of decision variables for a system where the performance is evaluated based on the output of a simulation model of this system.

Another feature of the real-world complex problems is that most of them are multi-objective in nature as they require the simultaneous optimization of multiple, often competing objectives. One pertinent instance is the manufacturing system design, where some typical performance measures, such as throughput, ability to meet deadline, resource utilization, in-process inventory, and overall system cost are simultaneously considered [2]. Due to the trade-offs involved in the objectives, a unique optimal solution to the multi-objective problem exists only when objectives are combined into a single criterion to be optimized according to a known utility function. When the utility function is not well known in advance, the solution to the multi-objective problem is a set of equally good, non-dominated solutions known as the Pareto-optimal set.

As most real-world complex problems fall into the multi-objective simulation optimization (MSO) category, we now give a general formulation of the MSO problem

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 91–110.
springerlink.com

which minimizes the expected value of the objective function with respect to its constraint set as follows:

$$\min_{\theta \in \Theta} J(\theta). \tag{1}$$

where $J(\theta) = E[L(\theta, \varepsilon)]$ is the vector of expected performance measure of the problem, $L(\theta, \varepsilon)$ is the vector of sample performance, ε represents the stochastic effects in the system, θ is a p-vector of discrete controllable factors and Θ is the discrete constraint set on θ.

Several features associated with the above MSO problem make it both challenging and difficult to solve: the uncertainties (ε) involved in the performance measures, the possible huge size of solution space (Θ), and multi-objective which requires a non-dominated Pareto set of solutions. Research addresses this problem as a whole is few and far between in the literature. However, some simplified versions of the problem are relatively easy and have been studied more extensively.

One straightforward simplification of the problem is when the objective is a scalar function rather than a vector-valued function. With a finite and relatively small solution space (Θ), formulation (1) is usually known as the Ranking and Selection (RS) problem [3]. Solution approaches to the RS problem all exhaustively evaluate each alternative and focus on how to statistically select the best alternative through statistical comparison at the least expense of simulation budget. These include indifference-zone ranking and selection [4], optimal computing budget allocation [5], and decision theoretic methods [6]. When the solution space is infinite or finite but very huge, formulation (1) becomes more complex and is known as the single objective simulation optimization problem. In this case, exhaustive evaluation becomes impractical or impossible. Search powers of optimization procedures need to be integrated with the statistical selection procedure to more efficiently explore the solution space for finding improving solutions. Research in this area shows that most studies integrate meta-heuristics with certain statistical analysis techniques, such as genetic algorithm (GA) with both a multiple comparison procedure and a RS procedure [7]; GA with indifference-zone RS procedure [8]; modified simulated annealing (SA) algorithm with confidence interval [9]; SA with RS procedure [10]; nested partitions search with two-stage RS procedure[11], evolutionary algorithm (EA) with subset selection of top-m designs [12].

When formulation (1) is considered having multi-objectives, the problem becomes even more complex. With no uncertainties (θ) involved in the performance measures, the problem is a well-studied deterministic multi-objective optimization problem. One typical solution approach is the multi-objective programming method, such as Lexicographic Weighted Tchebycheff Sampling Program [13], Wierzbicki's Aspiration Criterion Vector Method [14, 15], STEM [16], Global Shooting [17], TRIMAP [18], Light Beam Search [19], Pareto Race [20], etc. More recently, another approach called the multi-objective evolutionary algorithm (MOEA) is gaining more and more attention from researchers in various fields [21, 22, 23, 24, 25]. The popularity of MOEA in solving multi-objective optimization problems can be reflected from the considerable amount of research currently reported in the literature [26]. When uncertainties (θ) are involved in the performance measures but the solution space (Θ) is finite and relatively small, the problem is known as the Multi-objective Ranking and Selection (MORS) problem and is studied in [27, 28]. They developed a multi-objective computing budget

allocation (MOCBA) procedure which incorporates the concept of Pareto optimality into the RS scheme to find all non-dominated solutions.

With all three difficult features of formulation (1) being considered, we are confronted with the MSO problem as a whole. Research in current literature shows that the MSO problem is not well-studied in the sense that either it is solved through single objective optimization techniques or the uncertainties (ε) in the performance measures are overlooked. In the former case, the problem is usually transformed into single objective problem through goal programming [29] or some multiple attribute utility theory [30]. In the latter case, some EA frameworks deal with the uncertainties in the performance measures completely relying on the robustness of EA due to its tolerance to nonlinear and noisy system models and performance measures [31, 32]. Some researchers try to account for the uncertainties by sampling the performance measures a fixed number of times for each alternative and using the sample average as estimate of the performance measures [33]. In both cases, the problem is technically treated as a deterministic one. Two papers try to explicitly address the uncertainties involved in the performance measures within EA framework. In [34], the authors proposed a probabilistic ranking method to select chromosomes based on uncertain multi-objective fitness measurements. In [35], three noise-handling features, an experiential learning directed perturbation operator, a gene adaptation selection strategy and a possibilistic archiving model are proposed. However, these studies tend to assume that problems caused by noise and uncertainties can be overcome by some noise-handling techniques incorporated into the MOEA process without considering what is the appropriate noise level MOEA can tolerate and how to reduce noise to the appropriate level. In the MSO problem considered in this study, as performance measures are outputs from simulation model, high uncertainties are oftentimes inevitable. This may render the above sampling and average method and the noise-handling techniques inefficient. In this study, we propose a solution framework which integrates MOCBA [28] and MOEA for solving the MSO problem. The integrated MOEA solution framework would explicitly address the uncertainties (ε) in the performance measures through efficient control of the estimation accuracy of the performance measures. This chapter is organized as follows. In Section 2, we first give a general overview of the MOEA procedure and discuss its limitation when solving the MSO problems. Then we briefly introduce the MOCBA algorithm. Finally we present the integrated MOEA framework. In Section 3, a benchmark test problem is applied to quantify how much the integrated MOEA framework can benefit from MOCBA in comparison with equal allocation of simulation replications. Convergence and diversity of the elite set of solutions are also investigated. Finally some conclusions and future research directions are summarized in Section 4.

2 An Integrated MOEA Framework for Solving the MSO Problem

When a general MSO problem is considered, the solution space (Θ) is very likely to be very huge or infinite. In this case, a search procedure is essential in automatically identifying those decision scenarios with better decisions and therefore more desirable to be investigated. In this study, we employ MOEA to efficiently explore the solution space for more promising solutions due to its popularity in multi-objective optimization.

However, we are not intended to propose a new more efficient MOEA to complement the current MOEA literature. Our main purpose is to present a framework to integrate the current available MOEA with a statistical selection procedure so that the integrated framework can deal with the uncertainties involved in the performance measures more effectively and efficiently. The solution framework is an integration of MOEA with MOCBA - a statistical selection procedure developed in [28].

2.1 An Overview of MOEA and Its Limitations When Solving MSO Problems

Evolutionary Algorithms (EAs) are adaptive heuristic search algorithm inspired by evolutionary theory: natural selection and survival of the fittest. The use of EA in multi-objective optimization has been widely studied, experimented and applied in many fields due to its particular suitableness for multi-objective optimization [26]: 1) its ability to work simultaneously with a population of promising solutions which would evolve into a set of Pareto optimal solutions at termination; 2) its insusceptibility to the shape or continuity of the Pareto front. The MOEAs developed with various features include: the Vector Evaluated Genetic Algorithm (VEGA) proposed in [36], the Multi-Objective Genetic Algorithm (MOGA) in [22], the Niched Pareto Genetic Algorithm (NPGA) in [23], the Non-dominated Sorting Genetic Algorithm (NSGA) in [24], the Strength Pareto Evolutionary Algorithm (SPEA) in [25], the Pareto Archived Evolution Strategy (PAES) in [37], the Incrementing Multi-objective Evolutionary Algorithm (IMOEA) with dynamic population size in [38], and the two-archive MOEA for solving problems with a large number of objectives in [39]. Though these algorithms differ in one way or another, they all share some MOEA's basic features and common steps.

As an adaptive search algorithm, MOEA simulates the survival of the fittest among individuals over consecutive generations for solving a problem. Each generation consists of a population of chromosomes representing possible solutions in the solution space (Θ). As the population evolves from generation to generation, MOEA identifies non-dominated individuals in each population (elite population) and have them evolve towards the final Pareto set. Specifically, starting with an initial population, at each iteration, MOEA evaluates the chromosomes and rank them in terms of their "fitness"; then possible non-dominated solutions are archived as elite population; if termination condition is not met, fitter solutions would be selected as parents and undergo recombination and /or mutation to generate new solutions which are hopefully biased towards regions with good solutions already seen; then a selection procedure is possibly performed to get rid of less fit solutions. The procedure is iterated until the termination condition is met and the archived solutions are output as the final Pareto set. A general MOEA procedure is illustrated through the flowchart in Figure 1.

Though MOEA procedure is generally considered as robust due to its tolerance to noise involved in the performance measures, its effectiveness is likely to be limited to low noise situations. When MOEA is applied in solving an MSO problem, where performance measures are evaluated through simulation with output subjected to high variability, it may be confronted with several difficulties as described below:

- How to determine the number of replications needed to estimate the performances of solutions in the population;

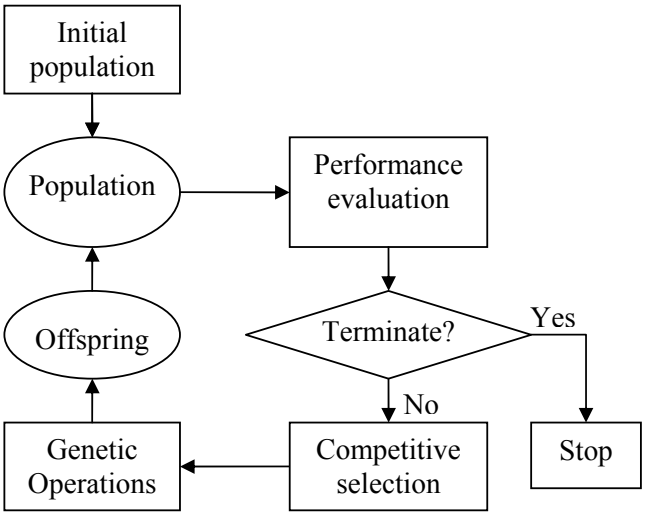


Fig. 1. Flowchart for a General MOEA Procedure

- How to evaluate "fitness" of the solutions and rank them when variability is involved in the performance measures;
- How to identify the non-dominated solutions and ensure that the final set of solutions are truly non-dominated among all visited solutions with statistical significance.

To overcome these difficulties, a statistical selection procedure which hopefully can address all the above issues is needed to work together with MOEA. The MOCBA procedure presented in [28] is such a statistical selection procedure specially developed for the above purposes.

2.2 The Statistical Selection Procedure – MOCBA Procedure

When the solution space (Θ) is finite and relatively small, formulation (1) becomes the MORS problem: Given a set of n design alternatives with H performance measures which are evaluated through simulation, how to determine an optimal allocation of the simulation replications to the designs so that the non-dominated set of designs can be found at the least expense in terms of simulation replications. In this section, we present a brief description of a solution framework developed for solving this problem: the Multi-objective Optimal Computing Budget Allocation (MOCBA) algorithm. For more details on how MOCBA works, please refer to [28].

2.2.1 A Performance Index to Measure the Non-dominated Designs

Suppose we have a set of designs $i(i = 1, 2, ..., n)$, each of which is evaluated in terms of H performance measures $\mu_{ik}(k = 1, 2, ..., H)$ through simulation. Within the Bayesian framework, μ_{ik} is a random variable whose posterior distribution can be derived based

on its prior distribution and the simulation output [27]. We use the following performance index to measure how non-dominated design i is:

$$\psi_i = \bigcap_{j \neq i} [1 - P(\mu_j \prec \mu_i)]. \tag{2}$$

where $P(\mu_j \prec \mu_i)$ represents the probability that design j dominates design i. Under the condition that the performance measures are independent from one another and they follow continuous distributions, we have

$$P(\mu_j \prec \mu_i) = \prod_{k=1}^{H} P(\mu_{jk} \leq \mu_{ik}). \tag{3}$$

Performance index ψ_i measures the probability that design i is non-dominated by all the other designs. At the end of simulation, all designs in the Pareto set should have ψ_i close to 1, and those designs outside of the Pareto set should have ψ_i close to 0, because they are dominated. ψ_i can be estimated by the following two bounds:

$$\prod_{j \in S, j \neq i} \left[1 - \prod_{k=1}^{H} P(\mu_{jk} \leq \mu_{ik}) \right] \leq \psi_i \leq \min_{j \in S, j \neq i} \left[1 - \prod_{k=1}^{H} P(\mu_{jk} \leq \mu_{ik}) \right]. \tag{4}$$

2.2.2 Construction of the Observed Pareto Set

In the computing budget allocation process, the Pareto set is constructed based on observed performance. Therefore we call it the observed Pareto set (S_p). At a certain stage of the allocation process, the observed Pareto set can be constructed as described below. Given $\bar{\mu}_{ik}$ is the sample mean of the kth objective of design i, then design j dominates design i by observation, denoted by $j \widehat{\prec} i$, if the following condition holds with at least one inequality being strict:

$$\overline{\mu}_{jk} \leq \overline{\mu}_{ik} \forall k = 1, 2, ..., H. \tag{5}$$

The observed Pareto set is then constructed by putting those designs i into the Pareto set if we can not find a design j such that $j \widehat{\prec} i$. The rest of the designs are then put into the observed non-Pareto set ($\overline{S}_p$).

2.2.3 Two Types of Errors of the Observed Pareto Set

The quality of the observed Pareto set depends on whether designs in S_p are all non-dominated and designs outside S_p are all dominated. We can evaluate it by two types of errors: Type I error (e_1) and Type II error (e_2) as defined below.

Type I error (e_1) is defined as the probability that at least one design in the observed non-Pareto set is non-dominated; while Type II error (e_2) is defined as the probability that at least one design in the observed Pareto set is dominated by other designs. When both types of errors approach 0, the true Pareto set is found. The two types of errors can be bounded by the approximated errors ae_1 and ae_2 respectively as given below.

$$e_1 \leq ae_1 = \sum_{i \in \overline{S}_p} \psi_i. \tag{6}$$

$$e_2 \leq ae_2 = \sum_{i \in S_p} (1 - \psi_i). \tag{7}$$

When the noise in the simulation output is high, ae_1 and ae_2 can be large. However, once the observed Pareto set approaches the true Pareto set, both ae_1 and ae_2 approach 0, as ψ_i approaches 1 for $i \in S_p$ and ψ_i approaches 0 for $i \in \overline{S}_p$.

2.2.4 The Asymptotic Allocation Rules and a Sequential Solution Procedure

To get the true Pareto set with high probability, we need to minimize both Type I and Type II errors. In the MOCBA algorithm of [28], it is realized by iteratively allocating the simulation replications until some termination conditions are met according to some asymptotic allocation rules stated below.

Rule 1: Given a maximum total number of simulation replications R_{max} to be allocated among n competing designs with H objectives, whose performance for kth objective is described by random variables with sample means $\overline{\mu}_{1k}, \overline{\mu}_{2k}, ..., \overline{\mu}_{nk}$ and finite sample variances $\hat{\sigma}^2_{1k}, \hat{\sigma}^2_{2k}, ..., \hat{\sigma}^n_{1k}$, suppose S, S_p and $\bar{S_p} = S \backslash S_p$ represent the design space, observed Pareto and non-Pareto set respectively, then as $R_{max} \to \infty$, the upper bound of Type I error (ub_1) can be asymptotically minimized when
For a design $l \in \bar{S_p}$,

$$\alpha_l = \frac{\beta_l}{\sum_{l \in \bar{S_p}} \beta_l + \sum_{d \in S_p} \beta_d}$$

For a design $d \in S_p$,

$$\alpha_d = \frac{\beta_d}{\sum_{l \in \bar{S_p}} \beta_l + \sum_{d \in S_p} \beta_d}$$

where $\beta_l = \frac{\alpha_l}{\alpha_m} = \frac{\left(\hat{\sigma}^2_{lk^l_{j_l}} + \hat{\sigma}^2_{j_l k^l_{j_l}} / \rho_l\right) / \delta^2_{l j_l k^l_{j_l}}}{\left(\hat{\sigma}^2_{mk^m_{jm}} + \hat{\sigma}^2_{jmk^m_{jm}} / \rho_m\right) / \delta^2_{mjmk^m_{jm}}}$, given that m is any fixed design in $\bar{S_p}$;

$\beta_d = \frac{\alpha_d}{\alpha_m} = \sqrt{\sum_{i \in \Omega_d} \frac{\hat{\sigma}^2_{dk^i_d}}{\hat{\sigma}^2_{ik^i_d}} \beta_i^2}$, where α_i is the fraction of R_{max} to be allocated to design i;

$\delta_{ijk} = \bar{\mu}_{jk} - \bar{\mu}_{ik}$; $j_i \equiv arg \max_{j \in S, j \neq i} \prod_{k=1}^{H} P(\mu_{jk} \leq \mu_{ik})$; $k^i_{j_i} \equiv arg \min_{k=1,2,...,H} P(\mu_{jk} \leq \mu_{ik})$ with $P(\mu_{jk} \leq \mu_{ik})$ following $N\left(\delta_{ijk}, \frac{\hat{\sigma}^2_{jk}}{\alpha'_j N_{max}} + \frac{\hat{\sigma}^2_{ik}}{\alpha'_i N_{max}}\right)$, and α'_i is the fraction of R_{max} allocated to design i at the immediate previous iteration. $\Omega_d \equiv \{\text{design}i \mid i \in \bar{S_p}, j_i = d\}$; $\rho_i = \frac{\alpha'_{j_i}}{\alpha'_i}$ initially and after getting α_i and α_{j_i}, it is determined iteratively until it converges.

Rule 2: Under the same conditions and notations as given in **Rule 1**, the upper bound of Type II error (ub_2) can be asymptotically minimized when
For a design $l \in S_P^A$,

$$\alpha_l = \frac{\beta_l}{\sum_{l \in S_P^A} \beta_l + \sum_{u \in S_P^B} \beta_u + \sum_{d \in \bar{S_P}} \beta_d}$$

For a design $u \in S_P^B$,

$$\alpha_u = \frac{\beta_u}{\sum_{l \in S_P^A} \beta_l + \sum_{u \in S_P^B} \beta_u + \sum_{d \in \bar{S_P}} \beta_d}$$

For a design $d \in \bar{S_P}$,

$$\alpha_d = \frac{\beta_d}{\sum_{l \in S_P^A} \beta_l + \sum_{u \in S_P^B} \beta_u + \sum_{d \in \bar{S_P}} \beta_d}$$

where $\beta_l = \frac{\alpha_l}{\alpha_m} = \frac{\left(\hat{\sigma}^2_{lk^l_{j_l}} + \hat{\sigma}^2_{j_l k^l_{j_l}} / \rho_l\right) / \delta^2_{l j_l k^l_{j_l}}}{\left(\hat{\sigma}^2_{mk^m_{jm}} + \hat{\sigma}^2_{j_m k^m_{jm}} / \rho_m\right) / \delta^2_{m j_m k^m_{jm}}}$, given that m is any fixed design in S_p^A; $\beta_u = \frac{\alpha_u}{\alpha_m} = \sqrt{\sum_{i \in \Theta_u^*} \frac{\hat{\sigma}^2_{uk^i_u}}{\hat{\sigma}^2_{ik^i_u}} \beta_i^2}$, and $\beta_d = \frac{\alpha_d}{\alpha_m} = \sqrt{\sum_{i \in \Theta_d^*} \frac{\hat{\sigma}^2_{dk^i_d}}{\hat{\sigma}^2_{ik^i_d}} \beta_i^2}$, where, in addition to notations given in **Rule 1**, $\Theta_d \equiv \{\text{design} i \mid i \in S_p, j_i = d\}$; $S_P^A \equiv \left\{ \text{design } l \in S_p \mid \frac{\delta^2_{l j_l k^l_{j_l}}}{\hat{\sigma}^2_{lk^l_{j_l}} / \alpha_l + \hat{\sigma}^2_{j_l k^l_{j_l}} / \alpha_{j_l}} < min_{i \in \Theta_l} \frac{\delta^2_{ilk^i_l}}{\hat{\sigma}^2_{ik^i_l} / \alpha_i + \hat{\sigma}^2_{lk^i_l} / \alpha_l} \right\}$ with α_i being approximated by $\alpha_i^{'}$; $S_P^B = S_P \backslash S_P^A$; $\Theta_d^* \equiv \left\{\Theta_d \cap S_P^A\right\}$.

The MOCBA algorithm is now outlined below.

MOCBA algorithm

Step 0: Perform R_0 replications for each design. Calculate the sample mean and variance for each objective of the designs. Set iteration index $v := 0$. $R_1^v = R_2^v = \ldots = R_n^v = R_0$.

Step 1: Construct the observed Pareto set S_P as stated in Section 2.2.2; calculate ae_1 and ae_2 according to equations (6) and (7) respectively.

Step 2: If termination conditions are met, go to Step 7.

Step 3: Increase the simulation replications by a certain amount Δ. If $ae_1 \leq ae_2$, go to Step 5.

Step 4: Calculate the new allocation according to **Rule 1**. Go to Step 6.

Step 5: Calculate the new allocation according to **Rule 2**.

Step 6: Perform additional $\min\left(\delta, \max\left(0, R_i^{v+1} - R_i^v\right)\right)$ replications for design i ($i = 1,,n$). Update the sample mean and variance of each objective of design i based on cumulative simulation output. Set $v = v + 1$ and go to Step 1.

Step 7: Output designs in the observed Pareto set (S_p).

2.3 The Integrated MOEA Procedure - Integration of MOEA with MOCBA

Through the integration of MOEA with MOCBA, we expect that the integrated MOEA procedure is capable of overcoming all limitations of MOEA stated in Section 2.1. A flowchart depicting how MOCBA is integrated into MOEA is illustrated in Figure 2. Here MOCBA's involvement in the integrated procedure takes place through three general MOEA steps: performance evaluation, fitness assignment and formation of the Pareto set.

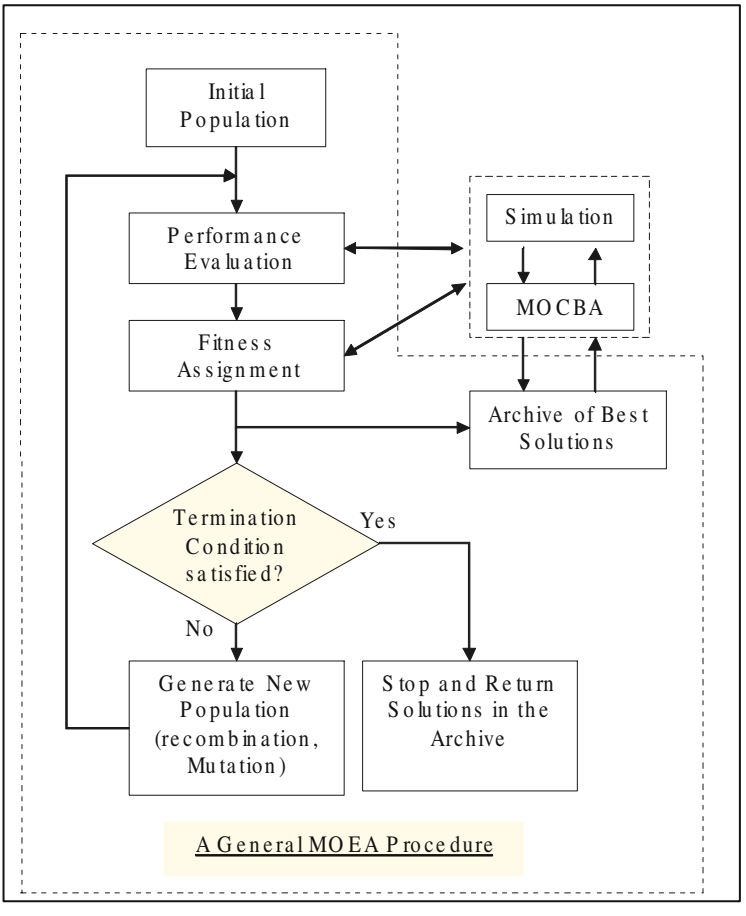


Fig. 2. Flowchart for the Integrated MOEA Procedure

2.3.1 Performance Evaluation

In any search procedure, search direction determines how the current solution(s) move to new solution(s). If performance evaluation is inaccurate, search direction would be misguided and therefore the overall performance of the search procedure deteriorates. In the MSO problem considered here, first and foremost, we are to determine how many simulation replications need to be run for each individual of the population. Certainly more simulation replications would definitely result in more accurate estimation of the performance measures. Nevertheless, as a population based search algorithm, MOEA involves visiting a large number of solutions while exploring the solution space. If simulation replications are to be uniformly allocated among the individuals and performance measures are to be estimated with high accuracy, the total simulation cost can easily become prohibitive. Hence we need to determine the right number of replications for each individual in a more intelligent manner. Intuitively, for individuals with poor

performances which are obviously dominated by others, it is a waste if further simulation replications are allocated to them, though the noise involved in the performance measures maybe still high. Thus it is economically beneficial to control the noise in performance measures of each individual at the right level. This can be done by only running more replications (therefore lower the noise level) for those designs which are likely to be competitors for the "best". The MOCBA algorithm stated in Section 2.2 can help to optimally determine the right number of simulation replications needed for each individual. Specifically, at each generation of MOEA, we apply MOCBA to smartly and efficiently allocate simulation replications to each individual of the current population so that performance measures of each individual of the population are estimated with the right accuracy.

2.3.2 Fitness Assignment

Fitness evaluation and assignment mechanism is an important component and the guiding system of MOEA. This is because, calculated based on the performance measures, fitness value defines the relative strength of an individual. At every generation of MOEA, it is based on the fitness values that parents are selected to perform recombination and mutation to generate new population. A key point in fitness assignment is how to ensure that fitness truly reveals quality of the solution it represents. In case fitness assignment method is not appropriate, less fit solutions may be selected and survive into the next generation, whereas truly fit solutions are neglected and lose the chance for further consideration. This will lead the search to less productive regions and eventually impair the overall performance of MOEA. In MOEA developed for deterministic problems, one common way to estimate the fitness of an individual is to first count the number of individuals in the current population by which it is dominated and then rank and assign fitness for each individual according to this number [22]. In our study, this fitness assignment method becomes unsuitable as the uncertainties involved in the performance measures would affect the dominance relationship among the individuals. To ensure that fitness of individuals can best represent the individuals' qualities and therefore it would properly guide the MOEA in high variability environment, we employ the performance index ψ_i defined in Section 2.2.1 to evaluate the fitness of the individuals. Performance index ψ_i measures the probabilistic dominance relationship among the individuals. ψ_i best serves the purpose here, as it not only considers the multi-objective nature of the individuals, but also takes uncertainty into account by including both variance and sample mean information into the fitness evaluation. Due to the difficulty in calculating ψ_i accurately, we use the lower bound of ψ_i in (4) to approximate it. Here, the MOCBA procedure also plays an important role in fitness evaluation step of MOEA, because ψ_i is calculated based on performance measures of the individuals and MOCBA helps to guarantee the performance measures of each individual estimated with the right accuracy by optimally allocating the simulation replications.

2.3.3 Formation of the Pareto Set

In MOEA developed for deterministic problems, elite population is commonly formed by keeping an archive of mutually non-dominated solutions at each iteration. The archive is updated over generations of MOEA by replacing dominated individuals with

newly found non-dominated ones according to the dominance relationships among the individuals. In our study, the uncertainties involved in the performance measures cause the dominance relationships among the individuals uncertain: it is not known, in a statistical sense, which individuals are non-dominated in the current population; it is also unknown, with what significance level, the individuals in the final Pareto set are non-dominated among all individuals visited during the MOEA. These difficulties also can be overcome through the use of performance index ψ_i and running of the MOCBA. At each MOEA generation, the MOCBA can identify non-dominated individuals in the current population and provide an estimation of the quality of the observed Pareto set in form of Type I and II errors. We can use individuals in the observed Pareto set as elite population. However, when simulation replications are limited, individuals in the observed Pareto set may not all be non-dominated with high probability. To form the elite population, we can select those individuals with performance index ψ_i greater than a certain value, say ψ_i^*. One thing to note here is that, when noise involved in the performance measures is high, even truly non-dominated individuals may not have high performance indices. Therefore it is wise to set ψ_i^* at a proper value: not too low to include too many dominated individuals into the elite population; not too high to miss truly non-dominated individuals in the elite population. At the termination of the MOEA, MOCBA is run again on the elite population to identify the final optimal Pareto set. In this run of MOCBA, we can set desired error limits ε^* for the Type I and II errors, as we intend to find the final Pareto non-dominated solutions with high confidence.

Though MOCBA is integrated with the MOEA procedure through the aforementioned three steps, it has great impact on other parts of MOEA as well, such as selection of parents to do recombination; selection of individual to do mutation; deletion of poor individuals from the current population, etc. A detailed integrated MOEA procedure is illustrated below.

2.3.4 Outline of the Integrated MOEA

Step 0: Initialization: Randomly generate an initial feasible population POP_t of size N_t; set elite population $E_t = \Phi$; set generation index $t = 0$.

Step 1: Run MOCBA (Section 2.2.4) to determine the number of replications for each individual in population POP_t, and form the observed Pareto set.

Step 2: Formation of elite population: Form the elite population E_t with individuals having performance index $\psi_i \geq \psi^*$.

Step 3: Check the termination condition. If it is not satisfied, go to Step 5.

Step 4: Termination: Run MOCBA on the Elite Population E_t with both types of errors within error limit ε^*. Output the Pareto set as the final set of non-dominated solutions.

Step 5: Evaluation and fitness assignment: Use performance index ψ_i at the termination of MOCBA as the fitness value of individual i.

Step 6: New population: Set $t = t + 1$; let $POP_t = POP_{t-1}$. Create a new population by repeating the following steps until M pairs of parents are selected.

a. Selection: select one pair of individuals by tournament selection from population POP_{t-1}.

b. Crossover: Generate one pair of offspring by performing certain crossover operators. Check the feasibility of the offspring.
c. Add the new offspring into population POP_t.

Step 7: Mutation: Run MOCBA on population POP_t to determine fitness value for the new offspring. If $N_t > N_{max}$, delete $(N_t - N_{max})$ individuals with least fitness value. For each individual i in POP_t, perform mutation with probability P_m. Check the feasibility of the mutated individual. Go to Step 1.

One merit of the above integrated MOEA procedure is its ease of integration: we need not tailor the MOEA for integration with the MOCBA. When solving a particular MSO problem, a special purpose EA procedure may be desired. The integrated procedure shows that the MOCBA procedure can be easily incorporated into any EA procedure, or any other population based search algorithm, such as the Nested Partitions method [40]. Another merit is its ease of implementation. In any search algorithm, when new solutions are generated, we need to evaluate their performance measures. MOCBA is only needed to run when performance of the new individuals are to be evaluated. And with one run of MOCBA, you can get all necessary information simultaneously: the right number of simulation replications for each individual; the fitness (performance index) of each individual; the observed Pareto set of non-dominated solutions; and the estimated quality of the Pareto set (Type I and II errors).

3 Experimental Results

In this section, we test the effectiveness and efficiency of the integrated MOEA framework. When MOEA is applied to solve MSO problems, the commonly used approach to addressing noise explicitly is to generate equal number of samples for each solution and use the average to represent the performance of the solutions. We call this MOEA with uniform computing budget allocation MOEA_UCBA; similarly the proposed integrated MOEA framework in this study is called MOEA_MOCBA. To quantify how much MOEA_MOCBA can improve over MOEA_UCBA, they are applied to solve a test problem with known Pareto-optimal set (Section 3.1) and compared through a number of performance metrics (Section 3.2).

3.1 The Test Problem

The test problem considered here is a two-objective minimization problem proposed in [41]:

$$f_1(x_1, x_2, ..., x_n) = 1 - exp\left(-\sum_{i=1}^{n}(x_i - 1/\sqrt{n})^2\right)$$

$$f_2(x_1, x_2, ..., x_n) = 1 - exp\left(-\sum_{i=1}^{n}(x_i + 1/\sqrt{n})^2\right)$$

where the decision variables x_i, $i = 1, 2, ..., n$ take on real values and the number of decision variables n can be arbitrary. The Pareto-optimal set S_p of this problem corresponds to all points on the line defined by:

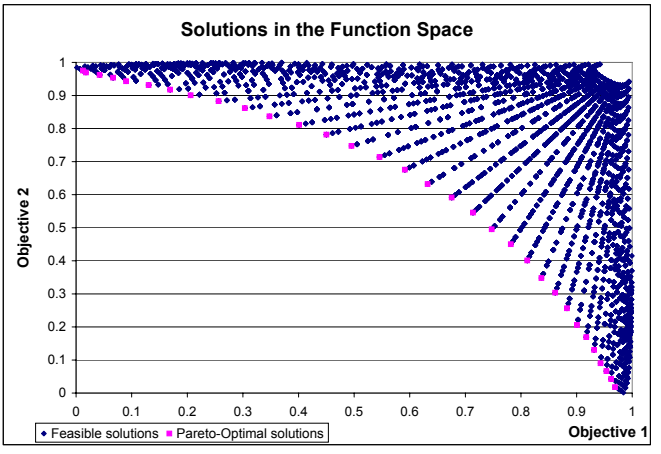


Fig. 3. Illustration of solutions in the function space

$$S_p \equiv \left\{(x_1, x_2, ..., x_n) | x_1 = x_2 = ... = x_n \in \left[-1/\sqrt{n} \leq x_1 \leq 1/\sqrt{n}\right]\right\}$$

In this study, we assume that the number of decision variables n is set to 3, and the decision variables are bounded with $x_i \in [-1,1]\ \forall\ i = 1,2,3$. To change it into a stochastic problem, we add a normally distributed iid noise with zero mean and a standard deviation of σ, $N\left(0,\sigma^2\right)$, to the two objective functions. Then the problem is discretized in such a way that the decision variables $x_i\ \forall\ i = 1,2,3$ are set to have one decimal place $(-0.1, 0.0, 0.1, 0.2, ...)$. This would create a search space with 9261 designs. After the problem is discretized, the Pareto-optimal set S_p of the discrete problem corresponds to all points satisfying the following condition:

$$S_p \equiv \left\{(x_1, x_2, x_3) | x_1 = x_2 = x_3 \in \left[-1/\sqrt{3} \leq x_1 \leq 1/\sqrt{3}\right], \max_{i=1,2,3} x_i - \min_{i=1,2,3} x_i \leq 0.1\right\}$$

As a result, in our test problem, there are 85 solutions in the Pareto-optimal set. However, due to the symmetric nature of the test problem, different set of decision variables may produce the same objective values. Therefore only 37 of the Pareto-optimal solutions are distinct. An illustration of the solutions in the function space in presented in Figure 3.

3.2 Comparison of MOEA_MOCBA and MOEA_UCBA

In both MOEA_MOCBA and MOEA_UCBA, the MOEA procedure basically follows steps as given in Section 2.3 with some problem specific information as given below:

- The MOEA used here is real parameter genetic algorithm.
- The coding scheme defines each chromosome as n genes x_i $(i = 1, 2, ..., n)$, each of which represents the value of decision variable x_i $(i = 1, 2, ..., n)$.

- Crossover is done according to the blend crossover with $\alpha = 0.5$. For two parent solutions $x_i^{(1,t)}$ and $x_i^{(2,t)}$ (ith gene at tth generation), given u_i is a random number between 0 and 1, an offspring would be generated as $x_i^{(1,t+1)} = (1-\gamma_i)x_i^{(1,t)} + \gamma_i x_i^{(2,t)}$, where $\gamma_i = (1+2\alpha)u_i - \alpha$.
- Mutation is randomly applied to a gene in a solution with probability P_m. When mutation takes place, a random value within the range of the gene will be generated and be used to replace the existing gene.
- The termination condition is defined as: A fixed number of MOEA generations.

When comparing different MOEA algorithms, some commonly used performance metrics are: solution quality when computational time is fixed; computational time with given solution quality; convergence of the algorithm; and diversity of the resulted non-dominated set. In this section, we also employ these performance metrics but with some modifications due to the fact that simulation is used to evaluate the performance of the solutions. Since simulation can be very time-consuming, we ignore time spent on MOEA and consider computational time as simulation budget needed. Specifically, we will compare the two algorithms in the following manner depending on how we manage to terminate MOCBA or UCBA at each MOEA generation: 1) Fix the total amount of computing budget available; 2) Fix the performance estimation accuracy of the solutions. In the former case, at each MOEA generation, MOCBA (UCBA) is run until total computing budget is consumed, and the comparison is done in terms of the quality (percentage of true Pareto-optimal solutions identified) of the elite population; how close the elite population is to the true Pareto-optimal solutions (convergence), and how evenly solutions in the elite population are distributed along the Pareto front (diversity). In the latter case, MOCBA (UCBA) is run until performance indices for solutions in and outside of the Pareto set reach certain value, and the comparison is done in terms of the total number of simulation replications consumed.

3.2.1 Results with Fixed Computing Budget

In this case, the noise added to the test problem is set as following normal distribution: $N\left(0, 0.03^2\right)$. Also the simulation budget available for each MOEA generation is fixed to be 1000. For the integrated MOEA procedure, we set the parameters involved as follows. For MOCBA: initial number of replications $R_0 = 10$; incremental number of replications $\Delta = 10$; maximum of additional replications $\delta = 5$. For the MOEA: population size is fixed as $N_t = N_{max} = 100$; mutation probability $P_m = 0.05$; performance index for selecting elite population $\psi^* = 0.9$; tournament selection parameter: $m = 2, M = 5$; total number of MOEA generations $T = 30$.

Figure 4 displays the percentage of true Pareto-optimal solutions identified in the elite population of both MOEA_MOCBA and MOEA_UCBA. We can observe that, throughout the MOEA generations, MOCBA can help MOEA to identify about 10% more true Pareto-optimal solutions. Moreover, MOEA_UCBA exhibits earlier convergence to local optimal solutions (at about generation 12) than MOEA_MOCBA (at about generation 20).

As the quality of the elite population also depends on how close other solutions are to the true Pareto-optimal solutions, and how the solutions in the elite set distribute

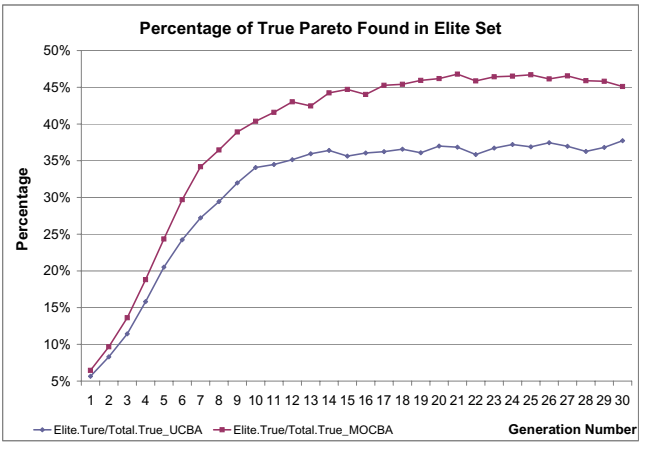


Fig. 4. Percentage of True Pareto-optimal Solutions in the Elite Set

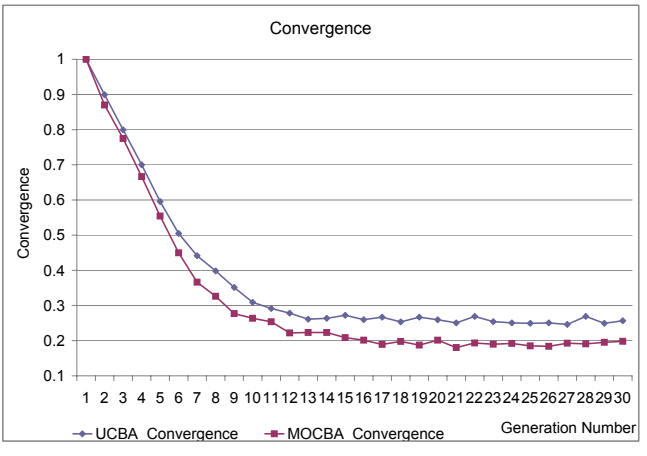


Fig. 5. Convergence Metric of the Elite Set

along the Pareto front, we employ the ”convergence” and ”diversity” metrics developed in [42] to compare the performance of MOEA_MOCBA and MOEA_UCBA. Results are shown in Figure 5 (convergence) and Figure 6 and 7 (diversity) respectively. In terms of convergence, we can observe from Figure 5 that, at the beginning of the MOEA, solutions in the elite set generated from both algorithms are far away from the Pareto-optimal solutions. As generation increases, elite population of both algorithms approaches true Pareto-optimal solutions, with MOEA_MOCBA consistently capable of producing solutions even closer to the true Pareto-optimal set.

In terms of diversity, one observation from Figures 6 and 7 is that, even at the beginning of the MOEA procedure, solutions in the elite set spread quite well (with diversity

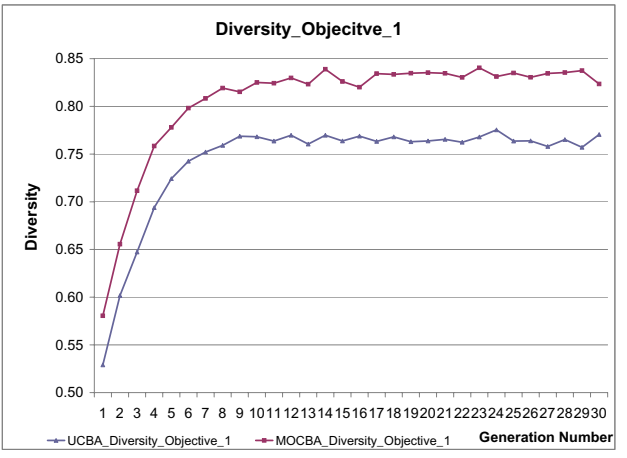


Fig. 6. Diversity Metric of the 1st objective of the Elite Set

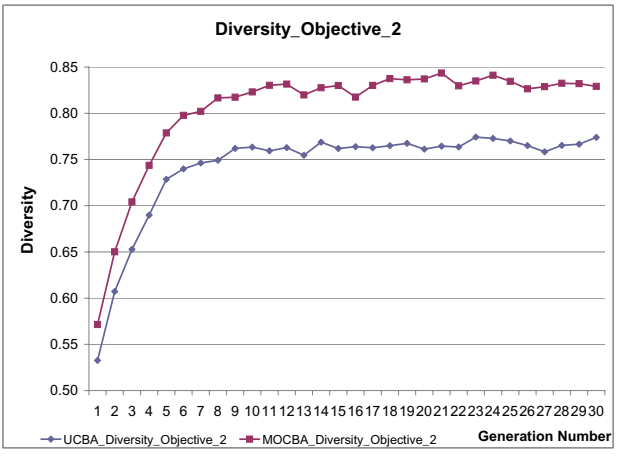


Fig. 7. Diversity Metric of the 2nd objective of the Elite Set

metric being 0.5 and above) for both objectives though the solutions may not be close to the true Pareto-optimal set (with convergence metric being 1). This may be due to the fact that all feasible solutions in the test problem are relatively uniformly distributed within the solution space and the initial population is generated randomly. Another observation is that, the diversity metrics obtained from both objectives are quite similar; this is possibly due to the symmetric nature of the test problem. Moreover, as generation increases, diversity of the solutions in elite set becomes better for both objectives but it can not be further improved after about 10 generations. Throughout the MOEA

procedure, MOEA_MOCBA can consistently obtain solutions more evenly distributed along each objective than MOEA_UCBA.

One other observation from Figures 4 to 7 is that, after about 30 MOEA generations, solutions in the elite set cannot be further improved in terms of true Pareto-optimal solutions found, convergence and diversity, this maybe resulted from the fact that the MOEA algorithm employed here is rather primitive and simple. However, we do observe improvements of MOCBA over UCBA from all performance metrics considered, which justified our motivation of doing this research in this study.

3.2.2 Results with Fixed Estimation Accuracy

When performance estimation accuracy to be reached is fixed at each generation of both algorithms, our purpose is to study how much more simulation budget MOEA_UCBA needs than MOEA_MOCBA. In this case, we assume that noise added to the test problem following normal distribution: $N\left(0,0.01^2\right)$; and at each intermediate generation, MOCBA (UCBA) is terminated when designs in the observed Pareto (non-Pareto) set have performance index greater (less) than 0.6 (0.4). Most parameters of the integrated MOEA procedure are set as the same values as when computing budget is fixed, except for the following: population size is fixed as $N_t = N_{max} = 40$; total number of MOEA generations $T = 20$; error limit for both types of errors of the final Pareto set $\varepsilon^* = 0.3$. Results are shown in Figure 8.

Figure 8 illustrates that, when estimation accuracy to be reached is fixed at each MOEA generation, MOCBA can save at least half of the simulation budget UCBA consumes. Moreover, simulation budget needed by UCBA increases at a much faster rate than that of MOCBA as generation increases. This improvement of MOEA_MOCBA over MOEA_UCBA is impressive, as evaluation of performance through simulation can be very costly and time-consuming especially when real-life complex systems are to be optimized.

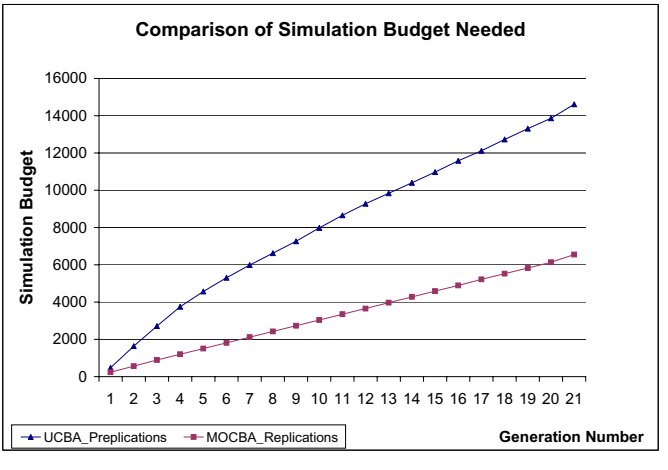


Fig. 8. Comparison of Simulation Budget Needed

4 Conclusion

In this chapter, we present a solution framework for solving the multi-objective simulation optimization problem. The solution framework is an integration of MOEA (a search procedure) and MOCBA (a statistical selection procedure). One merit of the framework is its ease of integration: MOEA can be easily changed to any special purpose EA procedure or any other population based search procedures. To test its performance, the integrated MOEA is applied to solve one benchmark test problem with known Pareto-optimal set. Computational results show that, for the test problem, MOCBA can help to improve performance of MOEA in terms of several performance metrics considered: solution quality, simulation budget, as well as convergence and diversity of solutions in the elite set. In the current study, the main focus is on how MOCBA can help to improve performance of MOEA by properly allocating the simulation replications and identifying the Pareto set. In future research, it is important to study the integration itself in more detail. Specifically similar to studies in the single objective case: we need to investigate what information regarding ranking of the individuals (which pair-wise comparisons) is necessary for the proper function of MOEA, and how to adapt MOCBA in such a way that it can more efficiently generate the key information required by MOEA.

References

1. Mollaghasemi, M., Evans, G.W.: Multicriteria design of manufacturing systems through simulation optimization. IEEE Transactions on Systems, Man, and Cybernetics 24, 1407–1411 (1994)
2. Pritsker, A.: Introduction to simulation and SLAM II. John Wiley and Sons, New York (1986)
3. Swisher, J.R., Jacobson, S.H., Yücesan, E.: Discrete-event simulation optimization using ranking, selection, and multiple comparison procedures: A survey. ACM Transactions on Modeling and Computer Simulation 13, 134–154 (2003)
4. Nelson, B.L., Swann, J., Goldsman, D., Song, W.M.: Simple procedures for selecting the best simulated system when the number of alternatives is large. Operations Research 49, 950–963 (2001)
5. Chen, C.H., Lin, J.W., Yücesan, E., Chick, S.E.: Simulation budget allocation for further enhancing the efficiency of ordinal optimization. Discrete Event Dynamic Systems: Theory and Applications 10, 251–270 (2000)
6. Chick, S.E., Inoue, K.: New two-stage and sequential procedures for selecting the best simulated system. Operations Research 49, 732–743 (2001)
7. Boesel, J., Nelson, B.L., Ishii, N.: A framework for simulation-optimization software. IIE Transactions 35, 221–229 (2003)
8. Hedlund, H.E., Mollaghasemi, M.: A genetic algorithm and an indifference-zone ranking and selection framework for simulation optimization. In: Proceedings of the 2001 Winter Simulation Conference, pp. 417–421 (2001)
9. Alkhamis, T.M., Ahmed, M.A.: Simulation-based optimization using simulated annealing with confidence interval. In: Proceedings of the 2004 Winter Simulation Conference, pp. 514–519 (2004)
10. Ahmed, M.A., Alkhamis, T.M.: Simulation-based optimization using simulated annealing with ranking and selection. Computers and Operations Research 29, 387–402 (2002)
11. Ólafsson, S.: Iterative ranking-and-selection for large-scale optimization. In: Proceedings of the, Winter Simulation Conference, pp. 479–485 (1999)

12. Chen, C.H., He, D., Fu, M., Lee, L.H.: Efficient selection of an optimal subset for optimization under uncertainty. In: Proceedings of the 2007 INFORMS Simulation Society Research Workshop: Simulation in Decision Making, July 5-7, INSEAD, Fontainebleau, France (2007)
13. Steuer, R.E., Choo, E.U.: An interactive weighted Tchebycheff procedure for multiple objective programming. Mathematical Programming 26, 326–344 (1983)
14. Wierzbicki, A.P.: A mathematical basis for satisficing decision making. Mathematical Modelling 3, 391–405 (1982)
15. Wierzbicki, A.P.: On the completeness and constructiveness of parametric characterizations to vector optimization problems. OR Spectrum 8, 73–87 (1986)
16. Benayoun, R., de Montgolfier, J., Tergny, J., Larichev, O.: Linear Programming with Multiple Objective Functions: Step Method (STEM). Mathematical Programming 1, 366–375 (1972)
17. Benson, H.P., Sayin, S.: Towards Finding Global Representations of the Efficient Set in Multiple Objective Mathematical Programming. Naval Research Logistics 44, 47–67 (1997)
18. Climaco, J., Antunes, C.: Implementation of a User-Friendly Software Package—A Guided Tour of TRIMAP. Mathematical and Computer Modelling 12, 1299–1309 (1989)
19. Jaszkiewicz, A., Slowinski, R.: The Light Beam Search Approach: An Overview of Methodology and Applications. European Journal of Operational Research 113, 300–314 (1999)
20. Korhonen, P., Wallenius, J.: A Pareto Race. Naval Research Logistics 35, 277–287 (1988)
21. Deb, K.: Multiobjective Optimization Using Evolutionary Algorithms. Wiley, Chichester (2001)
22. 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, pp. 416–423. University of Illinois at Urbana-Champaign. Morgan Kauman Publishers, San Mateo (1993)
23. Horn, J., Nafpliotis, N., Goldberg, D.E.: A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In: Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, pp. 82–87. IEEE Service Center, Piscataway (1994)
24. Srinivas, N., Deb, K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation 2(3), 221–248 (1994)
25. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)
26. Coello Coello, C.A.: A short tutorial on evolutionary multiobjective optimization. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, p. 21. Springer, Heidelberg (2001)
27. Lee, L.H., Chew, E.P., Teng, S.Y., Goldsman, D.: Optimal computing budget allocation for multi-objective simulation models. In: Ingalls, R.G., Rossetti, M.D., Smith, J.S., Peters, B.A. (eds.) Proceedings of the 2004 Winter Simulation Conference, pp. 586–594 (2004)
28. Lee, L.H., Chew, E.P., Teng, S.Y., Goldsman, D.: Finding the non-dominated Pareto set for multi-objective simulation models. Submitted to IIE Transactions (2005)
29. Baesler, F.F., Sepúlveda, J.A.: Multi-response simulation optimization using stochastic genetic search within a goal programming framework. In: Joines, J.A., Barton, R.R., Kang, K., Fishwick, P.A. (eds.) Proceedings of the 2000 Winter Simulation Conference, pp. 788–794 (2000)
30. Butler, J., Morrice, D.J., Mullarkey, P.W.: A multiple attribute utility theory approach to ranking and selection. Management Science 47(6), 800–816 (2001)
31. Hammel, U., Bäck, T.: Evolution strategies on noisy functions, how to improve convergence properties. In: Davidor, Y., Männer, R., Schwefel, H.-P. (eds.) PPSN 1994. LNCS, vol. 866. Springer, Heidelberg (1994)

32. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - a survey. IEEE Transactions on Evolutionary Computation 9(3), 303–317 (2005)
33. Lee, L.H., Lee, C.U., Tan, Y.P.: A multi-objective genetic algorithm for robust flight scheduling using simulation. European Journal of Operational Research 177(3), 1948–1968 (2007)
34. Hughes, E.J.: Evolutionary multi-objective ranking with uncertainty and noise. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, p. 329. Springer, Heidelberg (2001)
35. Goh, C.K., Tan, K.C.: An investigation on noisy environments in evolutionary multiobjective optimization. IEEE Transactions on Evolutionary Computation 11(3), 354–381 (2007)
36. Schaffer, J.D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In: Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms, pp. 93–100. Lawrence Erlbaum, Mahwah (1985)
37. Knowles, J.D., Corne, D.W.: Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. Evolutionary Computation 8(2), 149–172 (2000)
38. Tan, K.C., Lee, T.H., Khor, E.F.: Evolutionary Algorithms With Dynamic Population Size and Local Exploration for Multiobjective Optimization. IEEE Transactions on Evolutionary Computation 5 (6), 565–588
39. Praditwong, K., Yao, X.: A new multi-objective evolutionary optimisation algorithm: The two-archive algorithm. In: Wang, Y., Cheung, Y.-m., Liu, H. (eds.) CIS 2006. LNCS (LNAI), vol. 4456, pp. 95–104. Springer, Heidelberg (2007)
40. Shi, L., Ólafsson, S.: Nested partitions method for global optimization. Operations Research 48(3), 390–407 (2000)
41. Fonseca, C.M., Fleming, P.J.: Multiobjective genetic algorithms made easy: selection, sharing and mating restriction. In: Proceedings of the, international conference on Genetic algorithms in engineering systems: innovations and applications, pp. 45–52 (1995)
42. Deb, K., Jain, S.: Running performance metrics for evolutionary multi-objective optimizaiton. KanGal Report No. 2002004 (2002)

Feature Selection Using Single/Multi-Objective Memetic Frameworks

Zexuan Zhu[1], Yew-Soon Ong[1], and Jer-Lai Kuo[2]

[1] School of Computer Engineering, Nanyang Technological University, Singapore
zhuzexuan@pmail.ntu.edu.sg, asysong@ntu.edu.sg
[2] School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore
jlkuo@ntu.edu.sg

Memetic frameworks for the hybridization of wrapper and filter feature selection methods have been proposed for classification problems. The frameworks incorporate filter methods in the traditional evolutionary algorithms to improve classification performance while accelerating the search in the identification of crucial feature subsets. Filter methods are introduced as local learning procedures in the evolutionary search to add or delete features from the chromosome which encodes the selected feature subset. Both single/multi-objective memetic frameworks are described in this chapter. Single objective memetic framework is shown to speedup the identification of optimal feature subset while at the same time maintaining good prediction accuracy. Subsequently, the multiobjective memetic framework extends the notion of optimal feature subset as the simultaneous identification of full class relevant (FCR) and partial class relevant (PCR) features in multiclass problems. Comparison study to existing state-of-the-art filter and wrapper methods, and the standard genetic algorithm highlights the efficacy of the memetic framework in facilitating a good compromise of the classification accuracy and selected feature size on binary and multi class problems.

1 Introduction

Feature selection has attracted increasing research interests in many application domains in recent years. With the rapid advance of computer and database technologies, datasets with hundreds and thousands of variables or features are now ubiquitous in pattern recognition, data mining, and machine learning [1, 2]. To process such high dimensional dataset is a challenging task for traditional machine learning. Feature selection addresses this problem by removing the irrelevant, redundant, or noisy features. It improves the performance of the learning algorithms, reduces the computational cost, and provides better understandings of the datasets. Feature selection has been widely used in many cutting-edge research areas. For instance, it has been applied on cancer diagnosis using microarray data [3, 4, 5, 6], which is characterized with thousands of genes. A significant amount of new discoveries have been made and new biomarkers for various cancer have been detected from the microarray data analysis. Feature selection is also used on the synergy of new nanoscale materials [7, 8], where it identifies the

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 111–131.
springerlink.com

critical multisite interactions in Ising model to map direct quantum-mechanical evaluation and provides accurate prediction on the properties of new materials.

Feature selection algorithms are widely categorized into two groups: filter and wrapper methods [9]. Filter methods evaluate the goodness of the feature subset by using the intrinsic characteristic of the data. They are computationally inexpensive since they do not involve the use of any induction algorithm. However, they take the risk of selecting subsets of features that may not match the induction algorithm chosen in the actual prediction. Wrapper methods, on the contrary, directly use the induction algorithm to evaluate the feature subsets. They generally outperform filter methods in terms of prediction accuracy, but are relatively more computationally intensive.

Another key issue for feature selection algorithm is the curse of dimensionality that causes significant difficulties in searching for the optimal feature subsets which is capable of generating accurate predictions. This fundamental difficulty arises from the fact that the number of hypercubes required to fill out a compact region of a N-dimensional space grows exponentially with N. As it is well established that Genetic algorithm (GA) [10] is capable of producing high quality solutions within tractable time even on complex problems, it has been naturally used for feature selection and promising performance has been reported in the recent years [5, 6, 11, 12, 13, 14, 15, 7, 8]. Unfortunately, due to the inherent nature of GA, it often takes a long time to locate the local optimum in a region of convergence and sometimes may not find the optimum with sufficient precision.

Up to date, the question of which scheme one should use for solving a given new problem, i.e., Filter or Wrapper, remains debatable. To address the aforementioned problems, memetic algorithm (MA) frameworks that synergizes filter and wrapper methods have recently been proposed for feature selection [13, 14, 15]. Memetic framework thus complements the strengths of wrapper and filter methods towards more efficient and effective search. A single objective hybridization of filter method based local search and GA is first proposed by Zhu et al. [13, 14] for general feature selection problem, i.e., selecting minimal feature subset while obtaining best prediction accuracy[1]. In particular, the filter method based local search is introduced to add or delete features from a genetic algorithm solution so as to quickly improve the solution and fine-tune the search.

Many class prediction problem or otherwise often known as multiclass prediction problem represents a research field of feature selection that has drawn increasing interests recently. The field poses a bigger challenge due to the significant statistical and analytical implications involved over the binary counterparts. With multiclass feature selection approaches, it is now possible to identify whether the features are relevant to the set of classes considered. Nevertheless, instead of merely to identify the relevant features to the set of multiple classes, it is generally more informative if the relevance of each feature to specific classes or subset of classes is also revealed. To pinpoint the specific classes a feature is relevant to, the concepts of feature relevance, namely, i) full

[1] Feature selection is naturally a multiobjective optimization problem of minimizing the number of selected features while maximizing the prediction accuracy. In practice, the two objectives are always aggregated into a single one using aggregating function method, and the prediction accuracy is always given higher priority.

class relevant (FCR) and ii) partial class relevant (PCR) are introduced by Zhu et al. in [15]. FCR denotes features that serve as candidates for discriminating any classes. PCR, on the other hand, represents features that serve to distinguish subsets of classes. Subsequently, the multiobjective memetic framework for simultaneous identification of both FCR and PCR features is introduced in [15] for multiclass classification, where each objective corresponding to the search for an optimal PCR feature subset. The approximate Pareto optimal set [16] of solutions representing distinct options for solving the multiobjective optimization problem based on different tradeoffs of the objectives is then obtained with a single simulation run of the algorithm. The extremal solution fo each objective then represents the corresponding PCR feature subset. The FCR feature subset is then formed by the intersection of all solutions.

The rest of this chapter is organized as follows: Section 2 presents the background knowledge of feature selection, relevance, redundancy, and the FCR/PCR features. Section 3 presents the single objective memetic algorithm for feature selection. Section 4 presents the problem formulation of FCR/PCR feature identification and the details of the proposed multiobjective memetic algorithm. Section 5 presents the experimental results and some discussions on benchmark datasets. Finally, Section 6 summarizes this study.

2 Background

This section begins with a brief introduction on the background on feature selection, feature relevance, and feature redundancy. Next, the notions and definitions of FCR and PCR features are discussed.

2.1 Feature Selection

Feature selection involves the problem of selecting a minimal subset of M features from the original set of N features ($M \leq N$), so that the feature space is optimally reduced and the performance of the learning algorithm is improved or not significantly

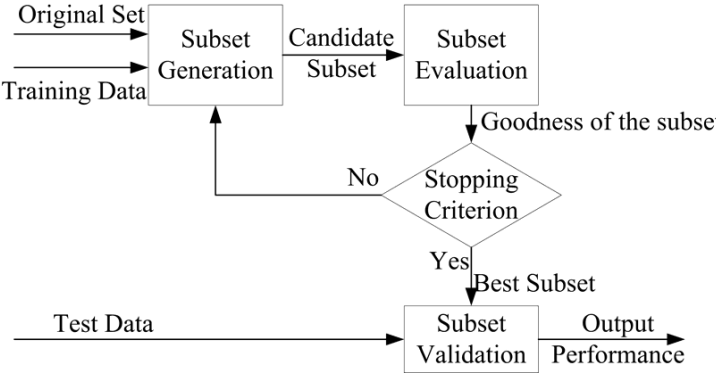


Fig. 1. Feature selection procedure

decreased [1, 17, 18, 19]. Generally, a typical feature selection method consists of four components: a generation procedure or search procedure, evaluation function, stopping criterion, and validation procedure. Every time a candidate subset of features is generated randomly or based on some heuristics, it is evaluated based on some independent (without involving any induction algorithm, i.e., filter method) or dependent criteria (performance of the induction algorithm, i.e., wrapper method). This procedure of feature selection is repeated until a predefined stopping condition is satisfied, which may be due to the completion of search, an achievement of a sufficiently good subset, or violation of the maximum allowable number of iterations. Next, the selected subset should be validated either using prior knowledge about the data or using some unseen independent testing datasets. This general process of feature selection is illustrated in Figure 1.

2.2 Feature Relevance and Redundancy

Let $C = \{c_1, \ldots, c_k\}$ be the class set to be considered, X be a full set of features, X_i be a feature, and $\overline{X}_i$ be the set of all features that excludes X_i, i.e., $\overline{X}_i = X - \{X_i\}$. Based on the definition given in [9, 20], features can be categorized as strongly relevant, weakly relevant, or irrelevant.

Definition 1. Strong Relevance: A feature X_i is strongly relevant if and only if $P(C|X_i, \overline{X}_i) \neq P(C|\overline{X}_i)$.

Definition 2. Weak Relevance: A feature X_i is weakly relevant if and only if $P(C|X_i, \overline{X}_i) = P(C|\overline{X}_i)$ and $\exists \overline{X}'_i \subseteq \overline{X}_i$ such that $P(C|X_i, \overline{X}'_i \neq P(C|\overline{X}'_i)$.

A feature **relevant** to the learning classes C can pose as strongly relevant or weakly relevant, otherwise, it is regarded as **irrelevant** to C. An optimal feature subset contains all the strongly relevant features and some weakly relevant features that are not redundant. In this subsection, the definition of redundancy based on Markov blanket [21, 22] is described.

Definition 3. Markov Blanket: Let M be a subset of features which does not contain X_i, i.e., $M \subseteq X$ and $X_i \notin M$. M is a Markov blanket of X_i if X_i is conditionally independent of $(X \cup C) - M - \{X_i\}$ given M, i.e., $P(X - M - \{X_i\}, C|X_i, M) = P(X - M - \{X_i\}, C|M)$.

Definition 4. Redundancy: Let G be the current set of selected features, a feature F_i is redundant and hence should be removed from G if and only if it is weakly relevant and has a Markov blanket M in G.

Currently, feature selection research has focused on dealing with redundant features and searching for the optimal feature subset [2]. Koller and Sahami [22] proposed a correlation based feature selection method and Markov blanket for identifying and eliminating redundant features. However, the computational complexity for finding the conditional independence of features in Markov blanker is typically very high, which limits the efficiency of the method. To mitigate this deficiency of Markov blanket, Yu and Liu [20] proposed the efficient fast correlation-based filter method (FCBF) that approximates the Markov blanket based on single feature.

Definition 5. Approximate Markov Blanket: For two features X_i and $X_j (i \neq j)$, X_j is said to be an approximate Markov blanket of X_i if $SU_{j,C} \geq SU_{i,C}$ and $SU_{i,j} \geq SU_{i,C}$ where the symmetrical uncertainty SU [23] measures the correlation between features (including the class, C).The symmetrical uncertainty is defined as:

$$SU(X_i, X_j) = 2\left[\frac{IG(X_i|X_j)}{H(X_i)+H(X_j)}\right] \tag{1}$$

where $IG(X_i|X_j)$ is the information gain between features X_i and X_j, $H(X_i)$ and $H(X_j)$ denote the entropies of X_i and X_j respectively. $SU_{i,C}$ denotes the correlation between feature X_i and the class C, and is named *C-correlation*.

2.3 Full and Partial Class Relevance

Besides identifying features of relevance to C in multiclass problems, it would be more effective to be able to pinpoint the specific subset of classes a feature is relevant to. Hence, the original concepts of relevant feature is extended in [15] so as to differentiate partial class relevant (PCR) from full class relevant (FCR) features in multiclass problems. In what follows, the notions of PCR and FCR based on the concept of Subset-Versus-Subset are presented:

Definition 6. Subset-Versus-Subset: A Subset-Versus-Subset (SVS) of classes C is defined as SVS(C)$= \{A, B\}$ where $A, B \subseteq C$ and $A \bigcap B = \varnothing$.

Definition 7. Full Class Relevance: A feature X_i is said to be full class relevant if and only if it is relevant to all possible SVSs.

Definition 8. Partial Class Relevance: A feature X_i is said to be partial class relevant if and only if it is relevant to only some of the SVSs and there exists a SVS C', such that X_i is irrelevant to C'.

FCR features are relevant to all SVSs, including all pairs of classes $\{\{c_i\},\{c_j\}\}$ $(i, j \in (1,\ldots,k))$, they are important for distinguishing between any two classes. On the other hand, PCR features are relevant to only some of the SVSs, they are helpful for distinguishing only subset of classes but not all classes. An ideal solution for feature selection on multiclass problem would be a FCR feature subset that can distinguish between any pair of classes. In practice, it is generally hard to find such an ideal FCR feature subset that would perform well on all classes. A selected feature subset performing well on some SVSs would not be successful on other SVSs. Hence PCR features are indispensable for learning the multiclass problem.

3 Single Objective Memetic Algorithm for Feature Selection

In this section, the details of the single objective memetic algorithm for feature selection is presented. The algorithm is a hybridization of filter method based local search and GA wrapper method. The pseudo code of the algorithm is outlined in Figure 2.

Filter Local Search Embedded Genetic Algorithm
BEGIN

1. **Initialize:** Randomly generate an initial population of feature subsets encoded with binary strings.
2. **While** (*not converged or computational budget is not exhausted*)
3. Evaluate fitness of all feature subsets in the population based on $J(s)$.
4. Select the elite chromosome s_e to undergo filter method based local search.
5. Replace s_e with an improved new chromosome s'_e using Lamarckian learning.
6. Perform evolutionary operators based on restrictive selection, crossover, and mutation.
7. **End While**

END

Fig. 2. Filter local search embedded genetic algorithm for feature selection

At the start of the search, an initial GA population is initialized randomly with each chromosome encoding a candidate feature subset. In the present work, each chromosome is composed of a bit string of length equal to the total number of features in the feature selection problem of interest. Using binary encoding, a bit of '1' ('0') implies the corresponding feature is selected (excluded). The fitness of each chromosome is then obtained using an objective function based on the induction algorithm:

$$Fitness(s) = J(s) \tag{2}$$

where s denotes the selected feature subset encoded in a chromosome, and the feature selection objective function $J(s)$ evaluates the significance for the given feature subset. Here, $J(s)$ is the generalization error obtained for s which can be estimated using cross validation or bootstrapping techniques. Note that when two chromosomes are found having similar fitness (i.e., for a misclassification error of less than one data instance, the difference between their fitness is less than a small value of $\varepsilon = 1/n$, where n is the number of instances), the one with a smaller number of selected features is given higher chance of surviving to the next generation.

In each GA generation, the elite chromosome, i.e., the one with the best fitness value then undergoes filter method based memetic operators/local search in the spirit of Lamarckian learning [24, 25]. The Lamarckian learning forces the genotype to reflect the result of improvement through placing the locally improved individual back into the population to compete for reproductive opportunities. Two memetic operators, namely an *Add* operator that inserts a feature into the elite chromosome, and a *Del* operator that removes some of the existing features from the elite chromosome, are introduced in the following subsection. The important question is which feature(s) to add or delete. Ideally, each added feature should be strong relevant and each deleted feature should be irrelevant or redundant.

3.1 Filter Method Based Local Search

For a given candidate solution s encoded in a chromosome, $\mathscr{X}$ and $\mathscr{Y}$ define the sets of selected and excluded features encoded in s, respectively. The purpose of the *Add* operator is to select a highly correlated feature $\mathscr{Y}_i$ from $\mathscr{Y}$ to $\mathscr{X}$ based on a filter method. The *Del* operator on the other hand selects least correlated features $\mathscr{X}_i$ from $\mathscr{X}$ and remove them to $\mathscr{Y}$. The work operations of *Add* and *Del* operators are depicted in Fig. 3 and 4, respectively.

***Add* Operator:**
BEGIN

1. Rank the correlation of features in $\mathscr{Y}$ in a descending order based on a filter method.
2. Select a feature $\mathscr{Y}_i$ in $\mathscr{Y}$ using linear ranking selection [26] such that the larger the correlation of a feature in $\mathscr{Y}$ is the more likely it is to be selected.
3. Add $\mathscr{Y}_i$ to $\mathscr{X}$.

END

Fig. 3. *Add* operation

***Del* Operator:**
BEGIN

1. Rank the correlation of features in $\mathscr{X}$ in a descending order based on a filter method.
2. Select a feature $\mathscr{X}_i$ in $\mathscr{X}$ using linear ranking selection [26] such that the lower the correlation of a feature in $\mathscr{X}$, the more likely it will be selected.
3. Remove $\mathscr{X}_i$ to $\mathscr{Y}$.

END

Fig. 4. *Del* operation

The correlation measure of each feature in both memetic operators need only be calculated once in the search. This feature ranking information is then archived for use in any subsequent *Add* and *Del* operations, for fine-tuning the GA solutions throughout the search. The details of the *Add* and *Del* operations are further illustrated in Figure 5. For instance, $F5$ and $F4$ represent the highest and lowest ranked features in $\mathscr{Y}$, while $F3$ and $F6$ are the highest and lowest ranked features in $\mathscr{X}$, respectively. In the *Add* operation, $F5$ is thus the most likely feature to be moved to $\mathscr{X}$, whereas $F6$ is the most likely feature to be moved to Y.

It is worth noting that the proposed memetic framework is designed to accommodate a diverse of filter-based methods. Among them, the univariate ranking (*Gain Ratio* [27], *ReliefF* [28], and *Chi Square* [29]) and Markov blanket based filter methods have been investigated and systematically studied in [13] and [14], respectively. With both types of filter methods based local search, the memetic frameworks have been shown

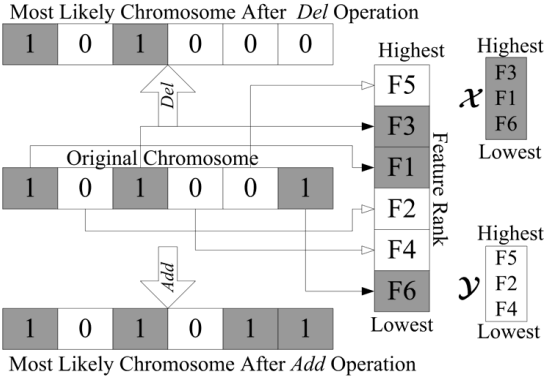


Fig. 5. Add and Del operations ($\mathscr{X}$ and $\mathscr{Y}$ denote the selected and excluded feature subsets, respectively)

to identify the relevant features more efficiently than many existing approaches in the literature for mining dataset containing very large number of features. At the same time, they converges to improved or competitive classification accuracy with smaller number of identified features. Further, it was shown in [14] that the Markov blanket based filter method is capable of dealing with dataset plagued with many redundant features. The *Del* operation in Markov blanket based local search is designed to identify and eliminate redundant features, as it removes features that from $\mathscr{X}$ that are covered by highly correlated features through approximate Markov blanket. The details of the *Del* operator based on Markov blanket is depicted in Figure 6.

***Del* Operator Based on Markov Blanket:**
BEGIN

1. Rank the correlation features in $\mathscr{X}$ in a descending order based on *C-correlation* measure.
2. Select a feature $\mathscr{X}_i$ in $\mathscr{X}$ using linear ranking selection [26] such that the larger the *C-correlation* of a feature in $\mathscr{X}$ is the more likely it is to be selected.
3. Eliminate all features in $\mathscr{X} - \{\mathscr{X}_i\}$ which are in the approximate Markov blanket [20] of $\mathscr{X}_i$. If no feature is eliminated, try removing $\mathscr{X}_i$ itself.

END

Fig. 6. *Del* operation based on Markov blanket

It is possible to quantify the computational complexity of the two memetic operators based on the search range l, which defines the maximum numbers of *Add* and *Del* operations. Therefore, with l possible *Add* operations and l possible *Del* operations, there are a total of l^2 possible combinations of *Add* and *Del* operations applied on a chromosome. The l^2 combinations of *Add* and *Del* are applied to the candidate chromosome in a random order and the procedure stops once an improvement is obtained either in

Filter Method Based Local Search Using Improvement First Strategy
BEGIN

1. Select the elite chromosome s to undergo memetic operations..
2. **For** $j = 1$ **to** l^2
3. Generate a unique random pair $\{\mathfrak{a}, \mathfrak{d}\}$ where $0 \leq \mathfrak{a}, \mathfrak{d} < l$.
4. Apply $\mathfrak{a}$ times *Add* and $\mathfrak{d}$ times *Del* on s to generate a new chromosome s'.
5. Calculate fitness of modified chromosome $F(s')$.
6. **IF** $f(s') > f(s)$
7. Break local search and return s'.
8. **End IF**
9. **End For**
10. **End For**

END

Fig. 7. Filter method based local search

terms of fitness or reduction in the number of selected features without deterioration in the fitness value. The procedure of the filter method based memetic operation applied on the elite chromosome of each GA search generation is outlined in Figure 7.

After applying the above Lamarckian learning process on the elite chromosome, the GA population then undergoes the usual evolutionary operations including linear ranking selection[26], uniform crossover, and mutation operators with elitism[10].

4 Multiobjective Memetic Algorithm for Simultaneous FCR/PCR Feature Identification

Since single objective feature selection methods do not distinguish PCR or FCR features in multiclass problems, multiobjective memetic framework for simultaneously identification of both FCR and PCR features have been introduced in [15].

4.1 Identification of FCR/PCR Features, Problem Formulation

The search for true FCR and PCR features may pose to be computationally intractable due to the large number of possible SVSs. One key issue is to choose an approximate scheme that can generate a coverage of the classes C. The One-Versus-All (OVA) scheme is generally regarded as an efficient and effective strategy in the literature of high-dimensional multiclass classification [30]. Hence, it is used for generating the SVSs in the present study. For classes C, OVA scheme creates k pairwise two-class SVSs (labeled as **OVA sets** in the rest of this chapter), with each of them constructed as $\{c_i, \overline{c}_i\}$, where $i \in (1, \ldots, k)$ and $\overline{c}_i = C - c_i$.

The search for optimal PCR feature subsets of k OVA sets can naturally be casted as a multiobjective optimization problem (MOP)[16, 31, 32, 33] with each objective

corresponding to the feature selection accuracy of each OVA set. The MOP considered is thus defined as:

$$\begin{aligned} min\, F(s) &= (f_1(s), \ldots, f_k(s))\ subject\ to\ s \in S \\ f_i(s) &= -Acc(s, \{c_i, \overline{c}_i\}), (i \in (1, \ldots, k)) \end{aligned} \tag{3}$$

where $F(s)$ is the objective vector, s is the candidate selected feature subset, k is the number of classes, and S is the feasible domain of s. The i-th objective $f_i(s)$ is then $-Acc(s, \{c_i, \overline{c}_i\})$, which gives the classification accuracy of the i-th OVA set for the selected feature subset s. Finding a single solution that optimizes all objectives is not always possible because a selected feature subset performing well on an OVA set would not be successful on other OVA sets.

The straightforward way to solve this MOP is to identify the optimal PCR feature subset of each OVA set individually. The intersection of PCR feature subsets then form the FCR subset. However, such a process would be computationally intensive. Hence, a MOEA based method MBE-MOMA for simultaneous identification of both FCR and PCR features in a single run is proposed by Zhu et al. in [15].

Pareto-based multiobjective evolutionary algorithm (MOEA) [16] is one of the most popular approaches for handling MOPs, due to its ability to find multiple diverse solutions and approximating the Pareto optimal set in a single simulation run, which provides distinct options to solve the problem based on different tradeoffs of the objectives.

Solving the MOP defined in Equation (3) with the MOEA would lead to an approximate Pareto optimal set of solutions $(s_1, s_2, \ldots, s_p)$ with different tradeoffs of classification accuracy on the OVA sets. One core advantage of the MOP approach to classification accuracy on the OVA sets is that the user is able to choose a solution according to the problem at hand. Subsequently, based on the preferences of the user or decision maker on the objectives, different solutions in the approximate Pareto optimal set could be selected in the final predictions. For instances, if a class c_i is considered as more crucial than others, it would be possible to select a solution performing superiorly on f_i, such that c_i is better distinguished from other classes. In the experimental study, each class is considered with equal importance and the use of both PCR and FCR feature subsets in the final prediction is demonstrated (the details are provided later in Section 4.3). Particularly, the optimal PCR feature subset for the i-th OVA set is presented as the extremal solution on f_i, i.e., $PCR_i = \arg\min_{s_j} f_i(s_j)$, and the FCR feature subset is defined as intersection of all solutions i.e., $FCR = \{s_1 \cap s_2 \cap, \ldots, \cap s_p\}$.

4.2 Filter Local Search Embedded Multiobjective Memetic Algorithm

This subsection presents the filter local search embedded multiobjective memetic algorithm, which is a synergy of MOEA [16] and filter method based local search [14] for simultaneous identification of FCR and PCR features by solving the MOP defined in Equation (3). The pseudo code of the algorithm is outlined in Fig. 8.

At the start of the search, an initial population of solutions is randomly generated with each chromosome encoding a candidate feature subset. In the present work, each chromosome is composed of a bit string of length equal to the total number of features in the feature selection problem of interest. Using binary encoding, a bit of '1' ('0') implies

Filter Local Search Embedded Multiobjective Memetic Algorithm
BEGIN

1. t = 0.
2. **Initialize:** Randomly generate an initial population $P(t)$ of feature subsets encoded with binary strings.
3. Evaluate fitness $F(s)$ of each solution in $P(t)$.
4. Rank $P(t)$ using Pareto dominance and calculate the crowding distance [31].
5. **While**(*Termination Criterion Not Fulfilled*)
6. Select a temporary population $P'(t)$ from $P(t)$ based on Pareto ranking and crowding distance.
7. Perform crossover and mutation on $P'(t)$.
8. Evaluate fitness $F(s)$ of each solution in $P'(t)$.
9. Rank $\{P'(t) \bigcup P(t)\}$ using Pareto dominance.
10. Apply **filter method based local search** on the non-dominated solutions of $\{P'(t) \bigcup P(t)\}$ and generate an improved population $P''(t)$.
11. Rank the population $\{P(t) \bigcup P'(t) \bigcup P''(t)\}$ using Pareto dominance calculate the crowding distance.
12. Select solutions from $\{P(t) \bigcup P'(t) \bigcup P''(t)\}$ to create a new population $P(t+1)$ based on Pareto ranking and crowding distance.
13. t = t+1.
14. **End While**

END

Fig. 8. Outline of filter local search embedded multiobjective memetic algorithm for feature selection

the corresponding feature is selected (excluded). The fitness of each chromosome is then obtained using an objective vector defined in Equation (3).

In each generation, an offspring population $P'(t)$ is created from the parent population $P(t)$ using the genetic operators, i.e., selection, crossover, and mutation. $P(t)$ and $P'(t)$ are merged together as a mating pool $\{P'(t) \bigcup P(t)\}$. Subsequently, a non-dominated sorting [16] is used to categorize the solutions of the mating pool into levels of Pareto fronts. The non-dominated solutions of $\{P'(t) \bigcup P(t)\}$ then undergo the filter based local search as greater detailed in Subsection 3.1. The locally improved solution s' of a selected solution s on a randomly selected objective $f_i(s)$ through individual learning is then archived in a temporary population $P''(t)$. Subsequently, a new population $P(t+1)$ of the same size as $P(t)$ is generated from $\{P(t) \bigcup P'(t) \bigcup P''(t)\}$. Elitism and diversity in $P(t+1)$ is maintained based on Pareto dominance and crowding distance [31]. The evolutionary operators applied in this multiobjective MA include binary tournament selection [31], uniform crossover, and mutation operators [10].

4.3 Synergy between FCR and PCR Feature Subsets

The output of the multiobjective memetic algorithm is a set of non-dominated solutions. For k OVA sets, there exist k extremal solutions which each represents the optimal

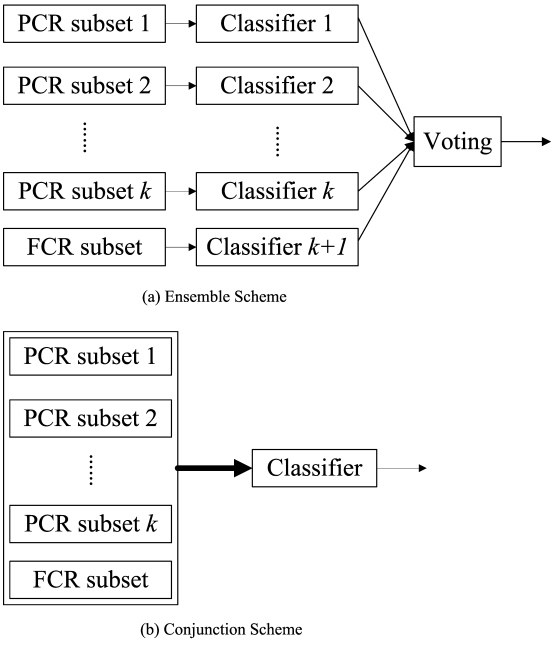


Fig. 9. Synergy schemes of FCR and PCR features

solution of the respective objective. In other words, each extremal solution encodes an optimal PCR feature subset for the corresponding OVA set. On the other hand one FCR feature subset is formed by intersection of all non-dominated solutions. The next key issue to consider is how to combine these $k+1$ feature subset systematically for the final prediction.

Two schemes (as shown in Fig. 9) for synergizing the $k+1$ feature subsets are considered in [15]. The first ensemble scheme is widely used in feature selection and classification [34, 35]. In this scheme, each of the $k+1$ feature subsets is employed for classifying all classes, and the predictions of all trained classifiers are then aggregated based on voting. The second alternative considered is the conjunction scheme which is a union of all $k+1$ feature subsets as one feature subset. The newly formed union feature subset is then used in the classification accuracy prediction. It is worth noting that the number of selected features in multiobjective MA is determined based on the number of unique features among all $k+1$ feature subsets.

5 Empirical Study

The performance of the single/multi-objective memetic frameworks against the state-of-the-art filter and wrapper feature selection algorithms using high-dimensional benchmark datasets is investigated in this section. In particular, the Markov blanket based filter method is used for local search due to its advantage on handling redundant

features. The corresponding single/multi-objective memetic algorithms with Markov blanket based local search are denoted as Markov Blanket Embedded GA (MBEGA) [14] and Markov blanket embedded multiobjective memetic algorithm (MBE-MOMA) [15], respectively.

The standard GA and MOEA are also considered as benchmark algorithms for single/multi-objective problems, respectively. All evolutionary algorithms, i.e, GA, MBEGA, MOEA, and MBE-MOMA, use the same parameter setting of population size $= 50$, crossover probability $= 0.6$, and mutation rate $= 0.1$. Note that like most existing work in the literature, if prior knowledge on the optimum number of features is available, it make sense to constrain the number of bits '1' in each chromosome to a maximum of m in the evolutionary search process. In this case, specialized restrictive crossover and mutation [13] instead of the basic evolutionary operators are necessary, so that the number of bits '1' in each chromosome does not violate the constraint derived from the prior knowledge on m throughout the search. In this study, the maximum number of selected features in each chromosome m is set to 50. The search stops when convergence to the global optimal has occurred or the maximum computational budget allowable (i.e., 2000 fitness functional calls) is reached. It is worth noting that the fitness function calls made in the memetic operations are also included as part of the total fitness function calls. The memetic operation range l in MBEGA and MBE-MOMA is set to 4.

To evaluate the performances of the feature selection algorithms considered in this study, the average of 30 independent runs of external .632 bootstrap [36] are reported. A comparison of various error estimation methods [36] has suggested that .632 bootstrap is generally more appropriate than other estimators including re-substitution estimator, k-fold cross-validation, and leave-one-out estimation on datasets with small sample size. For datasets with very few instances in some classes, a stratified bootstrap sampling is used instead so that the distribution of each class in the sampled dataset is maintained consistent to the original dataset.

5.1 Single Objective Study

Firstly, the performance of the single objective method MBEGA against recent filter and wrapper feature selection algorithms is studied on single objective feature selection problem. In particular, the FCBF [20], BIRS [37] and standard GA feature selection algorithms are considered. These algorithms have previously been successfully used for feature selection and demonstrated to attain promising performance [20, 38, 5, 37].

The study is performed on 4 binary-class benchmark datasets including an often used dataset Corral [39] and its three extended versions with additional redundant and irrelevant features. The purpose is to evaluate the four algorithms on different combinations of redundant and irrelevant features. The algorithms are expected to perform well on some but not all combinations.

The first dataset is the Corral data [39], which consist of 6 boolean features ($A0, A1, B0, B1, I, R75$) and a boolean class C defined by $C = (A0 \wedge A1) \vee (B0 \wedge B1)$. The features $A0, A1, B0$ and $B1$ are independent to each other, feature I is uniformly random and irrelevant to class C. $R75$ matching 75% of C is redundant. The second dataset, Corral-46 generated in [20], is obtained by introducing more irrelevant and

Table 1. Summary of the four synthetic datasets

Dataset	Features	
Corral	**OS:**	$A0, A1, B0, B1$
	RF:	$R75$
	IF:	I
Corral-46	**OS:**	$A0, A1, B0, B1$
	RF:	$A0_{\{1,...,10/16\}}, A1_{\{1,...,10/16\}}, B0_{\{1,...,10/16\}}, B1_{\{1,...,10/16\}}$
	IF:	$I0, I1, \ldots, I13$
Corral-50	**OS:**	$A0, A1, B0, B1$
	RF:	$A0_{\{1,...,10/16\}}, A1_{\{1,...,10/16\}}, B0_{\{1,...,10/16\}}, B1_{\{1,...,10/16\}}$ $R90, R85, R80, R75$
	IF:	$I0, I1, \ldots, I13$
Corral-10000	**OS:**	$A0, A1, B0, B1$
	RF:	$A0_{\{1,...,10/16\}}, A1_{\{1,...,10/16\}}, B0_{\{1,...,10/16\}}, B1_{\{1,...,10/16\}}$ $R90, R85, R80, R75$
	IF:	$I0, I1, \ldots, I9963$

OS: optimal subset; **RF:** redundant features; **IF:** irrelevant features.
$A0_{\{1,...,10/16\}}$ denotes the subset of 7 features matching $A0$ at levels of 1, 15/16, 14/16,...,10/16. (Similar definition applied to $A1_{\{1,...,10/16\}}$, $B0_{\{1,...,10/16\}}$, $B1_{\{1,...,10/16\}}$).

redundant features to the original Corral data. It includes the optimal feature subset $(A0, A1, B0, B1)$, 14 irrelevant features, and 28 additional redundant features. Among the 28 additional redundant features, $A0, A1, B0$ and $B1$, each has 7 redundant features that match at levels of $1, 15/16, 14/16, \ldots, 10/16$. The third dataset, Corral-50, is created by adding to Corral-46 with redundant features $R75, R80, R85$ and $R90$, that match C at the level of $75\%, 80\%, 85\%$ and 90%, respectively. In the following text, the subset of $R75, R80, R85$, and $R90$ is denoted as $R+$ for the sake of brevity. All features $R+$ are all highly correlated to C and have larger *C-correlation* measure than features in the optimal feature subset. To test the methods on problems of high-dimensions and high-redundancies, the last dataset Corral-10000 is generated from Corral-50 with additional 9964 irrelevant features.

The four synthetic datasets considered here are also summarized in Table 1. In the present study, C4.5 is considered as the classifier for feature subset evaluation since it provides the optimal subset of $(A0, A1, B0, B1)$ at a theoretical prediction accuracy of 100%.

FCBF, BIRS, and the classifiers C4.5 used in the following studies have been developed using the Weka environment [40]. The parameters for FCBF, C4.5 were based on the defaults in Weka. For BIRS, the configurations reported in [37] are used.

The feature selection performances of FCBF, BIRS, GA and MBEGA on the synthetic datasets using ten 10-fold cross-validations with C4.5 classifier are reported in Table 2. In particular, the selected feature subset, number of selected features, and average classification accuracy are tabulated. Due to the stochastic nature of MBEGA and GA, the average feature selection results of MBEGA and GA for ten independent runs are reported. The maximum number of selected features in each chromosome, m, is set to 50.

The results in Table 2 show that FCBF, BIRS, and MBEGA locate the optimal feature subset at a classification accuracy of 100% on the Corral-46 dataset. GA fails to find the optimal subset on the Corral-46 dataset, but it performs better than FCBF

Table 2. Feature selection by each algorithm on synthetic data

		FCBF	BIRS	MBEGA	GA
Corral	S_c	(R75,A0,A1,B0,B1)	(R75)	**(A0,A1,B0,B1)**	**(A0,A1,B0,B1)**
	(#)	5	1	4 ± 0	4 ± 0
	acc	96.02	75.00	100.00 ± 0	100.00 ± 0
Corral-46	S_c	**(A0,A1,B0,B1)**	**(A0,A1,B0,B1)**	**(A0,A1,B0,B1)**	(…)
	(#)	4	4	4 ± 0	12.4 ± 0.4
	acc	100.00	100.00	100.00 ± 0	100.00 ± 0
Corral-50	S_c	(R+,A0,A1,B0,B1)	(R90)	**(A0,A1,B0,B1)**	(…)
	(#)	8	1	4 ± 0	16.3 ± 1.2
	acc	97.97	90.63	100.00 ± 0	100.00 ± 0
Corral-10000	S_c	(R+,A0,A1,B0,B1)	(R90)	**(A0,A1,B0,B1)**	(…)
	(#)	8	1	4 ± 0	13.6 ± 2.5
	acc	97.97	90.63	100.00 ± 0	82.51 ± 4.72

S_c: selected feature subset; (#): average number of selected features; *acc*: classification accuracy.

and BIRS on Corral dataset. Only MBEGA successfully identifies the optimal feature subset at perfect accuracy of 100% on all the four synthetic datasets. FCBF manages to identify $A0, A1, B0, B1$ but fails to eliminate the features in $R+$. Since all features in $R+$ have higher *C-correlation* values than any feature in the optimal feature subset. FCBF fails to identify these redundant features based on the approximate Markov blanket. On the other hand, BIRS is a sequential forward search method that is incapable of considering interactions between features, hence it is more likely to find suboptimal results. BIRS selects the feature with the highest *C-correlation* value. For instance, on dataset Corral-50, BIRS selects only the single top ranked feature, $R90$, with a classification accuracy of 90.63% and any additional features do not bring any improvement. GA selects much more features than the other methods. Without Markov blanket based memetic operators, GA alone is inefficient in eliminating redundant features under the limited computational budget.

5.2 Multiobjective Study

The performance of the proposed method MBE-MOMA is investigated and compared with its single objective counterpart, i.e., the MBEGA and the non-local-search counterpart, i.e., the standard MOEA based on NSGAII [31].

Six synthetic multiclass datasets are used for studying the weaknesses and strengths of the feature selection algorithms as well as to illustrate the notion of FCR and PCR features. Three 3-class (C3_1,C3_2,C3_3) and three 4-class (C4_1,C4_2,C4_3) synthetic datasets are generated based on the approach described in [41], with each class containing 25 samples.

Each synthetic dataset consists of both relevant and irrelevant features. The relevant features in each dataset are generated from a multivariate normal distribution using the mean and covariance matrixes tabulated in Table 3. And 4000 irrelevant features are added to each dataset. Among these 4000 features, 2000 are drawn from a normal distribution of N(0,1) and the other 2000 features are sampled with a uniform distribution of U[-1,1].

Table 3. Mean and covariance matrixes used for generating synthetic datasets

Dataset	Mean	Covariance
C3_1	$\mu = \begin{bmatrix} 0 & 0 \\ 3.7 & 1 \\ 1 & 3.7 \end{bmatrix} \otimes \mathbf{A_1}$	$\sigma = I(2) \otimes \mathbf{e}$
C3_2	$\mu = \begin{bmatrix} 0 & 1.18 & -1.18 & -1.18 \\ 3.7 & -1.18 & 1.18 & -1.18 \\ 1 & -1.18 & -1.18 & 1.18 \end{bmatrix} \otimes \mathbf{A_1}$	$\sigma = I(4) \otimes \mathbf{e}$
C3_3	$\mu = \begin{bmatrix} 1 & -1 & -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \otimes \mathbf{A_{1.18}}$	$\sigma = I(6) \otimes \mathbf{e}$
C4_1	$\mu = \begin{bmatrix} 1 & 1 \\ 4.95 & 1 \\ 1 & 4.95 \\ 4.95 & 4.95 \end{bmatrix} \otimes \mathbf{A_1}$	$\sigma = I(2) \otimes \mathbf{e}$
C4_2	$\mu = \begin{bmatrix} 1 & 1.18 & -1.18 & 1.18 & 1.18 \\ 4.95 & -1.18 & 1.18 & 1.18 & 1.18 \\ 1 & -1.18 & -1.18 & -1.18 & 1.18 \\ 4.95 & -1.18 & -1.18 & 1.18 & -1.18 \end{bmatrix} \otimes \mathbf{A_1}$	$\sigma = I(5) \otimes \mathbf{e}$
C4_3	$\mu = \begin{bmatrix} 1 & -1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 & -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 & 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix} \otimes \mathbf{A_{1.18}}$	$\sigma = I(8) \otimes \mathbf{e}$

$\mathbf{A_x}$ is a 25×10 matrix with each element taking value x; $\otimes$ denotes the Kronecker product [42]; $I(n)$ is a $n \times n$ identify matrix; $\mathbf{e}$ represents a 10×10 symmetrical matrix with each diagonal element having value of 1 while all other elements having value of 0.9.

C3_1 and C4_1 are designed to contain only 20 FCR and 4000 irrelevant features. The centroids of the three classes in C3_1 are located at (0,0), (3.7,1), and (1,3.7). While the centroids of the four classes in C4_1 are located at (1,1), (4.95,1), (1,4.95), and (4.95, 4.95). Both C3_1 and C4_1 have two groups of relevant features generated from a multivariate normal distribution, with 10 features in each group. The variance of each relevant feature is 1. The correlation between intra-group features is 0.9, whereas the correlation between inter-group features is 0. Features in these two groups are FCR features, since they are able to distinguish any SVS sets of classes; and furthermore, features in the same group are redundant with each other and the optimal feature subset for distinguishing the three classes consists of any 2 relevant feature from different groups.

C3_2 and C4_2 are designed to contain FCR, PCR, and irrelevant features. They are composed of 4040 (40 relevant and 4000 irrelevant) and 4050 (50 relevant and 4000 irrelevant) features, respectively. The centroids of the three classes in C3_2 are located at (0,1.18), (3.7,1.18), and (1,1.18)[2]. While the centroids of the four classes in C4_2 are

[2] The two coordinates(features) of the centroids for each class may not be the same. For example, the centroid of class 1 is specified on G0 and G1, while the centroid of class 2 is specified on G0 and G2.

Table 4. Feature selected by each algorithm on synthetic data

	Features	MBEGA	MOEA	MBE-MOMA
C3_1	*(#)*	8.2	57.5	25.4
	OPT(2)	**2**	**2**	**2**
	Red	3.3	3.2	7.0
	Irr	2.9	52.3	16.4
C3_2	*(#)*	6.6	61.3	19.3
	OPT(4)	3.3	3.8	**4**
	Red	2.4	3.1	7.2
	Irr	0.9	54.4	8.1
C3_3	*(#)*	11.9	58.1	21.6
	OPT(6)	5.3	5.6	**5.9**
	Red	5.2	3.6	10.1
	Irr	1.4	48.9	4.6
C4_1	*(#)*	12.0	92.2	27.8
	OPT(2)	**2**	**2**	**2**
	Red	4.3	4.8	8.4
	Irr	5.7	85.4	17.4
C4_2	*(#)*	6.9	98.4	32.0
	OPT(5)	4.2	4.9	**5**
	Red	1.2	5.3	11.4
	Irr	1.5	88.2	15.6
C4_3	*(#)*	10.1	103.6	34.5
	OPT(8)	5.9	7.9	**8**
	Red	3.0	6.8	18.3
	Irr	1.2	88.9	8.2

(#): Number of Selected features; *OPT(x)*: Number of selected features with in the optimal subset, *x* indicates the optimal number of features; *Red*: Redundant Features; *Irr*: Irrelevant Features.

located at (1,1.18), (4.95,1.18), (1,-1.18), and (4.95,-1.18). In C3_2, 4 groups of relevant features (G0, G1, G2, and G3) are generated from a multivariate normal distribution, with 10 features in each group. Features in the same group are redundant to each other. In this dataset only features in G0 are FCR features. While those features in G1, G2, and G3 are PCR features. Further, features in G1 are drawn under different distributions for class 1 as compared to the other two classes, i.e., N(1.18,1) for class 1 and N(-1.18,1) for the remaining two classes, hence it is possible to distinguish class 1 from any other classes. However, since features in G1 are drawn under the same distribution of N(-1.18,1) for samples of classes 2 and 3, it is difficult to distinguish between these two classes. As such, there are 5 groups of relevant features in C4_2, with only G0 consisting of FCR features and the other 4 groups containing PCR features. The optimal feature subset in C3_2 and C4_2 consists of one FCR feature from G0 and another PCR feature from the corresponding group. For instance, to correctly classify class 1 in C3_2, two features each from G0 and G1 are required. The optimal feature subset to distinguish

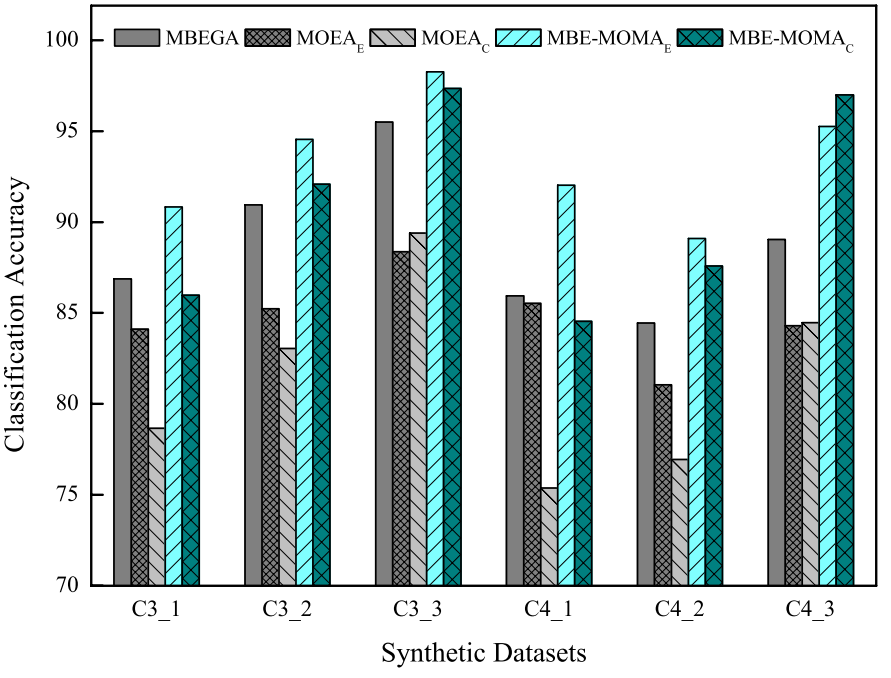


Fig. 10. Classification accuracy by each algorithm on synthetic data. The subscript $_E$ and $_C$ denote the algorithms using ensemble and conjunction scheme, respectively.

all the three classes thus consists of 4 features, one FCR feature from G0 and three PCR features each from G1, G2, and G3.

Last but not least, C3_3 and C4_3 have been designed to contain only PCR and irrelevant features. They are composed of 4060 (60 relevant and 4000 irrelevant) and 4080 (80 relevant and 4000 irrelevant) features, respectively. The centroids of the three(four) classes for C3_3(C4_3) are all located at (1.18,-1.18) but on different features. Six(eight) groups of relevant features are generated from a multivariate normal distribution in C3_3(C4_3), with 10 features in each group. Features in the same group are designed to be redundant to each other. All the relevant features are PCR features and the optimal feature subset for distinguishing one class from the others consists of two PCR features each from the corresponding group. For instance, two features each from the first and second group form the optimal feature subset for separating class 1 and the other classes. The optimal feature subset to distinguish all the three(four) classes thus consists of 6(8) features with one from each group.

The MBEGA, MOEA, and MBE-MOMA algorithms are used to search on each of the six synthetic datasets and the list of corresponding selected features and classification accuracies obtained by all three algorithms are presented in Tables 4 and Fig 10, respectively. The results in Table 4 suggest that MBE-MOMA selects more features among the optimal subset than the other algorithms. On C3_1 and C4_1, all three algorithms have successfully identified the 2 FCR features among the optimal subset.

On the other four datasets (C3_2, C3_3, C4_2, and C4_3) which contain both FCR and PCR features, only MBE-MOMA manages to identify majority of the FCR and PCR features belonging to the optimal set, while at the same time producing classification accuracy superior to both MBEGA and MOEA (see Fig 10). Further, the ensemble MBE-MOMA exhibits the best classification accuracies on the first 5 datasets, and the conjunction MBE-MOMA obtains best classification accuracy on the last one, C4_3. In comparison, MBEGA fails to locate many of the important features belonging to the optimal subset. Note that since $k+1$ feature subsets are used in MBE-MOMA, it is expected to select more features than MBEGA, where only a single optimal feature subset is considered. For the same computational budget, MOEA selects a larger number of features while arriving at lower classification accuracy on the datasets, since it lacks the ability to remove the irrelevant features.

6 Conclusions

The study of memetic frameworks for mining high dimensional dataset is a research area that has attracted increasing attention in recent years. This chapter describes some state-of-the-art memetic frameworks for single/multi-objective feature selection. The single objective memetic framework is designed for identifying the crucial feature subset that is capable of generating accurate predictions. The multiobjective memetic framework, on the other hand, is designed for simultaneous identification of FCR and PCR features on multiclass problems. Comparison study of the memetic frameworks to recently proposed feature selection schemes suggests that memetic search lead to competitive or superior search performances. Both single/multi-objective memetic feature selection algorithms have been further demonstrated to eliminate irrelevant and redundant features efficiently when incorporating filter method based local search into basic genetic wrapper model. The multiobjective memetic algorithm is also illustrated to be capable of identifying FCR and PCR features efficiently and at the same time generating more accurate predictions on multiclass problems than both the single objective memetic search counterpart, as well as the standard MOEA which does not use local search. To summary, it is worth noting that memetic feature selection frameworks serve as modern tool for crucial features discovery to assist researchers in analyzing the growing amounts of data in various research fields. Hence it is hope that the work presented here on memetic frameworks would help promote greater research in the identification of new research directions of feature selection.

References

1. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. IEEE Transaction on Knowledge and Data Engineering 17(4), 491–501 (2005)
2. Yu, L.: Redundancy-based feature selection for high-dimensional data and application in bioinformatics, Phd Thesis, Arizona State University (2005)
3. Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., Bloomfield, C.D., Lander, E.S.: Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286(5439), 531–537 (1999)

4. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine Learning 46(1-3), 389–422 (2002)
5. Li, L., Weinberg, C.R., Darden, T.A., Pedersen, L.G.: Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the ga/knn method. Bioinformatics 17(12), 1131–1142 (2001)
6. Deutsch, J.M.: Evolutionary algorithms for finding optimal gene sets in microarray prediction. Bioinformatics 19(1), 45–52 (2003)
7. Hart, G.L.W., Blum, V., Walorski, M.J., Zunger, A.: Evolutionary approach for determining first-principles hamiltonians. Nature Materials 4, 391–394 (2005)
8. Blum, V., Hart, G.L.W., Walorski, M.J., Zunger, A.: Using genetic algorithm to map first-principle results to model hamiltonians: Application to the generalized ising model for alloys. Physical Review B 72(165113) (2005)
9. Kohavi, R., John, G.H.: Wrapper for feature subset selection. Artificial Intelligence 97(1-2), 273–324 (1997)
10. Holland, J.H.: Adaptation in natural artificial systems, 2nd edn. MIT Press, Cambridge (1992)
11. Yang, J.H., Honavar, V.: Feature selection using a genetic algorithm. IEEE Transaction on Intelligent Systems 13(2), 44–49 (1998)
12. Raymer, M.L., Punch, W.F., Goodman, E.D., Kuhn, L.A., Jain, A.K.: Dimensionality reduction using genetic algorithms. IEEE Transaction on Evolutionary Computation 4(2), 164–171 (2000)
13. Zhu, Z., Ong, Y.S., Dash, M.: Wrapper-filter feature selection algorithm using a memetic framework. IEEE Transactions On Systems, Man and Cybernetics - Part B 37(1), 70–76 (2007)
14. Zhu, Z., Ong, Y.S., Dash, M.: Markov blanket-embedded genetic algorithm for gene selection. Pattern Recognition 49(11), 3236–3248 (2007)
15. Zhu, Z., Ong, Y.S., Zurada, M.: Simultaneous identification of full class relevant and partial class relevant genes. IEEE/ACM Transactions on Computational Biology and Bioinformatics (2008) (Under Review)
16. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
17. Dash, M., Liu, H.: Feature selection for classification. Intelligent Data Analysis: An International Journal 1(3), 131–156 (1997)
18. Dash, M., Liu, H.: Consistency-based search in feature selection. Artificial Intelligence 151(1-2), 155–176 (2003)
19. Liu, H., Motoda, H.: Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic, Boston (1998)
20. Yu, L., Liu, H.: Efficient feature selection via analysis of relevance and redundancy. Journal of Machine Learning Research 5, 1205–1224 (2004)
21. Pearl, J.: Probabilitic Reasoning in Intelligent Systems. Morgan Kaufmann, San Mateo (1988)
22. Koller, D., Sahami, M.: Toward optimal feature selection. In: Proceedings of the 13th International Conference on Machine Learning, pp. 284–292 (1996)
23. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C. Cambridge University Press, Cambridge (1998)
24. Krasnogor, N.: Studies on the Theory and Design Space of Memetic Algorithms, Ph.D. Thesis, Faculty of Computing, Mathematics and Engineering, University of the West of England, Bristol, U.K (2002)
25. Ong, Y.S., Keane, A.J.: Meta-lamarckian in memetic algorithm. IEEE Transaction on Evolutionary Computation 8(2), 99–110 (2004)

26. Baker, J.E.: Adaptive selection methods for genetic algorithms. In: Proceedings of the 1st International Conference on Genetic Algorithms, pp. 101–111 (1985)
27. QuinLan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo (1993)
28. Robnic-Sikonja, M., Kononenko, I.: Theoretical and empirical analysis of relieff and rrelieff. Machine Learning 53(1-2), 23–69 (2003)
29. Liu, H., Setiono, R.: Chi2: Feature selection and discretization of numeric attributes. In: Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence (1995)
30. Statnikov, A., Aliferis, C.F., Tsamardinos, I., Hardin, D., Levy, S.: A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. Bioinformatics 21(5), 631–643 (2005)
31. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Transaction on Evolutionary Computation 6(2), 182–197 (2002)
32. Ishibuchi, H., Murata, T.: A multi-objective genetic local search algorithm and its application to flowshop scheduling. IEEE Trans. on Systems, Man, and Cybernetics -Part C: Applications and Reviews 28(3), 392–403 (1998)
33. Yoshida, T., Ishibuchi, H., Murata, T.: Balance between genetic search and local search in memetic algorithm for multiobjective permutation flowshop scheduling. IEEE Transaction on Evolutionary Computation 7(2), 204–223 (2003)
34. Oliveira, L.S., Morita, M., Sabourin, R.: Feature Selection for Ensembles Using the Multi-Objective Optimization Approach. In: Jin, Y. (ed.) Multi-Objective Machine Learning, ch. 3. Springer, Heidelberg (2006)
35. Tsymbal, A., Puuronen, S., Patterson, D.W.: Ensemble feature selection with the simple bayesian classification. Information Fusion 4(2), 87–100 (2003)
36. Braga-Neto, U.M., Dougherty, E.R.: Is cross-validation valid for small-sample microarray classification? Bioinformatics 20(3), 374–380 (2004)
37. Ruiz, R., Riquelme, J.C., Aguilar-Ruiz, J.S.: Incremental wrapper-based gene selection from microarray data for cancer classification. Pattern Recognition 39(12), 2383–2392 (2006)
38. Li, L., Pedersen, L.G., Darden, T.A., Weinberg, C.R.: Computational analysis of leukemia microarray expression data using ga/knn method. In: Proceeding of the 1st Conference on Critical Assessment of Microarray Data Analysis, CAMDA 2000 (2000)
39. John, G.H., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: Proceeding of the 11th International Conference on Machine Learning, pp. 121–129 (1994)
40. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
41. Diaz-Uriarte, R., de Andres, S.A.: Gene selection and classification of microarray data using random forest. BMC Bioinformatics 7(3) (2006)
42. Graham, A.: Kronecker Products and Matrix Calculus With Applications. Halsted Press/John Wiley and Sons, NY (1981)

Multi-Objective Robust Optimization Assisted by Response Surface Approximation and Visual Data-Mining

Koji Shimoyama[1], Jin Ne Lim[1], Shinkyu Jeong[1], Shigeru Obayashi[1], and Masataka Koishi[2]

[1] Institute of Fluid Science, Tohoku University, 2-1-1 Katahira, Aoba-ku, Sendai, 980-8577, Japan
{shimoyama,lim,jeong,obayashi}@edge.ifs.tohoku.ac.jp

[2] Tire Research and Technology Development Dept., Yokohama Rubber Co., Ltd., 2-1 Oiwake, Hiratsuka, Kanagawa, 254-8610, Japan
koishi@hpt.yrc.co.jp

A new approach for multi-objective robust design optimization was proposed and applied to a real-world design problem with a large number of objective functions. The present approach is assisted by response surface approximation and visual data-mining, and resulted in two major gains regarding computational time and data interpretation. The Kriging model for response surface approximation can markedly reduce the computational time for predictions of robustness. In addition, the use of self-organizing maps as a data- mining technique allows visualization of complicated design information between optimality and robustness in a comprehensible two- dimensional form. Therefore, the extraction and interpretation of trade-off relations between optimality and robustness of design, and also the location of sweet spots in the design space, can be performed in a comprehensive manner.

1 Introduction

Design optimization can be described as the problem of determining the inputs of an objective function that will maximize or minimize its value at a certain design condition. Although this conventional (so-called *deterministic*) approach that considers only the *optimality* of design, i.e., performance at design condition, should work fine in a controlled environment, real-world applications inevitably involve errors and uncertainties (be it in the design process, manufacturing process, and/or operating conditions); so that the resulting performance may be lower than expected. More recently, *robust optimization* that considers not only optimality but also *robustness*, i.e., performance sensitivity against errors and uncertainties, has attracted considerable attention in the search for more practical designs.

A comparison between conventional optimization and robust optimization is illustrated in Fig. 1. The solution A obtained from a conventional optimization is the best in terms of optimality, but disperses widely in terms of objective function against the dispersion of design variable or environmental variable, and this dispersion may extend to

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 133–151.
springerlink.com

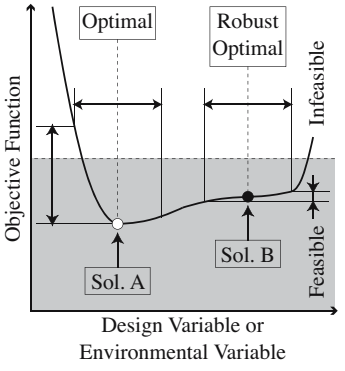


Fig. 1. Comparison between conventional optimization and robust optimization (in minimization problem)

the infeasible range. On the other hand, the solution B obtained from a robust optimization is moderately good in terms of optimality and also good in terms of robustness, i.e., dispersion of objective function is narrow against dispersion of design variable.

Improvements in optimality and robustness are usually competing in real-world design problems. Therefore, not a single, but multiple robust optimal solutions actually exist, so that finding these compromised solutions and revealing the trade-off relation between optimality and robustness are both objectives for robust design optimization. The newly acquired knowledge can aid the upper-level decision maker to pick one solution from the compromised solutions, together with an additional design consideration.

Up to the present, several robust optimization approaches have been proposed by many researchers. In the approaches proposed by [1] and [2], acceptable ranges are pre-specified for each objective and/or constraint function dispersions. Both approaches consist of inner and outer optimization problems. In the inner sub-problem, the maximum size (radius) of the dispersive region in the design variable space, so that the objective and/or constraint function dispersions are safely included within the pre-specified ranges, is evaluated as a robustness measure. In the outer main problem, Gunawan and Azarm's approach considers the radius as an additional constraint, while Li's approach considers the radius as an additional objective function. Clearly, both approaches result in searching only robust optimal solutions whose objective and/or constraint function dispersions fit their acceptable ranges. It means that these approaches require additional optimization runs with pre-specification of different acceptable ranges, in order to search other robust optimal solutions with more or less robust characteristics, i.e., to extract information on the trade-off between optimality and robustness. In addition, the resulting robust optimal solutions strongly depend on the pre-specified ranges. In the case that the pre-specified ranges are not appropriate (*e.g.* they are too severe) or some information about the ranges is unavailable, these approaches may fail in searching for robust optimal solutions.

On the other hand, the approach proposed by [3] pre-specifies ranges for all design variables' dispersions, and sets an upper limit for the objective function dispersion as a robustness measure in the form of a constraint. This approach can search not only

robust optimal solutions whose objective function dispersion fits the upper limit, but also those whose objective function dispersion becomes smaller (i.e., more robust) than the upper limit. It means that this approach can search for a wider range of robust optimal solutions when compared to the above approaches [1, 2]. However, this approach still has difficulty in pre-specifying the upper limit appropriately.

The approach proposed by [4] considers the worst value of a dispersive objective function value (for pre-specified design variable dispersions) as a single objective function. By optimizing the worst objective function value, this approach works well in terms of finding an extremely robust optimal solution, but it becomes difficult to search for global trade-off between optimality and robustness because this approach is based on a single-objective formulation.

Aiming at a practical method to obtain the global trade-off relation between optimality and robustness of design, the author proposed a robust optimization approach using multi-objective evolutionary algorithms (MOEAs) [5] in his past studies [6, 7]. This approach deals with two statistical values (mean value and standard deviation) of a dispersive objective function, and treats them as multiple separate objective functions corresponding to optimality and robustness measures, respectively. In addition, there exist other approaches based on a similar multi-objective-like formulation [8, 9]. In the author's studies [6, 7], the utilization of MOEAs was motivated by the fact that it allowed the consideration of an optimization problem from a multi-objective perspective; i.e., an m-objective conventional optimization problem can be converted to a $2m$-objective robust optimization problem consisting of m optimality measures and m robustness measures. Indeed, the approach using MOEA featured a superior capability to globally search trade-off relations among competing objective functions in multi-objective optimization problems, thus the trade-off between optimality and robustness can also be revealed in robust optimization problems.

However, robust optimization requires further considerations when applied to real-world design problems. One major issue is that robust optimization is considerably more time-consuming than conventional optimization. This is mainly because robust optimization requires evaluation of objective functions at many sample points (usually more than 10^3) [10] around each searching point, in order to derive statistical values such as mean value and standard deviation, which are then used as optimality and robustness measures for each objective function. Therefore, it is required to reduce function evaluation time for a more efficient robust optimization, especially in real-world design problems which utilize expensive computations for function evaluations.

Another important issue for robust optimization is the difficulty in interpreting complicated output data in order to obtain general design information. As mentioned above, robust optimization deals with twice as many objective functions as conventional optimization, and the resulting high-dimensional output data (large number of objective functions) is rather complicated to understand and to discuss. The situation goes increasingly severe according to the number of objective functions involved, and thus automated data analysis techniques are required to handle the large amounts of high-dimensional output data and to reduce human load in real-world design problems which consider a large number of objective functions.

This study has two main objectives: (1) Propose an efficient approach for multi-objective robust design optimization assisted by response surface method and a data mining visualization technique, and (2) Demonstrate the capabilities of the present approach in the robust optimization problem of an automobile tire. Essentially, this problem has a large number of objective functions and requires substantial computational time to evaluate their values using a commercial software; therefore this is an appropriate application to validate the present robust optimization approach.

2 Proposed Multi-Objective Robust Design Optimization Process

Figure 2 illustrates the flowchart of the present multi-objective robust design optimization process. This process consists of mainly three blocks: response surface (RS) approximation block, optimization block and visual data-mining (DM) block, as shown in the broken-line rectangles. Single-solid-line and double-solid-line rectangles indicate processes based on objective function values estimated by RS methods (RSMs), and real values obtained directly from expensive computations, respectively.

Details of each block are described in the following Sects. 2.1–2.3, while the synergetic effects of the blocks are summarized in Sect. 2.4.

2.1 Response Surface Approximation Block

RSMs [11] approximate response data (*e.g.* for objective and/or constraint functions) using simple algebraic functions, thus allowing for a considerable reduction in function evaluation time. These algebraic functions are derived from *real* function values given at several points distributed in the whole design space, so as to fit the given response data. The RSs are then constructed from the derived algebraic functions and promptly give *estimated* function values at other points where response data is unknown.

The most widely used RS is the polynomial-based model [11, 12] because of its simplicity and ease of use. However, this model is not suitable for representing multi-modality and non-linearity of objective functions in real-world design problems.

In this study, the Kriging model [13, 14] is adopted as the present RSM. The Kriging model is a stochastic RSM, which can adapt well to nonlinear functions. In addition, the Kriging model gives not only estimated function values but also approximation errors, which help to determine locations in the design space where additional points of response data should be considered for improvements in RS accuracy.

Consider the approximation of a function $f(x)$ in terms of n design variables $x = [x_1, x_2, \cdots, x_n]^T$, given M points of response data $f(x^1), f(x^2), \cdots, f(x^M)$. The Kriging model approximates $f(x)$ as

$$f(x) = \mu + \varepsilon(x) \tag{1}$$

where μ is the constant global model corresponding to the mean value over all given response data $f(x^1), f(x^2), \cdots, f(x^M)$. $\varepsilon(x)$ is the local model corresponding to the deviation from μ at x, defined as the Gaussian random variable $\mathrm{N}(0, \sigma^2)$. The correlation between the deviations at any two points x^i and x^j is related to the distance between the two corresponding points $d(x^i, x^j)$ as

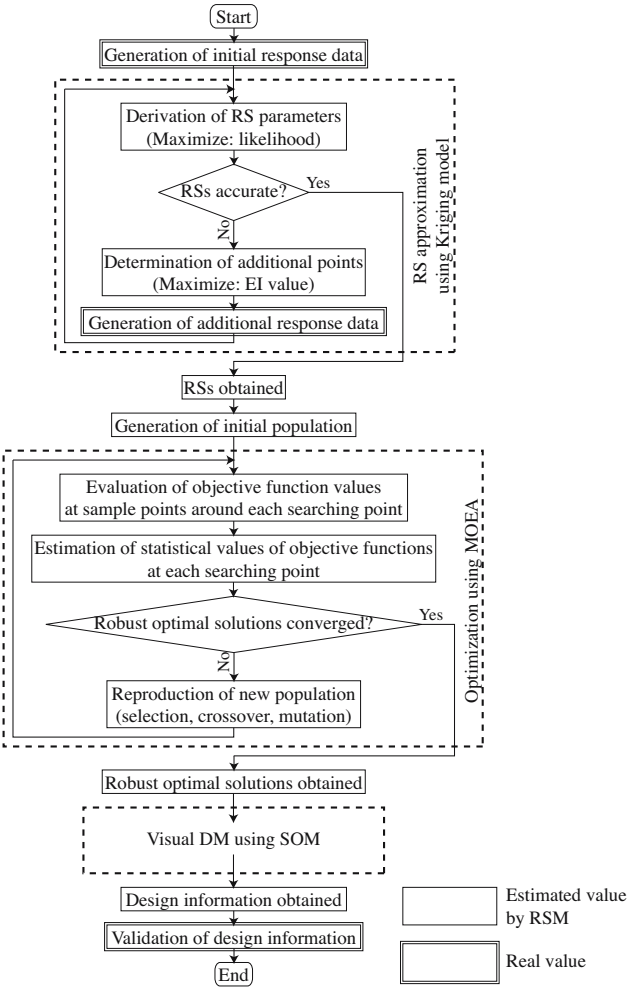


Fig. 2. Flowchart of multi-objective robust design optimization process

$$\mathrm{Corr}\left[\varepsilon(x^i),\varepsilon(x^j)\right] = \exp\left[-d(x^i,x^j)\right] \tag{2}$$

$$d(x^i,x^j) = \sum_{k=1}^{n} \theta_k \left|x_k^i - x_k^j\right|^2 \tag{3}$$

where $\theta = [\theta_1, \theta_2, \cdots, \theta_n]^T$ is the weighting factor for each design variable.

The Kriging model determines the parameters μ, σ^2 and θ so as to maximize the likelihood function $Ln(\mu, \sigma^2, \theta)$. μ and σ^2 that maximizes $Ln(\mu, \sigma^2, \theta)$ are given in closed form as

$$\mu = \frac{1^T R^{-1} f}{1^T R^{-1} 1} \tag{4}$$

$$\sigma^2 = \frac{(f - 1\mu)^T R^{-1} (f - 1\mu)}{M} \tag{5}$$

where R is an $M \times M$ matrix whose (i, j) entry is Corr $\left[\varepsilon(x^i), \varepsilon(x^j)\right]$, $f = [f(x^1), f(x^2), \cdots, f(x^M)]^T$, and 1 is an M-dimensional unit vector. Then, θ is searched for by maximizing the following $Ln(\mu, \sigma^2, \theta)$ as

$$Ln(\mu, \sigma^2, \theta) = -\frac{M}{2} \ln\left(\sigma^2\right) - \frac{1}{2} \ln\left(|R|\right) \tag{6}$$

Thus, Eq. 1 is rewritten to the final form of Kriging model predictor as

$$f(x) = \mu + r^T R^{-1} (f - 1\mu) \tag{7}$$

where r is an M-dimensional vector whose i-th element is Corr $\left[\varepsilon(x), \varepsilon(x^i)\right]$. Detailed derivation of Eq. 7 is found in [13].

The accuracy of the estimated value on the Kriging model largely depends on the distance from the given response data points. Intuitively speaking, the closer point x is to the response data point $x^1, x^2, \cdots, x^M$, the more accurate the predictor $f(x)$ (Eq. 7) is. This is mathematically conveyed by the expression of the mean squared error $s^2(x)$ of $f(x)$ as

$$s^2(x) = \sigma^2 \left[1 - r^T R^{-1} r + \frac{\left(1 - 1^T R^{-1} r\right)^2}{1^T R^{-1} 1}\right] \tag{8}$$

A full derivation of Eq. 8 is also found in [13]. In general, if the accuracy of the current RS is insufficient, it is required to reconstruct a new RS by adding more response data points. Accuracy improvements in the present Kriging model is accomplished by iteratively adding points with a maximum value of *expected improvement (EI)*, which corresponds to the probability that the function approximation $f(x)$ (Eq. 7) may achieve a new global optimal on a reconstructed RS with the consideration of an additional point x. In an $f(x)$ minimization problem, the improvement value $I(x)$ and the EI value $E[I(x)]$ of $f(x)$ are expressed respectively as

$$I(x) = \max\left[f_{\min} - F, 0\right] \tag{9}$$

$$E[I(x)] = \int_{-\infty}^{f_{\min}} (f_{\min} - F)\, \phi(F) \mathrm{d}F \tag{10}$$

where $f_{\min}$ is the minimum function value in the current response data $f(x^1)$, $f(x^2), \cdots, f(x^M)$, F is the Gaussian random variable $\mathrm{N}(f, s^2)$, and $\phi(F)$ is the probability density function of F.

2.2 Optimization Block

Using the constructed RSs, a robust optimization is performed in the next block. In general, a conventional optimization problem (minimizing $f(x)$) is given as

$$\text{Minimize: } f(x) \tag{11}$$

In robust optimization, on the other hand, this problem is rewritten to the problem where the mean value μ_f and the standard deviation σ_f of $f(x)$ must be minimized when x disperses around a searching point due to errors and uncertainties as

$$\begin{cases} \text{Minimize:} & \mu_f \\ \text{Minimize:} & \sigma_f \end{cases} \tag{12}$$

Therefore, an m-objective $(f_1(x), f_2(x), \cdots, f_m(x))$ conventional optimization problem is converted to a $2m$-objective robust optimization problem consisting of m optimality measures $(\mu_{f_1}, \mu_{f_2}, \cdots, \mu_{f_m})$ and m robustness measures $(\sigma_{f_1}, \sigma_{f_2}, \cdots, \sigma_{f_m})$. In this study, the above multi-objective robust optimization problem is solved by using a multi-objective evolutionary algorithm [5]. MOEAs have the capability of revealing multiple compromised solutions and also the trade-off relations among competing objective functions for a given multi-objective design optimization problem. In the case of robust design optimization, the competing objective functions are optimality and robustness, so the trade-off between them can also be revealed [6, 7].

For the present MOEA, the initial population (N solutions $x^1, x^2, \cdots, x^N$) is generated randomly by considering the whole design space. For each solution x^i $(i = 1, 2, \cdots, N)$, numerous sample points are generated randomly around the searching point x^i by changing the dispersive (random) design variables so as to follow certain probability distributions, and $f(x)$ is evaluated at each sample point. Then, statistical values μ_f and σ_f are estimated from the sampled $f(x)$ for each solution. Comparing the estimated μ_f and σ_f among all solutions $x^1, x^2, \cdots, x^N$, fitness values are assigned for all the solutions. Next, from the N solutions in the current generation, better N^* solutions are selected as parents and new N^* solutions are reproduced as children through crossover and mutation. Finally, N better solutions among the current N solutions and the new N^* solutions are picked as the population for the next generation (alternation of generations). This process is iterated until the non-dominated solutions between μ_f and σ_f have converged and multiple robust optimal solutions revealing the trade-off relation between μ_f and σ_f have been obtained (specific MOEA operators are given in Sect. 3.2).

2.3 Visual Data-Mining Block

In optimizations with two or three objective functions, trade-off relations can be visualized simply by plotting the resultant non-dominated solutions, however, conventional visualization becomes virtually impossible when more than three objective functions are considered. DM visualization technique is thus applied in this block in order to discover indistinct patterns such as regularities and correlations existing in high-dimensional data. Therefore, it can help us to automatically extract and easily understand general design information among many competing objective functions.

In this study, a self-organizing map (SOM) [15] is adopted as the present DM visualization technique. The SOM is an unsupervised machine learning, nonlinear projection algorithm from high to low-dimensional space. This projection is based on self-organization of a low-dimensional array of neurons.

Consider the projection of N points of m-dimensional input data (m objective function values) $f^1, f^2, \cdots, f^N$ $(f^i = \left[f_1^i, f_2^i, \cdots, f_m^i\right]^T)$ onto L neurons in a low-dimensional

space. The j-th neuron is associated with the weight vector $w^j = \left[w_1^j, w_2^j, \cdots, w_m^j\right]^T$. Each neuron is connected to adjacent neurons by a neighborhood relation, and usually forms a two-dimensional rectangular or hexagonal topology in the SOM. The learning algorithm of SOM starts with the search for the best-matching unit w^{c_i}, which is the closest neuron to the i-th input vector f^i as

$$||f^i - w^{c_i}|| = \min \left|\left|f^i - w^j\right|\right|, \quad j = 1, 2, \cdots, L \tag{13}$$

Once the best-matching unit has been determined, the weight adjustments are performed not only for the best-matching unit but also for its neighbors. The adjustment for the j-th neuron w^j depends on the distance (similarity) from the input vector f^i and the neuron w^j. Based on the distance, the best-matching unit and its neighbors become closer to the input vector f^i. Repeating this learning algorithm, the weight vectors $w^1, w^2, \cdots, w^L$ become smooth not only locally but also globally. Applying this learning process to all input data $f^1, f^2, \cdots, f^N$, the sequence of close vectors in the original space results in a sequence of neighboring neurons in the two-dimensional map. Thus, the SOM reduces the dimension of the objective function data while preserving their own features, and the resultant two-dimensional map can be used for visualization and data interpretation.

The above approach describes a sequential SOM, which applies the learning processes to all input vectors one by one. However, the learning results may change according to the order of the applied processes. In this study, therefore, a batch SOM is adopted in order to guarantee the uniqueness of the learning results. The batch SOM adjusts weight vectors $w^1, w^2, \cdots, w^L$ after determining all the best-matching units $w^{c_1}, w^{c_2}, \cdots, w^{c_N}$ for all input vectors $f^1, f^2, \cdots, f^N$, while the sequential SOM does it after determining the best-matching unit w^{c_i} for an input vector f^i. In the batch SOM, the j-th weight vector w^j is adjusted to w_{adj}^j as

$$w_{adj}^j = \sum_{i=1}^{N} h_{jc_i} f^i \Bigg/ \sum_{i=1}^{N} h_{jc_i} \tag{14}$$

where h_{jk} is defined by the following Gaussian function as

$$h_{jk} = \exp\left(-\frac{{d_{jk}}^2}{{r_t}^2}\right) \tag{15}$$

where d_{jk} denotes the Euclidean distance between two neurons w^j and w^k, and r_t denotes the neighborhood radius, which decreases with the iteration of learning processes.

The general design information that can be visualized through the obtained SOM is based on the function values estimated by the utilized RSM. Naturally, the information must be validated by evaluating real function values with actual computations as the final step.

2.4 Synergetic Effects

The RS block provides the following optimization block with the approximation of objective functions in computationally inexpensive forms. It contributes to an efficient

function evaluation and optimal solution search in the optimization block which uses EA. Note that this contribution becomes more important as the number of function evaluations becomes larger, so the use of RS approximation is indispensable for the robust optimization, since it uses statistical values of the original objective functions through many sample points.

The optimization block provides many optimal (non-dominated) solution vectors; each of which has many objective function values. The following DM block visualizes these solution vectors in a lower-dimensional comprehensive form, and it contributes to an efficient extraction of important design information from the huge and complex optimization output. The robust optimization deals with more objective functions and provides higher-dimensional optimal solution vectors than the conventional optimization, thus the DM visualization is also indispensable for the robust optimization.

3 Application Problem

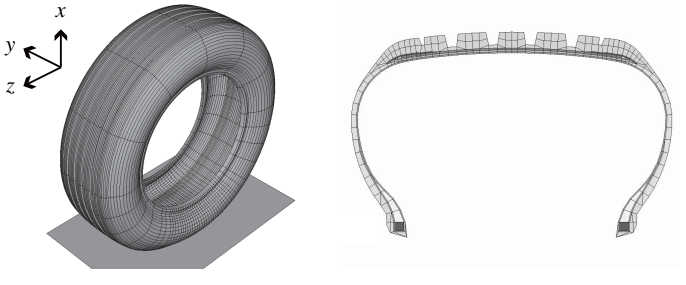
The proposed multi-objective robust optimization approach assisted by the Kriging-based RSM and the SOM-based DM technique is applied to an automobile tire design problem and its capabilities are demonstrated in this section.

3.1 Problem Definition

Figure 3 shows the automobile tire structure considered in this study. The present automobile tire design problem considers the following vertical, horizontal and circumferential tire stiffnesses (E_x, E_y and E_z) as three objective functions:

$$\begin{cases} \text{Minimize:} & E_x \\ \text{Maximize:} & E_y \\ \text{Maximize:} & E_z \end{cases} \tag{16}$$

A smaller E_x corresponds to better riding comfortability, and larger E_y and larger E_z correspond to better cornering and braking performances, respectively. This problem has a



(a) Overall view (b) Cross-sectional view

Fig. 3. Automobile tire structure

total of ten design variables: angle of belt layer for reinforcing tire ground-contact surface (one variable), rubber elasticities in cap, side and flange regions (three variables), and cross-sectional tire configuration parameters (six variables). The cross-sectional configuration r is defined simply in the form of weighted summation of baseline configuration r_0 and six basis vectors $\Delta r_1, \Delta r_2, \cdots, \Delta r_6$ as

$$r = r_0 + \sum_{i=1}^{6} \alpha_i \Delta r_i \tag{17}$$

where α_i is a coefficient for Δr_i, and set as the present design variable.

In this study, the above optimization problem (Eq. 16) is converted into the following robust optimization problem considering the optimality and the robustness of each tire stiffnesses as five objective functions, when tire internal pressure p_{int} disperses around the nominal condition as a random variable following the normal distribution with its standard deviation set at 25% of the mean (nominal) value:

$$\begin{cases} \text{Minimize:} & \mu_{E_x} \\ \text{Maximize:} & \mu_{E_y} \\ \text{Minimize:} & \sigma_{E_y}/\mu_{E_y} \\ \text{Maximize:} & \mu_{E_z} \\ \text{Minimize:} & \sigma_{E_z}/\mu_{E_z} \end{cases} \tag{18}$$

where $\mu_{E_{x,y,z}}$ and $\sigma_{E_{x,y,z}}$ are the mean value and the standard deviation of $E_{x,y,z}$ against the dispersion of p_{int} . The present robustness measure is expressed in a non-dimensional form ($\sigma_{E_{x,y,z}}/\mu_{E_{x,y,z}}$) in order to eliminate the scale effect where a smaller $\mu_{E_{x,y,z}}$ leads to a smaller $\sigma_{E_{x,y,z}}$; otherwise the comparison between $\mu_{E_{x,y,z}}$ and $\sigma_{E_{x,y,z}}$ would be unfair. In addition, the present robust optimization does not consider the robustness of E_x because it is not an important factor from the engineering point of view.

In this study, both the conventional optimization (Eq. 16) and the robust optimization (Eq. 18) are performed, and results are compared to demonstrate the capabilities of the proposed multi-objective robust optimization approach.

3.2 Numerical Methods and Conditions

The present conventional and robust optimization processes have already been shown in the flowchart of Fig. 2. Here, details of the processes are described as follows.

In the RS approximation block using Kriging model, 99 points with ten different design variables and one random variable are generated uniformly in the whole design space by the Latin hypercube sampling (LHS) [16]; and initial response data is generated by evaluating real values of E_x, E_y and E_z through actual computational structure analyses using the commercial software ABAQUS at these 99 points. Based on the response data, three RSs are constructed for E_x, E_y and E_z as functions of ten design variables and one random variable. If the RSs' accuracies are insufficient, the RSs are reconstructed again by adding more response data at a maximum EI value point.

In the optimization block using MOEA, currently a multi-objective genetic algorithm (MOGA), populations of 512 solutions evolve through 100 generations. Based on the RSs of $E_{x,y,z}$ constructed by the Kriging model, the objective functions are then evaluated. In the conventional optimization, $E_{x,y,z}$ are evaluated directly from the RSs at each searching point. In the robust optimization, on the other hand, $\mu_{E_{x,y,z}}$ and $\sigma_{E_{x,y,z}}$ are estimated by the Monte Carlo simulation (MCS) with descriptive sampling (DS) [17]. The present MCS random variable is p_{int}, and the sample size is set as 10^3 for each searching point. From the evaluated objective functions, fitness values are evaluated by using a Pareto-ranking method [18] and a fitness sharing [5, 18]. Parents are selected by the stochastic universal sampling (SUS) [19], and children are reproduced by the simulated binary crossover (SBX) [5] and polynomial mutation [5] at a 10% rate. The alternation of generations is performed by the Best-N selection [20, 21].

Finally, in the DM visualization block using SOM, the resultant data from the non-dominated solutions for the three-objective functions (conventional optimization) and five objective functions (robust optimization), is projected and clustered onto a two-dimensional hexagonal topology by using the commercial SOM software Viscovery SOMine [22].

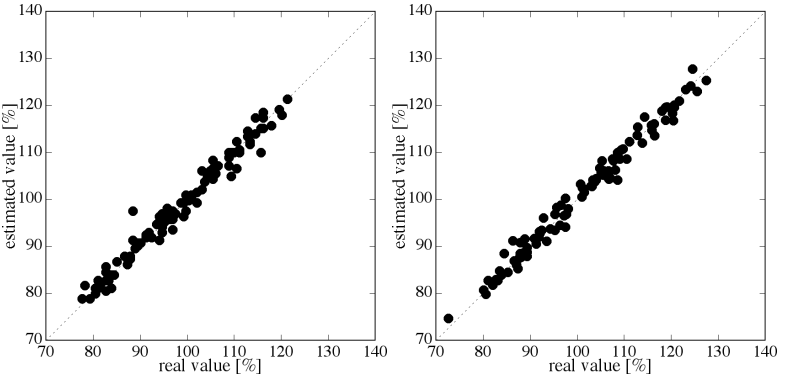
As mentioned above, real performance evaluations (ABAQUS simulations to evaluate tire stiffnesses E_x, E_y and E_z) are performed in the subblocks "generation of initial response data," "generation of additional response data," and "validation of design information" (shown in the double-solid-line rectangles in Fig. 2). Here, note that the present robust optimization approach can be easily extended to other design problems, simply by changing the evaluation methods which are used there. For example, in the case of an aircraft design optimization, a computational fluid dynamics solver can be utilized instead of the ABAQUS in the tire design, while the optimization process itself remains just the same.

4 Application Results

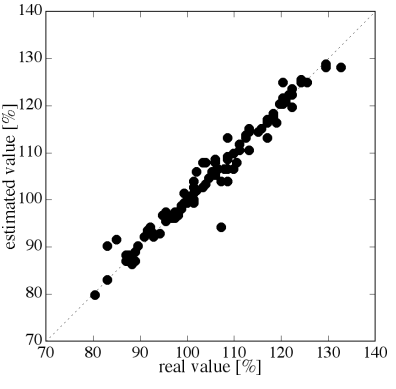
4.1 Validation of Response Surface Accuracy

The accuracy of the present Kriging-based RSs constructed from the initial dataset of 99 response points is demonstrated here by the cross-validation of values of the real function and values estimated by the RS using a different dataset. The cross-validation is to compare a real function value at the points where response data are given, with a function value estimated by the RS constructed from the response data lacking the data at the given point. Those results are depicted in Fig. 4 for three tire stiffnesses E_x, E_y and E_z (shown in percentages of the baseline values).

For all tire stiffnesses E_x, E_y and E_z, the values estimated by the current RSs have small discrepancies which are lower than 5% of the real values at most of the given points. These results indicate that the current Kriging-based RSs have enough accuracy to predict qualitative characteristics of E_x, E_y and E_z, which is important when estimating robustness measures. In this study, therefore, the current RSs need not be reconstructed again by adding more response data based on EI values, and the optimization will be performed using the current RSs constructed from the initial 99 points.



(a) E_x (b) E_y

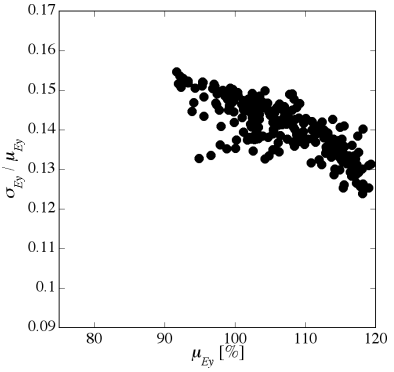


(c) E_z

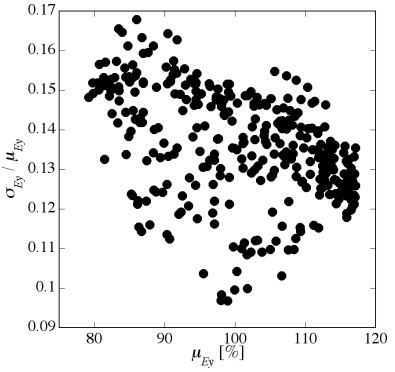
Fig. 4. Cross-validation of Kriging-based RSs for tire stiffnesses

4.2 Validation of the Capability to Search for Robust Optimal Solutions

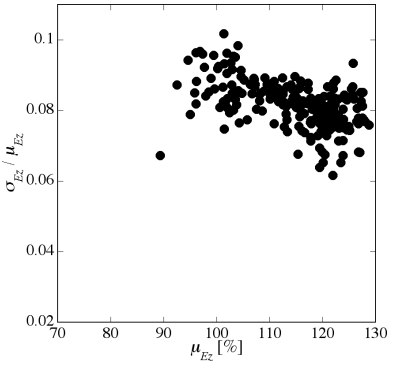
The capability to search for robust optimal solutions is demonstrated here by comparing the results of the present conventional optimization and robust optimization. The resultant non-dominated solutions obtained through conventional and robust optimizations are shown in Figs. 5 and 6, respectively (values are shown in percentages of the baseline values estimated by the RSs). In the conventional optimization case, the resultant non-dominated solutions that are optimal in terms of the three objective functions (Eq. 16) are plotted on the μ_{E_y}–(σ_{E_y}/μ_{E_y}) plane (Fig. 5(a)), and on the μ_{E_z}–(σ_{E_z}/μ_{E_z}) plane (Fig. 5(b)), after estimating the statistical values $\mu_{E_{x,y,z}}$ and $\sigma_{E_{x,y,z}}$ of these solutions by the same MCS used in the robust optimization. In the robust optimization case, on the other hand, the resultant non-dominated solutions that are optimal in terms of the five



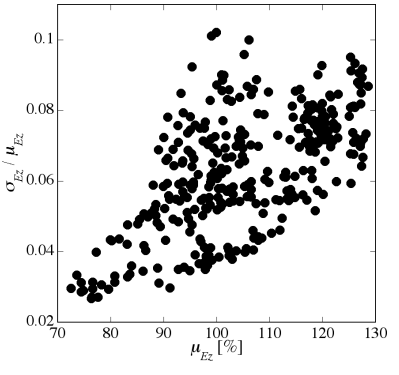
(a) μ_{E_y}–(σ_{E_y}/μ_{E_y})



(a) μ_{E_y}–(σ_{E_y}/μ_{E_y})



(b) μ_{E_z}–(σ_{E_z}/μ_{E_z})



(b) μ_{E_z}–(σ_{E_z}/μ_{E_z})

Fig. 5. Non-dominated solutions obtained through conventional optimization (shown in values estimated by RSs)

Fig. 6. Non-dominated solutions obtained through robust optimization (shown in values estimated by RSs)

objective functions (Eq. 18) are plotted directly on the μ_{E_y}–(σ_{E_y}/μ_{E_y}) plane (Fig. 6(a)), and on the μ_{E_z}–(σ_{E_z}/μ_{E_z}) plane (Fig. 6(b)).

Comparing Figs. 5(a) and 6(a), it can be seen that the non-dominated solutions obtained through the robust optimization have smaller σ_{E_y}/μ_{E_y} values than those obtained through the conventional optimization. Similar tendency can be seen in σ_{E_z}/μ_{E_z} by comparing Figs. 5(b) and 6(b). These results indicate that the present robust optimization successfully found the solutions with robust E_y and E_z characteristics, while the conventional optimization did not. Therefore, the proposed multi-objective robust optimization approach has a superior capability to search for robust optimal solutions.

4.3 Validation of the Capability to Reveal Design Information

The capability to reveal design information is demonstrated here. First, objective function data of the non-dominated solutions is clustered by the present SOM so that it can be visualized in a two-dimensional form. Figures 7 and 8 show the SOMs applied to the three-dimensional and the five-dimensional objective function data obtained through the conventional and the robust optimizations, respectively. Here note that a given solution keeps the same position in all SOMs, and each SOM is colored by an objective function value (shown in percentages of the baseline values estimated by the RSs).

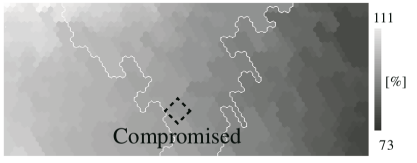
The conventional optimization (Fig. 7) does provide information on the trade-off relations between the minimization of E_x and the maximization of E_y and E_z, and also reveal that the tendencies for maximization of E_y and E_z are slightly different from each other. However, the existence of these trade-off relations can also be confirmed by comparing the SOMs of μ_{E_x}, μ_{E_y} and μ_{E_z} in the robust optimization case (Fig. 8). In addition, the robust optimization shows further trade-off relations between optimality and robustness measures: μ_{E_y} maximization and σ_{E_y}/μ_{E_y} minimization, μ_{E_z} maximization and σ_{E_z}/μ_{E_z} minimization. Compared to the conventional optimization, therefore, the proposed multi-objective robust optimization approach can provide us with richer design information, especially the trade-off relations between optimality and robustness measures, in a comprehensive way through SOM visualizations.

Next, six solutions are selected from the SOMs (Figs. 7 and 8) and the tendencies for real tire stiffnesses of these solutions are compared with those of the baseline design. The selected solutions are: E_y-maximum solution, E_z-maximum solution and a compromised solution from the conventional optimization, (σ_{E_y}/μ_{E_y})-minimum solution, (σ_{E_z}/μ_{E_z})-minimum solution and a compromised solution from the robust optimization. The history of values for three tire stiffnesses E_x, E_y and E_z along the variation of tire internal pressure p_{int} is depicted in Fig. 9 ($E_{x,y,z}$ and p_{int} values are shown in percentages of the real baseline and nominal values, respectively, as calculated by ABAQUS).

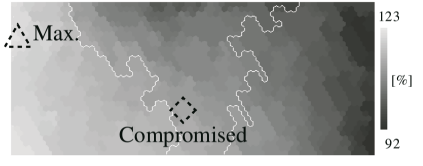
In the conventional optimization case shown in broken lines, both the E_y-maximum and the E_z-maximum solutions have better (i.e., larger) E_y and E_z at the nominal p_{int} than the baseline design, but almost the same E_y and E_z sensitivities against the variation of p_{int} as as those of the baseline design. In the compromised solution with all better E_x, E_y and E_z at the nominal p_{int}, the E_y and E_z sensitivities have not been improved either.

In the robust optimization case shown in solid lines, on the other hand, both the (σ_{E_y}/μ_{E_y})-minimum and the (σ_{E_z}/μ_{E_z})-minimum solutions actually have better (i.e., smaller) E_y and E_z sensitivities against the variation of p_{int} than the baseline design. Unfortunately, in these solutions, E_y and E_z values at the nominal p_{int} have not been improved. Here note that the compromised solution has not only all better optimality measures (E_x, E_y and E_z at the nominal p_{int}) but also all better robustness measures (E_y and E_z sensitivities against the variation of p_{int}) than the baseline, i.e. this design represents the so-called *sweet-spot* and is better than the baseline design in terms of all objective functions.

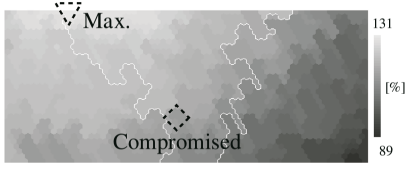
These results show that the proposed multi-objective robust optimization approach has enough accuracy to search for resultant robust optimal solutions. In addition, the proposed approach also has a superior capability to reveal not only trade-off relations



(a) E_x

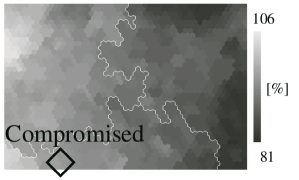


(b) E_y

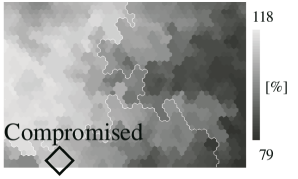


(c) E_z

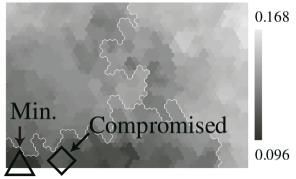
Fig. 7. SOMs of non-dominated solutions obtained through conventional optimization (shown in values estimated by RSs)



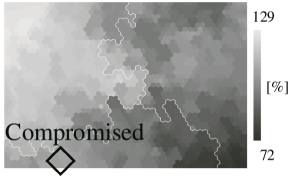
(a) μ_{E_x}



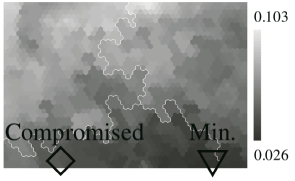
(b) μ_{E_y}



(c) $\sigma_{E_y} \triangleleft \mu_{E_y}$



(d) μ_{E_z}



(e) $\sigma_{E_z} \triangleleft \mu_{E_z}$

Fig. 8. SOMs of non-dominated solutions obtained through robust optimization (shown in values estimated by RSs)

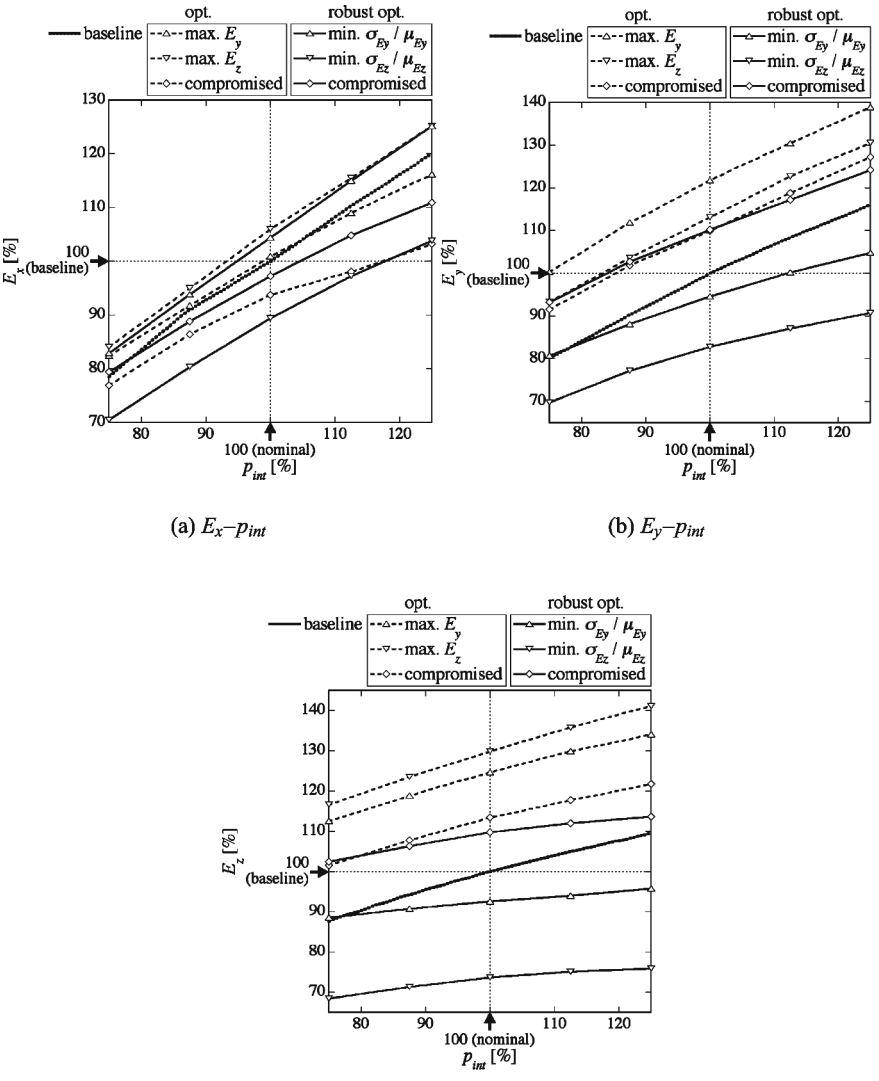


Fig. 9. History of tire stiffness values against the variation of p_{int} for non-dominated solutions obtained through conventional optimization and robust optimization (shown in real values as calculated by ABAQUS)

between optimality and robustness measures but also the location of sweet-spots, which is very useful in real-world engineering design problems with inevitable errors and uncertainties.

4.4 Validation of Efficiency to Perform Robust Optimization

The present robust optimization has performed 10^3 function evaluations by the MCS for each searching point, therefore the total number of function evaluations is

$$\begin{aligned} & 10^3 \text{ (MCS sample size)} \\ & \times 512 \text{ (MOEA population size)} \\ & \times 100 \text{ (number of generations in MOEA)} \\ & = 512 \times 10^5 \end{aligned}$$

The computational time required to evaluate a set of three tire stiffnesses through ABAQUS is about three hours, therefore the total function evaluation time increases to 175×10^2 years if all the function evaluations are performed directly by ABAQUS without RS approximation. On the other hand, the present robust optimization requires only

$$\begin{aligned} & 99 \text{ (initial response data points for RSs)} \\ & \times 7 \text{ (baseline and selected solutions)} \\ & \times 5 \text{ (number of computations to get an } E_{x,y,z}\text{–}p_{int} \text{ profile)} \\ & = 134 \end{aligned}$$

function evaluations using the ABAQUS until it provides the final data (Fig. 9), thus the total function evaluation time is reduced to less than 17 days. These results indicate the superior efficiency of the proposed multi-objective robust optimization approach as compared to the conventional approach.

5 Conclusions

An efficient approach for multi-objective robust design optimization, assisted by response surface (RS) approximation and visual data-mining (DM), has been proposed and then applied to an automobile tire design problem. Results indicate that the Kriging-based RSs helped the present approach to accurately predict objective function characteristics, and to reduce function evaluation time greatly compared to a direct optimization without RSs. In addition, the present SOM-based DM helped the present approach in the visualization of complicated design information between optimality and robustness measures in simple two-dimensional maps. Therefore, the use of Kriging model and SOMs realized a superior capability and efficiency to reveal the trade-off relations, and furthermore, sweet-spots between optimality and robustness measures in the design space. Finally, this study has proved the applicability of the present approach in real-world robust design problems with a large number of objective functions and expensive computations for function evaluation.

Acknowledgment

This study was partially supported through the Grant-in-Aid for Young Scientists (B), No. 19760098, 2007, by the Ministry of Education, Culture, Sports, Science and Technology of Japan. The first author would like to appreciate this support.

References

1. Gunawan, S., Azarm, S.: Multi-objective robust optimization using a sensitivity region concept. Structural Multidisciplinary Optimization 29, 50–60 (2005)
2. Li, M., Azarm, S., Aute, V.: A multi-objective genetic algorithm for robust design optimization. In: Proceedings of the 2005 Genetic and Evolutionary Computation Conference, pp. 771–778. ACM Press, New York (2005)
3. Deb, K., Gupta, H.: Searching for robust pareto-optimal solutions in multi-objective optimization. In: Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization, pp. 150–164. Springer, Heidelberg (2005)
4. Ong, Y.-S., Nair, P.B., Lum, K.Y.: Max–min surrogate-assisted evolutionary algorithm for robust design. IEEE Transactions on Evolutionary Computation 10, 392–404 (2006)
5. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Ltd., Chichester (2001)
6. Shimoyama, K., Oyama, A., Fujii, K.: A new efficient and useful robust optimization approach – design for multi-objective six sigma. In: Proceedings of the 2005 IEEE Congress on Evolutionary Computation, vol. 1, pp. 950–957. IEEE Press, Piscataway (2005)
7. Shimoyama, K., Oyama, A., Fujii, K.: Multi-objective six sigma approach applied to robust airfoil design for Mars airplane. AIAA Paper 2007–1966 (April 2007)
8. Ray, T.: Constrained robust optimal design using a multiobjective evolutionary algorithm. In: Proceedings of the 2002 IEEE Congress on Evolutionary Computation, vol. 1, pp. 419–424. IEEE Press, Piscataway (2002)
9. Jin, Y., Sendhoff, B.: Trade-off between performance and robustness: An evolutionary multiobjective approach. In: Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization, pp. 237–251. Springer, Heidelberg (2003)
10. Engineous Software, Inc., iSIGHT Reference Guide Version 7.1, pp. 220–233. Engineous Software, Inc. (2002)
11. Myers, R.H., Montgomery, D.C.: Response Surface Methodology: Process and Product Optimization Using Designed Experiments. John Wiley & Sons, Ltd., New York (1995)
12. Chen, W., Allen, J.K., Schrage, D.P., Mistree, F.: Statistical experimentation methods for achieving affordable concurrent systems design. AIAA Journal 33(5), 409–435 (1997)
13. Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Statistical Science 4(4), 409–435 (1989)
14. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box function. Journal of Global Optimization 13, 455–492 (1998)
15. Kohonen, T.: Self-Organizing Maps. Springer, Heidelberg (1995)
16. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics 21, 239–245 (1979)
17. Saliby, E.: Descriptive sampling: A better approach to Monte Carlo simulation. Journal of the Operational Research Society 41(12), 1133–1142 (1990)
18. Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Proceedings of the 5th International Conference on Genetic Algorithms, pp. 416–423. Morgan Kaufmann Publishers, Inc., San Mateo (1993)
19. Baker, J.E.: Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the 2nd International Conference on Genetic Algorithms, pp. 41–49. Morgan Kaufmann Publishers, San Mateo (1987)

20. Eshelman, L.J.: The CHC adaptive search algorithm: How to have safe when engaging in nontraditional genetic recombination. In: Foundations of Genetic Algorithms, pp. 265–283. Morgan Kaufmann Publishers, San Mateo (1991)
21. Tsutsui, S., Fujimoto, Y.: Forking genetic algorithms with blocking and shrinking modes (fGA). In: Proceedings of the 5th International Conference on Genetic Algorithms, pp. 206–213. Morgan Kaufmann Publishers, San Mateo (1993)
22. Eudaptics Software GmbH (2007), http://www.eudaptics.com/somine/ (cited January 10, 2007)

Multiobjective Metamodel–Assisted Memetic Algorithms

Chariklia A. Georgopoulou and Kyriakos C. Giannakoglou

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
kgianna@central.ntua.gr

The hybridization of global metaheuristics, such as evolutionary algorithms (EAs), and gradient-based methods for local search, in the framework of the so-called memetic algorithms (MAs) can be used to solve multi-objective optimization problems, either in the Lamarckian or Baldwinian spirit. Reducing the CPU cost of MAs is necessary for problems with computationally demanding evaluations. For the same purpose, in EAs, metamodels are in widespread use, giving rise to various kinds of metamodel-assisted EAs (MAEAs). Metamodels are surrogate evaluation models of various types: multilayer perceptrons, radial basis function networks, polynomial regressions models, kriging, etc. A good practice is to use local metamodels, trained on the fly for each new individual, using selected entries from a database where all the previously evaluated offspring are recorded. The selection of suitable training patterns is important in order to minimize the prediction error of the metamodel. The MAEA developed by the authors in the past uses the inexact pre–evaluation (IPE) technique which starts after running a conventional EA for just a few generations on the exact evaluation model. The exactly evaluated offspring are all stored in the database. For the subsequent generations, local metamodels are trained for each new offspring to get an approximation to the objective functions so that, based on it, a few top individuals (in the Pareto front sense) are selected for exact re-evaluation.

The availability of local metamodels for use in the EA was the basis for creating the metamodel-assisted memetic algorithm (MAMA). Local metamodels are used to approximate also the objective functions gradient to be used during local search. Among other, this chapter focuses on MAMAs that incorporate utility assignment techniques (based on dominance and strength criteria), how to choose target or targets for local refinement, the appropriate selection of training patterns, the management of outlying individuals and ways to handle the outcome of the local search (back-coding the genotype and/or phenotype).

The proposed MAMA is demonstrated on function minimization problems, the design of a gas turbine power plant with three objectives and the two–objective aerodynamic design of a cascade airfoil. In all cases, radial basis function networks are used as metamodels.

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 153–181.
springerlink.com

1 Introduction

1.1 Metamodel–Assisted Evolutionary Optimization

During the last decade, optimization methods based on or assisted by approximation evaluation models became of particular interest, being routinely used by industries which strive to produce new designs with shortened design cycle and minimum cost. This class of optimization methods relies on the use of low-cost analytical or semi–analytical models (to be referred to as *metamodels* or *surrogate evaluation models*) instead of the computational demanding, problem-specific software which should otherwise undertake the evaluation of candidate solutions. The extensive development and use of metamodel–assisted optimization methods is one of the reasons that research on metamodels is still going on; recall that most of the approximation models in use have been developed many years ago as general purpose tools. Current research on metamodels focuses on the development of new models with better generalization capabilities, faster training procedures and more dependable validation processes. In modern metamodel–assisted optimization algorithms, the way of using the metamodels (selection of training patterns, decision making based on fitness values provided by the metamodel, etc.) is much more important than the selection of the metamodel type. No doubt that the best way to fit a metamodel into an optimization method is still an open question. Metamodels are associated with single- and multiobjective optimization, optimization with more than one disciplines or under uncertainties.

There is a variety of metamodels capable to assist an optimization method, [24], [37], [41], [19]. Among them, polynomial response surface models, artificial neural networks including radial basis functions and multilayer perceptrons, Gaussian processes including Kriging, support vector machines, etc. Depending on the metamodel chosen, various training methods can be used. The possible ways of incorporating metamodels (used as surrogates to the evaluation model) within an evolutionary optimization method (Evolutionary Algorithms, *EAs*, Genetic Algorithms, *GAs*, Evolution Strategies, *ES*, to be referred to as Metamodel–Assisted Evolutionary Algorithms, *MAEAs*), can be classified as follows:

***EAs* based on off-line trained metamodels with no feedback:** During a preparatory phase, an adequate set of training examples is selected using a search space filling method [6] and evaluated. Measurement techniques and/or computational models can be used to evaluate the selected samples. It is obvious that such an algorithm is perfectly suited to applications where a new round of evaluations is impossible. The so-trained global metamodel can be used over the entire search space and is considered to be dependable. There is no possibility for feedback to update it. The optimization algorithm relies exclusively on the metamodel. No doubt that false optima might be located if an inappropriate metamodel (trained, for instance, on datasets that poorly populate the design space) is used.

***EAs* based on off–line trained metamodels with feedback:** The *EA*–based search uses a global metamodel which is regularly and incrementally updated (after a certain number of generations or according to user–defined criteria) on memorized

evaluations carried out during the evolution, [43], [54]. The generation–based evolution control algorithm proposed in [35] is a notable scheme. Starting from a randomly generated initial population with exactly evaluated members as in any conventional *EA*, a first metamodel is constructed. During the next generations, a low–cost *EA*–based search on this metamodel is performed until a convergence criterion is satisfied. Upon completion of this search, the population of the next generation is evaluated using the exact tool and the metamodel is updated to account also for the new training examples which became available. These steps are repeated as long as a global convergence criterion is not attained. Instead of being constant, the frequency with which the metamodel is updated can be controlled by the progress of evolution giving thus rise to adaptive schemes, [46]. Note that the construction of a global approximation model for the entire search space, once or repetitively during the optimization, is a delicate process in highly dimensional spaces and requires an adequate set of carefully selected training examples.

***EAs* based on on-line trained local metamodels:** A worthwhile alternative of *EAs* based on off–line trained metamodels is any *EA* that makes use of local metamodels, [19], [25], [26], [27], [16], [23], [36], [31]. The evaluated individuals are memorized and metamodels are trained separately for each new population member on its closest, previously evaluated solutions. The metamodels are practically used to identify promising population members (during the so–called Inexact Pre–Evaluation, *IPE*, phase) which are worth being evaluated using the exact evaluation tool. Thus, both approximate and exact evaluation models are used in each generation. These methods can be classified further into those selecting either some individuals at random, [46], or the best (as evaluated by the metamodel) ones, [20], for exact evaluation.

1.2 *EAs* and Local Search – From *MAEAs* to *MAMAs*

The complexity of many real world problems calls for the hybridization of optimization methods which leads to more efficient search tools. In the framework of a global optimization method, the search can be focused within the local neighborhood of some promising solutions. This gives rise to the so–called *Hybrid* or *Memetic Algorithms* (*MAs*). Whenever optimal solutions of high accuracy are required, local search can do the job through the refinement of quasi–optimal solutions located by the global search method, [5], [11], [12], [10].

Despite their conceptual simplicity, *EAs* are effective global search metaheuristics which, after reducing the computational cost by implementing metamodels (*MAEAs*), may become quite efficient. For this reason, they could certainly be (are, in fact) used as global search tools within a *MA*. Various local search methods can be associated with an *EA* (such as simulated annealing, gradient–based methods, etc.). Consequently, a *MA* is defined as an *EA* that employs refinement processes (based on local search) for individuals generated according to the Darwinian rules, [2]. Local search is conceptually associated with "memes" which represent a local learning strategy or a unit of cultural evolution, [1]. A meme is associated with cultural information which reproduces itself, while individuals cooperate and compete by exchanging ideas, [56]. In computer

science, any local search stands for the "learning" process which aims at refining existing individuals by forcing them to "imitate" and obtain successful behaviors. The ultimate goal is to create the optimal co–adapted set of information through the use of evolutionary operators, [56]. From a practical point of view, *MAs* are hybrid optimization systems based on the combination and mutual interaction of a global search method and a knowledge–driven (local search) tool. The so–devised *MA* possesses exploration and exploitation capabilities in equilibrium: we need exploration to be sure that no part of the search space remains unexplored and there is no possibility to miss a global optimal solution located there. Exploitation helps focusing around promising solutions found during previous search which, through refinement, may lead to global optimal solution(s).

Incorporating learning into an *EA* can be done according to the Lamarckian or Baldwinian rules, [51]. In the Lamarckian spirit, the outcome of learning affects/reflects on the genetic structure of the individual. In practice, this back–coding entails that both the improved genotype and the fitness values substitute the initial ones in the population and, hence, every successful behavior becomes inheritable, [42]. In contrast, the Baldwinian learning affects only the fitness value vector and avoids genotype replacements. It acts as if parent selection occurs on a smoothed fitness landscape which is proved to be helpful for the evolutionary process, [42].

A literature survey on *MAs* for multiobjective optimization (*MOO*, *M* objectives) problems leads only to a few relevant works. In [45], all candidate solutions undergo local search to improve the weighted sum of the *M* objective functions, with random weights for each individual. The CPU cost is balanced between global and local search. The latter terminates after a user–defined number of exact evaluations which fail to generate a new locally non–dominated individual. In [21], the method was modified so that only the most promising individuals undergo local search.

In [22], a genetic local search algorithm for multiobjective combinatorial optimization (such as the traveling salesman problem) is proposed. In each iteration, the best members of the population form a temporary population, from which two parents are selected at random and undergo recombination to create a new offspring. The latter is optimized by local search. If the outcome of local search outperforms the worst solution in the temporary population, this is added to the population and the archival set.

In [48], the Pareto Archived Evolution Strategy (PAES), which maintains a finite–sized archive of non–dominated solutions, is used in place of local search tool in a memetic algorithm framework. The global search procedure is based on crossover operators applied to a population of candidate solutions, which undergo local search and replace their origin in the Lamarckian spirit.

Local search may have a significant CPU cost as it implies so many calls to the evaluation tool. As a result, an *EA* combined with local search is expected to become costly enough. To suppress a great amount of evaluations, one may incorporate surrogate models, similar to those used in *MAEAs*, to also assist the local search. So, in a view, the method presented herein is based on the same concept with *MAEAs*; cheap surrogate models undertake not only the inexact pre–evaluation of the population members but the local search as well. Thus, the proposed optimization method is to be referred to as

Metamodel–Assisted Memetic Algorithm (*MAMA*). Without loss in generality, Radial Basis Function (*RBF*) networks are used as metamodels.

Since this chapter is concerned with a *MOO* method producing Pareto fronts, a utility assignment technique must be adopted. Its role is to assign a unique value to each population member, depending on the M cost function values and dominance criteria. The selected technique is the widely used *SPEA*2 (Strength Pareto Evolutionary Algorithm, [58]). Theoretically, any other "rival" method (among those existing in the literature) can be used instead.

The structure of this chapter is as follows: the designed *MAMA* is presented by putting emphasis on the appropriate selection of training patterns so as to improve the prediction capabilities of the surrogate models, which is important for the local search. A by–product of the pattern selection process is to indicate regions in the design variables' space which have inadequately been explored thus far. In *MOO*, local search targets at improving one of the objectives by considering all other objectives as appropriate inequality constraints so as to avoid worsening their fitness or cost values. At the end of local search, the improved individuals are re–injected into the population by replacing existing genotypes, according to the spirit of Lamarckian learning.

The proposed *MAMA* will be demonstrated on function minimization problems, the design of a gas turbine power plant with three objectives and the two–objective aerodynamic design of a cascade airfoil.

For the sake of clarity, a list of abbreviations follows:

EA	Evolutionary Algorithm
MAEA	Metamodel–Assisted Evolutionary Algorithm
MA	Memetic Algorithm
MAMA	Metamodel–Assisted Memetic Algorithm
RBF	Radial Basis Function
IPE	Inexact Pre–Evaluation
SOO	Single Objective Optimization
MOO	Multiobjective Optimization
SOM	Self Organizing Map
MST	Minimum Spanning Tree
SPEA	Strength Pareto Evolutionary Algorithm
DB	Database (of exactly evaluated solutions)

2 Metamodel Assisted Memetic Algorithm (*MAMA*)

Prior to the presentation of the algorithmic details of the proposed *MAMA*, its key features which distinguish it from other similar methods (see section 2.2) are listed below:

(a) The core search engine is a (μ,λ) *EA* with μ parents, λ offspring, N design variables and M objectives. At each generation (subscript g), the EA handles three population sets, namely the parent set ($\mathscr{P}_{\mu,g}$, $\mu=|\mathscr{P}_{\mu,g}|$), the offspring ($\mathscr{P}_{\lambda,g}$, $\lambda=|\mathscr{P}_{\lambda,g}|$) and the archival one $\mathscr{P}_{\alpha,g}$. The latter stores the $\alpha=|\mathscr{P}_{\alpha,g}|$ elite individuals (among the non–dominated in the Pareto front sense, in *MOO*) found thus far.

(b) In *MOO* problems, fitness assignment is based on the *SPEA*2 technique. Thinning, for truncating highly populated archival sets, is additionally used. If $\mathscr{P}_{\alpha,g}$ size exceeds a user-defined value a_{max}, [58], excess members are eliminated according to density–based criteria.

(c) *EAs* are assisted by *RBF* networks which act as local surrogate evaluation models. The same metamodels participate in both the *IPE* and the local search phase of the *MAMA*. Exact evaluations are carried out for a few top ranking population members and some of the outliers (population members which lie in unexplored regions of the search space, with insufficient information in their vicinity), if any. In contrast to previous *MAEAs*, [26], here the number of exact evaluations per generation remains fixed.

(d) A database (*DB*) of exactly evaluated solutions (paired inputs/outputs) is kept to avoid re-evaluating any previously seen individual and providing data for training the *RBF* networks. At the beginning of the evolution the *DB* is empty; the few first generations (usually, no more than 2 or 3) rely exclusively on exact evaluations to collect the minimum data that must be available before starting the *IPE* phase. During the subsequent generations, only the exactly evaluated individuals are archived in the *DB*. Note that penalized cost values are stored in the *DB*.

(e) In constrained optimization, slightly or moderately violated constraints are handled through exponential penalty functions aggravating the cost function value. In *MOO*, all cost values are multiplied by the same penalty. On the other hand, heavily violated constraints (depending on an arbitrary user–defined threshold value) lead to death penalties, i.e. the M cost functions take on the same "infinite" (for instance, 10^{20}) value.

(f) In constrained optimization, the *IPE* of population members is carried out according to a two-step procedure. In the first step, previously evaluated feasible solutions are used to estimate the fitness values and, then, the infeasible ones estimate the penalty, if any.

(g) Local search is based on the Augmented Lagrange Multipliers (*ALM*, [7]) method using approximately computed derivatives. Each local search costs as many as two exact evaluations, one for the starting and the other for the resulting individual. The top ranking population members that have already been selected for exact evaluation may undergo local search for the purpose of refinement. Outliers are excluded from local search since the corresponding metamodel is considered not to be dependable.

In the remaining of this chapter, the actions taken within each generation of the multiobjective *MAMA* are described in detail. This algorithm is not applied during the starting few generations (as explained in (d)) for which all evaluations are exclusively based on the exact tool.

2.1 The Proposed *MAMA* in Detail

Offspring evaluation: Aiming at reducing the number of "useless" evaluations within the proposed *MAMA*, the following steps are considered:

1. *Selection of Training Patterns & Preparation for IPE:*
   For each offspring, $\mathbf{x} \in \mathscr{P}_{\lambda,g}$, the most appropriate set of training patterns is selected from the *DB* to train the corresponding metamodel, see Algorithm 1. The selection of training patterns for each local metamodel is necessary even for population members in the vicinity of which the available *DB* information is poor. The so–called outliers are marked as candidates for exact evaluation and are excluded from becoming the origin of local search. Selecting some of the outliers for exact evaluation, even if these were not among the top ranking individuals according to the *IPE* screening, enhances diversification and leads to balanced exploration and exploitation. To sum up, the trained local *RBF* networks are used for all the members of $\mathscr{P}_{\lambda,g}$, except those found in the *DB*.
2. *IPE for Constrained Optimization:*
   IPE aims at computing approximate cost function values, $\tilde{f}(x)$. In constrained *MOO*, the *IPE* phase consists of the following steps for each and every population member:
   - "Feasible" Score Estimation: For each population member, the $K_{feasible}$ "feasible" individuals among the K selected training patterns (defined as those with $f_m \leq f_{m,thresh}, \forall m \in [1,M]$) are used to produce an estimation of its fitness function values without considering penalties due to any constraint violation. An *RBF* network is trained on the $\mathrm{K}_{feasible}$ solutions and $\tilde{\tilde{f}}_m$ is, thus, obtained.
   - Penalty Estimation: For the remaining $K_{infeasible}$ ($K_{infeasible}$=K–$K_{feasible}$) patterns, a distance based exponential decay of the value that quantifies the constraint violation is assumed. For the k_{th} closest "infeasible" training pattern, the quantity $p_m^k = f_m^k(e^{-\frac{l(k-1)}{K_{infeasible}}} - \frac{l(k-1)}{K_{infeasible}}e^{-l})$, where l is a natural number, is the corresponding penalty coefficient assigned to the objective m of the new individual.
   - Cost Value Estimation: The previously computed $\tilde{\tilde{f}}_m$ values for this population member is multiplied by the maximum of all penalty coefficients: $\tilde{f}_m = \max_k(1, p_m^k)\tilde{\tilde{f}}_m,\ k \in [1,\ \mathrm{K}_{infeasible}]$.
3. *Selection of Individuals for Exact Evaluation:*
   Based on the $\tilde{f}(x) = (\tilde{f}_1, \ldots, \tilde{f}_M)$ values computed by the metamodel, a provisional utility function value $\tilde{\phi}$ is assigned to each $\mathbf{x} \in \mathscr{P}_{\lambda,g}$, namely:

$$\tilde{\phi}(\mathbf{x}) = \tilde{\phi}(\tilde{\mathbf{f}}(\mathbf{x}), \{\tilde{\mathbf{f}}(\mathbf{z}) \mid \mathbf{z} \in \mathscr{P}_{\lambda,g} \setminus \{\mathbf{x}\}\}).$$

   In *SOO*, this assignment is straightforward ($\tilde{\phi}(\mathbf{x}) = \tilde{f}(\mathbf{x})$). In *MOO*, an assignment according to the *SPEA*2, [58], technique is used. Note that some of the population members might have been given exact cost values, because these have been extracted from the *DB*. Then, the pool $\mathscr{P}_{e,g}$ of individuals that are worth an exact evaluation is formed. The total number $\lambda_e = \eta_1\lambda$ ($\eta_1 < 1$) of exact evaluations per generation is user–defined and fixed, unless more than $\lambda - \lambda_e$ population members have been found in the *DB*. This algorithm doesn't mandate that only the most promising individuals, according to the metamodel–based evaluations, be exactly evaluated. In fact, it is possible that some of the outliers might also be exactly evaluated, but this would occur with a low

probability. Practically, the user defines the maximum percentage $\frac{\lambda_{eo}}{\lambda_e} = 1 - \eta_2$ ($0 < \eta_2 < 1$) of exact evaluations which might be assigned to the outliers. Thus, the number of exact evaluations in each generation is $\lambda_e = \lambda_{et} + \lambda_{eo}$, accounting for the λ_{et} top ranking individuals (on the metamodel) and λ_{eo} outliers (selected at random, if more than λ_{eo} outliers exist).

4. *Exact evaluations:*
   For each $\mathbf{x} \in \mathscr{P}_{e,g}$, exact cost function values $\mathbf{f}(\mathbf{x})$ are computed and stored in the *DB*.

Selection for local search: Individuals that are worth an extra local search are selected among the $\mathscr{P}_{\lambda,g}$ members for which an exact cost is available. Note that these may have been either generated in the current generation or pre–exist (found in the *DB*, without being locally searched/refined). Individuals for which the corresponding metamodel is marked as questionable (see Algorithm 1) are excluded from local search. The (upper) number λ_{ls} of local searches allowed per generation is user–defined. To select the λ_{ls} individuals for local search (if more than λ_{ls} candidates exist) a scalar utility function value is computed using dominance criteria and the strength, as defined in the *SPEA*2 method.

Local search: Each individual $x \in \mathscr{P}_{ls}$ is locally optimized, using the trained *RBF* network, as explained below:

1. *Prerequisites for local search:*
   The training patterns for x are increased by one through adding x itself. One of the objectives is selected for further refinement. Three possible criteria have been tested: (a) selection of the most "advanced" among the objectives, as determined by considering the relative location of x on the current front of non–dominated solutions, (b) selection of the "less advanced" and (c) random selection. None of them seems to clearly outperform the rest so, in the examined cases, we will refrain from making a clear distinction among them.
2. *Local Search:*
   During the local search, constraints related to the design variables' bounds and objectives (other than the one selected to be optimized in local search) are taken into account. The first class of constraints is related to the lower and upper bounds of the design variables,

$$\mathscr{A}: \quad x \in [x_{i,min}, x_{i,max}] \; \forall i \in [1, N]$$

   The second class serves to ensure that the outcome of local search (i.e. a new candidate solution) will not damage the cost function values of the starting individual $\mathbf{x}_{init}$ for all the objectives. It is expected that the new solution must dominate the current individual (i.e. the starting point of local search) or, at least, be a "partial improvement" (in the sense that some of the objectives worsen and some other improve). However, the aforementioned goal, which is mathematically expressed as

$$\mathscr{B}: \quad f_j(\mathbf{x}) \leq f_j(\mathbf{x}_{init}), \; \forall j \in [1, M]$$

is not always achieved since local search is based on the *RBF* network instead of the exact evaluation tool. Then, a gradient–based method capable of dealing with constraints (*ALM*) is used. Let $q, (\ q \in [1, M])$ be the objective to be optimized. According to the *ALM* method, the constrained optimization problem:

$$\min f_q \ \text{ st. } \mathscr{A}, \mathscr{B}$$

is replaced by an unconstrained one:

$$\min g_q^{ALM} \tag{1}$$

where g_q^{ALM} incorporates all of the constraints multiplied by three vectors of Lagrange multipliers $(\lambda_1, \lambda_2, \lambda_3)$. g_q^{ALM} is defined as

$$\begin{gathered} g_q^{ALM}(\mathbf{x},\ \lambda,\ \omega_p) = f_q(\mathbf{x}) + \textstyle\sum_{j=1}^{M\setminus q}(-\lambda_{3,j}\psi_{3,j} + \omega_p\psi_{3,i}^2) \\ + \textstyle\sum_{i=1}^{N}(-\lambda_{1,i}\psi_{1,i} + \omega_p\psi_{1,i}^2 - \lambda_{2,i}\psi_{2,i} + \omega_p\psi_{2,i}^2) \end{gathered}$$

where

$$\begin{gathered} \psi_{1,i} = \max[(x_{i,min} - x_i),\ \tfrac{\lambda_{1,i}}{2\omega_p}] \\ \psi_{2,i} = \max[(x_i - x_{i,max}),\ \tfrac{\lambda_{2,i}}{2\omega_p}] \\ \psi_{3,j} = \max[(f_j - f_{j,init}),\ \tfrac{\lambda_{3,j}}{2\omega_p}] \\ \lambda_{1,i} \leftarrow \lambda_{1,i} - 2\omega_p\psi_{1,i} \\ \lambda_{2,i} \leftarrow \lambda_{2,i} - 2\omega_p\psi_{2,i} \\ \lambda_{3,j} \leftarrow \lambda_{3,j} - 2\omega_p\psi_{3,j} \\ \omega_p \leftarrow \min(k_{\omega_p} * \omega_p, \omega_{p,max}), \\ \text{where } k_{\omega_p} \geq 1 \end{gathered}$$

Upon termination of the local search, the new individual $\mathbf{x}^*$ is re–evaluated using the exact tool and archived in the *DB*.

3. *Final Decision:*
If $\mathbf{x}^*$ dominates any of the $\mathscr{P}_{a,g}$ members, these are all eliminated from $\mathscr{P}_{a,g}$. An individual that enters $\mathscr{P}_{a,g}$ also enters $\mathscr{P}_{\lambda,g}$ by displacing $\mathbf{x}_{init}$. $\mathbf{x}^*$ may also enter $\mathscr{P}_{\lambda,g}$ (without however joining $\mathscr{P}_{a,g}$) if this is a (full or partial) improvement of $\mathbf{x}_{init}$.

Elitism: The elite operator E acts on $\mathscr{P}_{e,g}$, instead of $\mathscr{P}_{\lambda,g}$, and the previous elite archive, as follows

$$\mathscr{P}_{a,g+1} = \mathrm{E}(\mathscr{P}_{e,g} \cup \mathscr{P}_{\alpha,g}). \tag{2}$$

Evolution operators: New parents are selected,

$$\mathscr{P}_{\mu,g+1} = \mathrm{S}(\mathscr{P}_{\mu,g} \cup \mathscr{P}_{\lambda,g}) \tag{3}$$

and new offspring are generated via crossover, C, and mutation, M,

$$\mathscr{P}_{\lambda,g+1} = \mathrm{M}(\mathrm{C}(\mathscr{P}_{\mu,g+1}, \mathscr{P}_{\alpha,g+1})) \tag{4}$$

Termination: Unless termination criteria are satisfied, $g \leftarrow g+1$ and evolution continues from the offspring evaluation step.

2.2 Other *MAMAs* – A Literature Survey

In the literature, one may find a few papers on surrogate–assisted *MAs*; these papers are briefly overviewed below. In [52], a hybrid algorithm that combines fitness landscape approximation and *EAs* with local search is proposed. The population is initialized using a conventional *EA*. When an adequate *DB* of exactly evaluated solutions is available, individuals cooperate in building quadratic models, the minimum of which undergoes local search, hoping that, after learning, a better minimum will be reached. According to the authors, the main idea is to use information from local minima in building approximate models and, then, seek improved solutions in the deepest valley at the most promising search direction. Similar algorithms can be found in [28] and [44]. The hybrid optimization approach presented in [44] performs local search in specific points of the design space. The increased relative homogeneity of the design space constitutes an indication that the evolutionary process has converged to a small area, which warranties that a local search can detect local optima. If such a region is detected, a polynomial model is fitted to the relevant offspring members and local search initiates from the minimum point of this surrogate.

In [30], the metamodels are used in a two–fold manner. The algorithm starts by building a *DB* with a sufficient number of exact evaluations, while a conventional *EA* is running. Then, a global metamodel takes on the evaluation process, which is built on the top ranking archived design points in the *DB*. Among the individuals evaluated using the approximate tool, only the promising ones are refined using Lamarckian learning. The algorithm is based on interleaving the exact evaluation tool with local, online trained *RBF* networks and the *DB* is refilled during local search. The global metamodel changes whenever the top set of *DB* entries is modified. Although the method presented in [30] has many similarities with the present *MAMA*, important differences can be found. Exact evaluations are almost restricted in areas close to members associated with local search, reducing thus the probability to detect optimal solutions located far apart from these areas. In [31], multiple surrogates displace the exact evaluation tool during the local search leading to a multi–surrogate assisted *MA*. The impact of uncertainty on the convergence, due to the use of approximation tools is discussed and investigated on single objective mathematical functions. Multiple searches based on different metamodels are used for the same search origin and only the best outcome is inserted into the offspring population in the Lamarckian spirit. If all surrogates display problems of uncertainty in predicting this point, the most beneficial is chosen and used; otherwise, the one with the minimum prediction error must be chosen.

3 The *RBF* Networks as Metamodels

In the present *MAMA*, *RBF* networks serve as metamodels to perform the inexact pre–evaluation of the population members and, in addition, approximate gradient vectors used in local search. In this section, the structure and the training procedure for the *RBF* networks are presented in brief and the algorithm used to select the training patterns is described in detail. The reason for selecting *RBF* networks is two–fold: First, the *RBF* networks are universal approximators, [33], capable of constructing an "exact" mapping from the design to the objective space, once an adequate set of training patterns

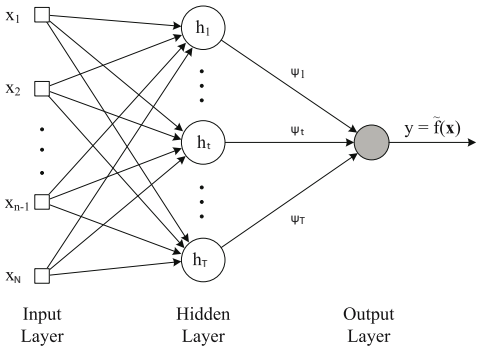


Fig. 1. *RBF* network with N inputs, T hidden neurons and a single output ($M=1$)

is available. Second, their training procedure reduces to the solution of a system of linear algebraic equations per response (objective). The use of *RBF* networks in *MOO* is advantageous (compared to the multilayer perceptron, for instance), since the training of an *RBF* network with M output units (M objectives) is equivalent to that of M networks with a single output. In such a case, a single matrix inversion is required.

3.1 Network Structure and Training Procedure – Comments

An *RBF* network that performs the mapping $\mathscr{R}^N \rightarrow \mathscr{R}^M$, [34], [3], has three layers of processing units, fig. 1: input layer with N nodes where the input vectors are applied to, the hidden layer with T processing neurons and the output layer with M nodes where the network responses appear. Signals propagate through the network in the forward direction by performing a nonlinear mapping from the input to the hidden layer followed by a linear mapping to the output nodes. The latter takes into account the values of the synaptic weights ψ computed during the training phase. The T hidden layer units, which are fully connected to the N input and M output units, are associated with the so–called *RBF* centers, $c^{(t)}$, $t=1,T$. The selection of *RBF* centers is crucial and affects strongly the prediction capability of the network. There are two possible ways. In the first, once the training patterns have been selected ($x^{(k)}$, $t=1,K$), these are also used as *RBF* centers ($c^{(k)}=x^{(k)}$; here, $T=K$). In the second, less *RBF* centers than training patterns are selected by minimizing an appropriate error function, [34], [8], [9], aiming at better generalization properties.

In the past, [26], the need for carefully selecting the *RBF* centers was demonstrated. A selection scheme based on self–organizing maps, *SOM*s was employed. The role of *SOM*s, [17], [3], is to act as a clustering mechanism that helps associating the *RBF* centers with the so–defined clusters. The placement of the *RBF* centers via *SOM*s is based on three distinct phases (competition, cooperation and adaptation). The training of the generalized *RBF* networks comprises two levels of learning (unsupervised and supervised), integrated in an iterative algorithm. During the unsupervised learning, the *RBF* centers are determined via *SOM*s whereas the *RBF* radii (r_t, in eq.5) via heuristics which take into account distances between the centers, [29], [3], [55]. During the

supervised learning, the synaptic weights are computed by minimizing the approximation error over the training set, while considering smoothness requirements.

Although this method has been presented by the same group, [26], in the proposed *MAMA*, the local *RBF* networks are used for the (exact) interpolation of the training patterns and, thus, the *RBF* centers must coincide with the training patterns. This was decided since the *RBF* networks are also used to get an approximation of gradients which are expected to be more accurate if the network interpolates, rather than approximates, the training patterns.

Whenever a training pattern $x^{(k)}$ is presented to the input layer, the hidden units take on values:

$$h_t^{(k)} = \Phi\left(\left\|x^{(k)} - c^{(t)}\right\|_2, r_t\right) \tag{5}$$

where $\Phi(d,r) = e^{-d/r}$ is the activation function. The network output $\tilde{f}^{(k)}$, which is the desired approximation to the exact objective function $f^{(k)}$, is a linear function of the hidden unit values

$$\tilde{f}_m^{(k)} = \psi_{t,m} h_t^{(k)},\ m=1,M \tag{6}$$

where summation applies to the repeated index t. The weights ψ_t, $t = 1,T$, are computed by minimizing the local error

$$E^{(k)} = \frac{1}{2}\sum_{m=1}^{M}(\tilde{f}_m^{(k)} - f_m^{(k)})^2 = \frac{1}{2}(\| \tilde{f}^{(k)} - f^{(k)} \|_2)^2 \tag{7}$$

The trained *RBF* network provides also its own estimates for the sensitivity derivatives of the output function with respect to the input parameters,

$$\frac{\partial \tilde{f}_m(x^*)}{\partial x_\rho} = \psi_{t,m}\frac{\partial \Phi\left(\left\|x^* - c^{(t)}\right\|_2, r_t\right)}{\partial x_\rho} \tag{8}$$

3.2 Selection of Training Patterns

Due to the importance of using dependable *RBFs* during the *IPE* phase and the local search, emphasis is put on the appropriate selection of training patterns and, consequently, the *RBF* centers. The K training patterns should all lie in the vicinity of the

Table 1. User–defined parameters for the training patterns' selection algorithm

K_1	Max. number of candidate training patterns for the *RBF* network (size of starting pool).
K_2	Min. number of training patterns for the *RBF* network.
K_3	Max. number of training patterns for the *RBF* network.
$\rho_1 = \rho_1(N)$	Min. allowed distance between any two training patterns.
$\rho_2 = \rho_2(\rho_1)$	Trust region radius for the *RBF* network.
$\eta_1(*100\%)$	Percentage of the offspring population to be exactly (re)evaluated (after the *IPE* phase).
$\eta_2(*100\%)$	Percentage of the number of exact evaluations the top ranking offspring (as evaluated by the metamodel) may undergo; $(1-\eta_2)*100\%$ of them at most can be selected among the outliers, if any.

new individual and are selected according to Algorithm 1. To facilitate its presentation, the user–defined parameters are listed in table 1.

for each *new* individual in the current population **do**
  -Pool Formation: Select its K_1 closest individuals from the *DB* and define a first pool of training patterns.
  -Pool Processing: Using the K_1 individuals and the *new* one build the corresponding Minimum Spanning Tree (*MST*, [4]).
  -Decision for local search: Identify the closest individual to *new*.
  if this was the starting or ending point of a previous local search **then**
    new takes on its cost function value(s). No *RBF* is to be trained.
  else
    -Thinning process: Eliminate individuals from the current pool if the distance between them is less than ρ_1. Practically, from each pair of individuals with distance less than ρ_1, the one with the minimum distance to any other individual in the pool is eliminated. During the iterative thinning process, the initial *MST* is continuously updated. The process terminates if there are no individuals at distance less than ρ_1 or the minimum allowed number of non–eliminated individuals (K_2) has been reached. Let K be the number of patterns remaining in the pool after thinning ($K_2 \leq K \leq K_1$).
    -Initial training set: The K_2 individuals in the pool that are closer to *new* are copied to the training set.
    -Labeling the new individual:
    if the closest pattern to *new* is at distance greater than ρ_2 **then**
      this individual is marked as *outlier*; although an *RBF* will be trained and used during the *IPE* phase, no local search will be allowed for *outliers*. An *outlier* can be exactly evaluated with probability $(1\text{-}\eta_2)\eta_1 100\%$, at most.
    else
      if the K_2 closest patterns are at distance less than ρ_2 **then**
        new is marked as *dependable*. The training set is additionally populated up to K_3 individuals at most. *Dependable* individuals can be selected for local search.
      else
        new is marked as *questionable*. No local search is allowed for individuals associated with a questionable metamodel.
      end if
    end if
  end if
end for

Algorithm 1. Training Pattern Selection

4 Method Demonstration – Applications and Discussion

4.1 *ZDT*3 – A Benchmark *MO* Case

A two–objective mathematical problem known as *ZDT*3, [38], is first analyzed. The target is to minimize F_1 and F_2 defined by

$$F_1(\mathbf{x}) = x_1, \quad F_2(\mathbf{x}) = h(\mathbf{x})G(\mathbf{x})$$

where $x_i \in [0,1]$, $N = 30$ and

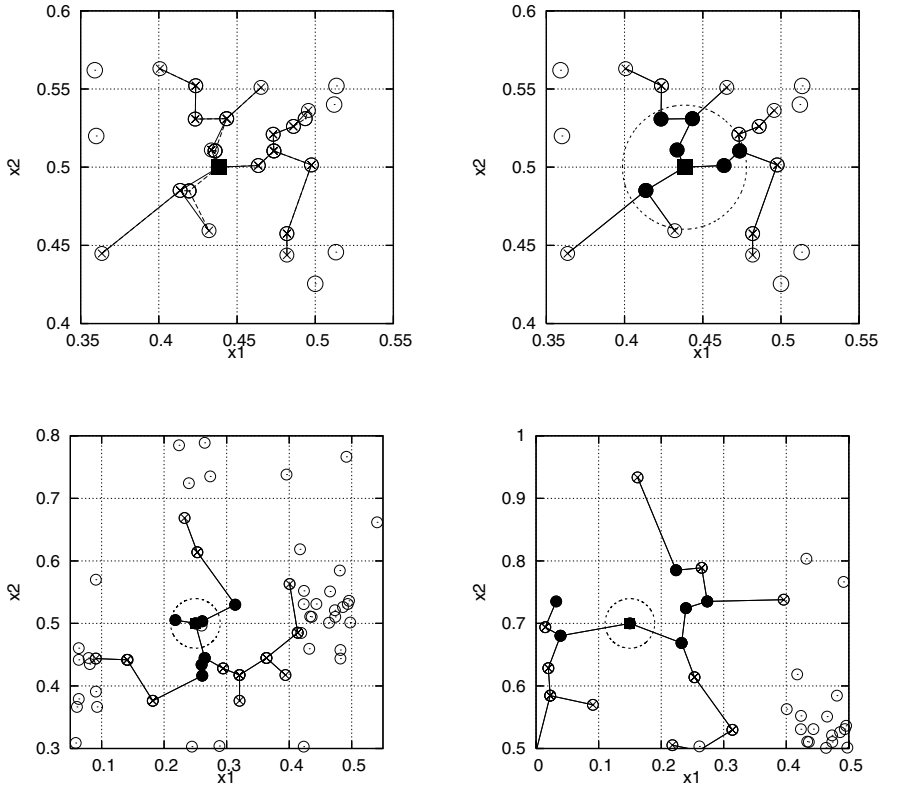


Fig. 2. Snapshots of the training pattern selection process corresponding to three different cases with two design variables each. In each case, the new individual is denoted by a black square. Empty circles correspond to *DB* entries; the ones marked with a cross designate the final pool (after the thinning process) which the training patterns are chosen from. The final training set is shown with filled circles. Minimum spanning trees are also shown along with a circular area of radius ρ_2 centered at the new individual. The two figures at the top correspond to a case that leads to an *RBF* network trained on a dependable set of patterns. The thinning process (top–left) is followed by the final training pattern selection (top–right) which eliminates some *MST* nodes being far apart from the new individual. The case shown at bottom–left leads to an *RBF* network with a questionable set of training patterns (many of them fall outside the circular area of radius ρ_2) which is likely to produce erroneous cost function guesses. Finally, a case is shown (bottom–right) with an outlier for which, although an *RBF* network has been trained, this cannot be used to support local search.

$$\mathrm{G}(\mathbf{x}) = 1 + \frac{9\sum_{i=2}^{N} x_i}{N-1}$$

$$h(\mathbf{x}) = 1 - \sqrt{\frac{\mathrm{F}_1(\mathbf{x})}{\mathrm{G}(\mathbf{x})}} - \frac{\mathrm{F}_1(\mathbf{x})}{\mathrm{G}(\mathbf{x})} sin\left(10\pi \mathrm{F}_1(\mathbf{x})\right)$$

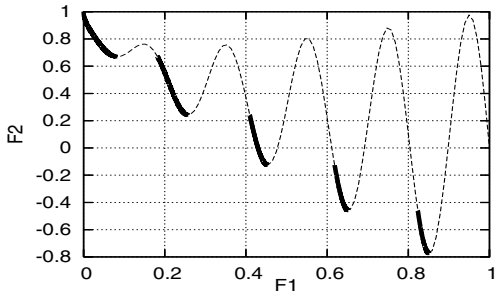


Fig. 3. *ZDT*3: The (analytically computed) Pareto front (continuous line) is shown over the dashed line which corresponds to $F_2 = F_2(F_1)$

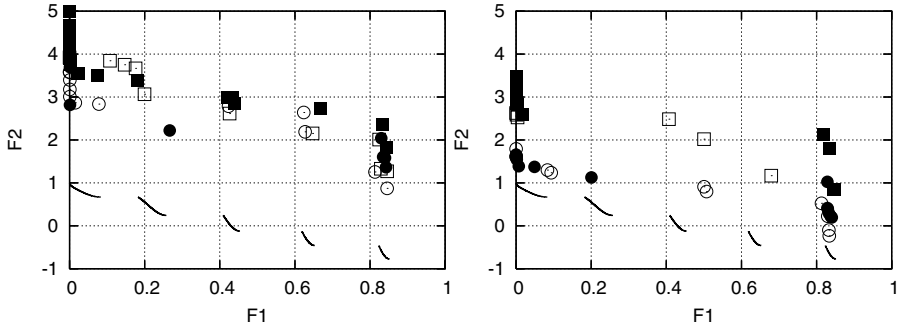


Fig. 4. *ZDT*3: Comparison of fronts of non–dominated solutions computed after 400 (left) and 800 (right) exact evaluations. Results based on the proposed *MAMA* are marked with • (RNG1) and ○ (RNG2) and those obtained using a *MAMA* which considers all metamodels as dependable are marked with ■ (RNG1) and □ (RNG2).

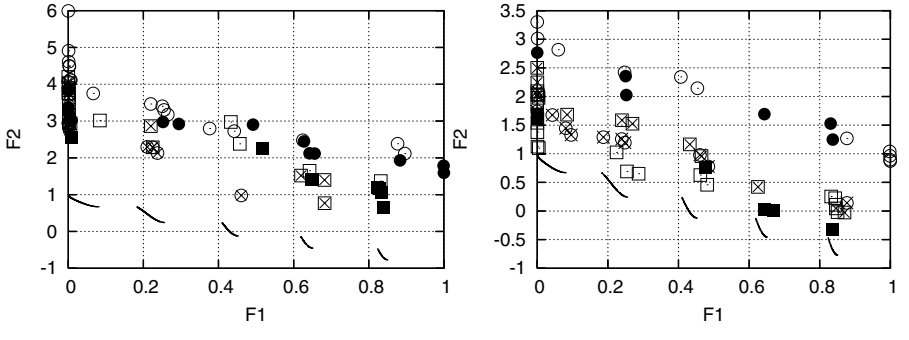


Fig. 5. *ZDT*3: Comparison of fronts of non–dominated solutions computed after 400 (left) and 800 (right) exact evaluations. Results obtained using the proposed *MAMA* with pure Lamarckian learning are marked with • (RNG1), ○ (RNG2) and ⊗ (RNG3). Results obtained using *MAMA* with combined Lamarckian–Baldwinian learning (as explained in the text) are marked with ■ (RNG1),□ (RNG2) and ⊠ (RNG3).

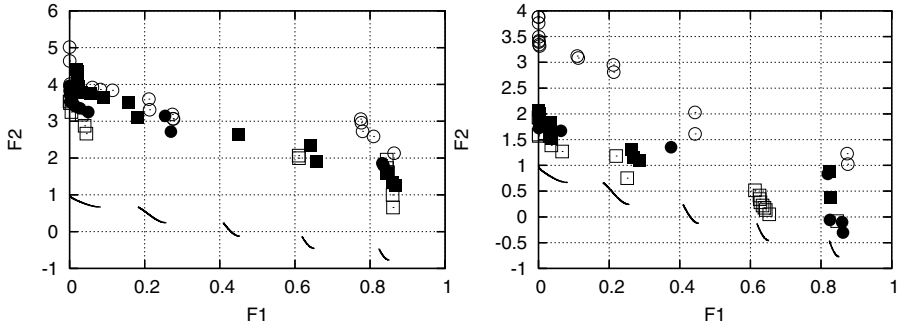


Fig. 6. *ZDT*3: Comparison of fronts of non–dominated solutions computed after 400 (left) and 800 (right) exact evaluations. Results obtained through local search for the improvement of the comparatively better objective are marked with • (RNG1) and ○ (RNG2). Results based on a randomly selected objective for each local search are marked with ■ (RNG1) and □ (RNG2).

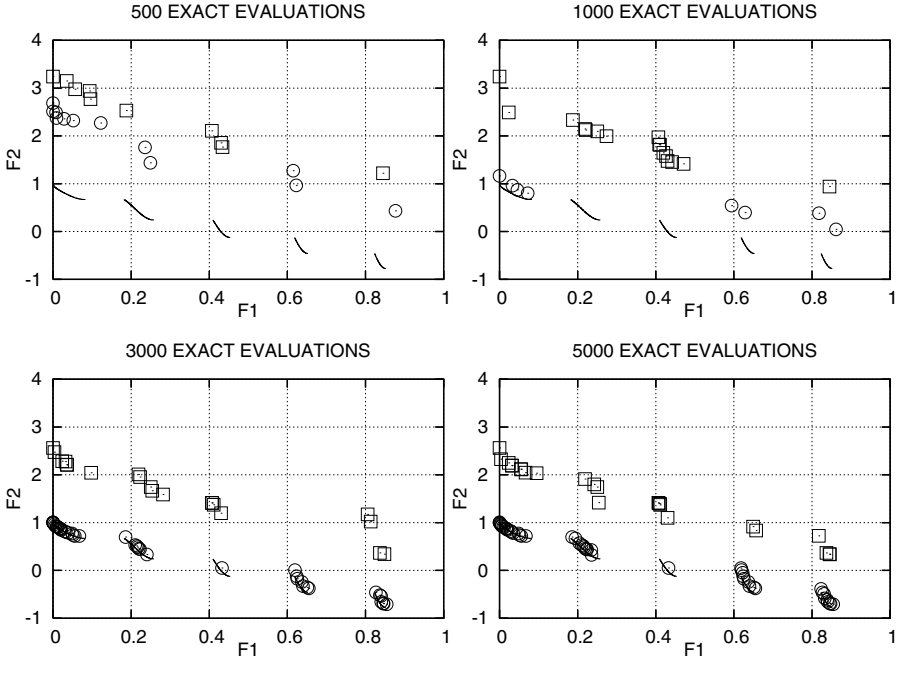


Fig. 7. *ZDT*3: Convergence history for RNG1. The non–dominated fronts computed after 500, 1000, 3000 and 5000 exact evaluations are presented for *MAEA* (□) and *MAMA* (○).

Although the computational cost of a single evaluation is practically negligible, *ZDT*3 is an appealing test case with a discontinuous optimal Pareto front, fig.3. Also, the low CPU cost per optimization allows running the case several times (with different pseudo–random number generator seed states, RNG) to comprehend the "average" behavior of the proposed method.

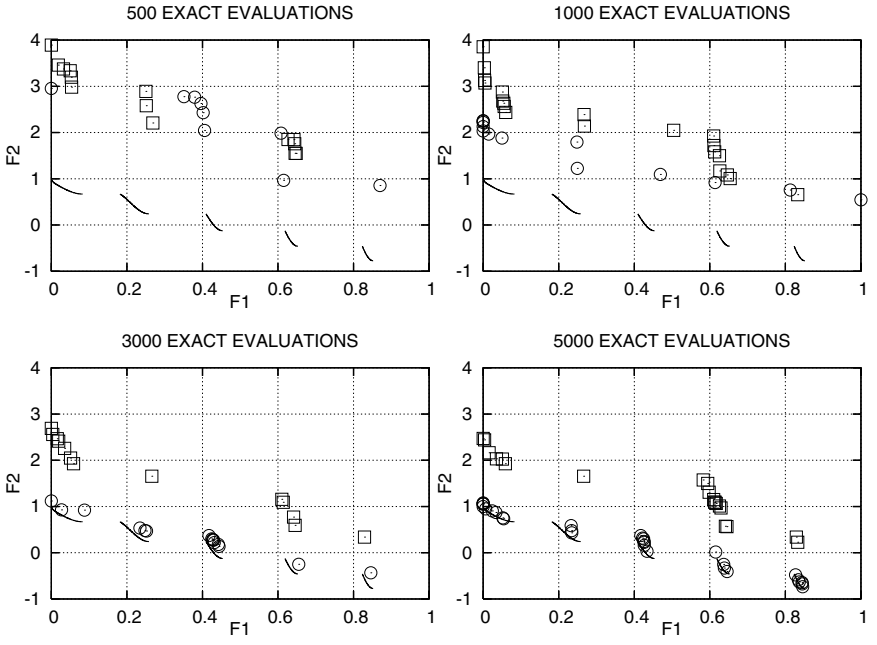


Fig. 8. *ZDT*3: Convergence history for RNG2. The non–dominated fronts computed after 500, 1000, 3000 and 5000 exact evaluations are presented for *MAEA* (□) and *MAMA* (○).

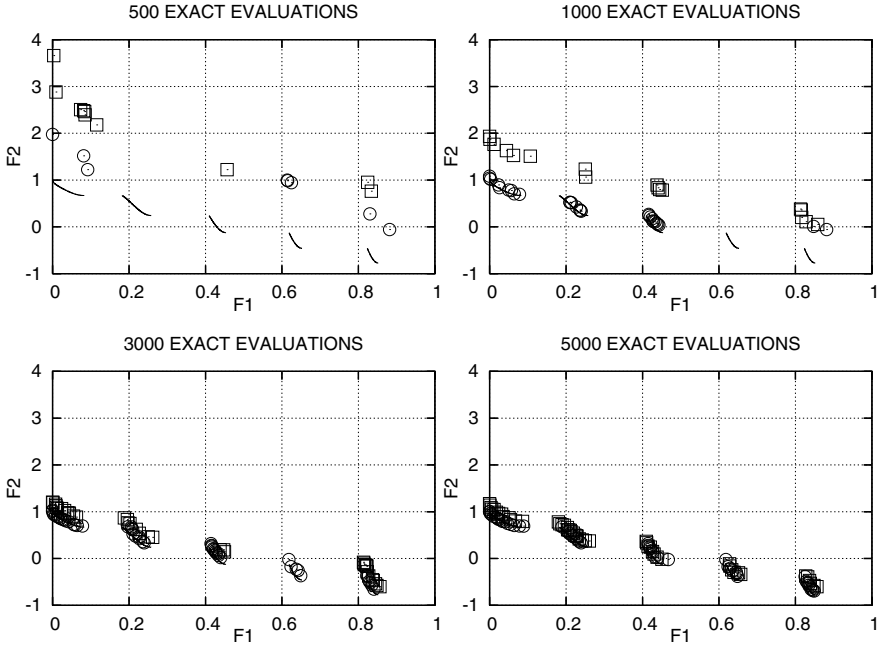


Fig. 9. *ZDT*3: Convergence history for RNG3. The non–dominated fronts computed after 500, 1000, 3000 and 5000 exact evaluations are presented for *MAEA* (□) and *MAMA* (○).

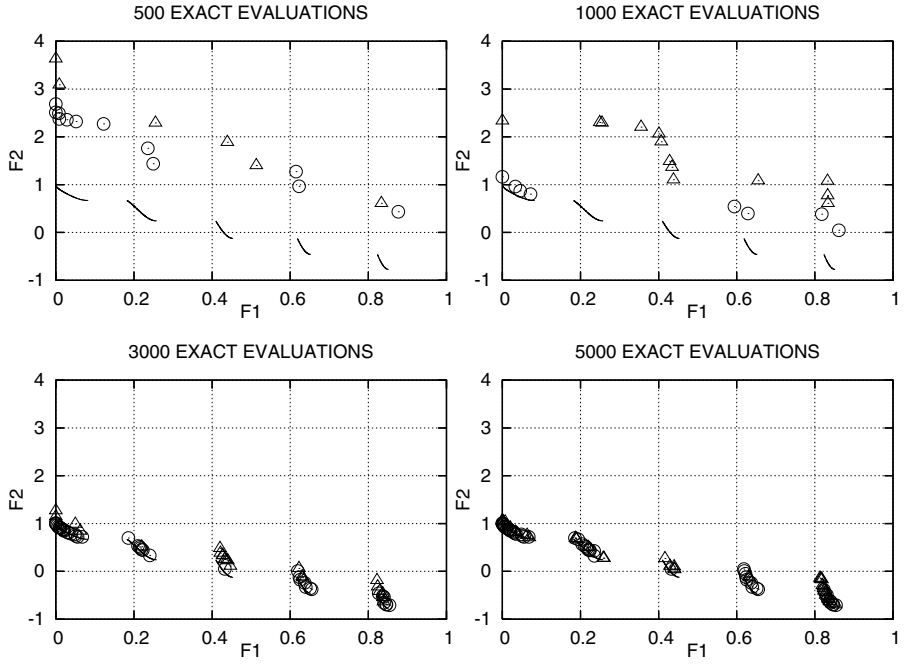


Fig. 10. *ZDT*3: Convergence history for RNG1. The non–dominated fronts computed after 500, 1000, 3000 and 5000 exact evaluations are presented for *MAMA* (○) and *MA* (△).

The first study investigates the importance of distinguishing dependable from questionable metamodels, a decision which, in any case, is a by–product of the algorithm used to select training patterns. For this purpose, a *MAMA* variant which labels all new individuals as dependable (by overcoming the ρ_2–based criterion or, practically, by letting ρ_2 take on an "infinite" value) is compared with the standard *MAMA* based on a reasonable value for ρ_2). Comparisons are shown in fig.4 for two different RNG seed states. Even during the first generations (after 400 and 800 exact evaluations) the superiority of the *MAMA* that handles dependable and questionable metamodels differently over the *MAMA* that handles all metamodels as dependable is clear.

The second investigation is concerned with the behavior of the proposed *MAMA* combined with either the Lamarckian or Baldwinian learning process. The outcome of this investigation is shown in fig. 5, for three runs per case. The comparison is made between a *MAMA* that uses pure Lamarckian learning and one which, if a better performing individual is found during the local search, this is inserted into the archival set $\mathscr{S}_{a,g}$ (compatible phenotype and genotype) without however changing the genotypic representation in the offspring population $\mathscr{S}_{\lambda,g}$. Although this is referred to as Baldwinian learning, it is in fact a mixed Lamarckian (for the archival population) and Baldwinian (for the offspring population) scheme. Definite conclusions cannot be drawn but it seems that the pure Lamarckian learning performs slightly better at the 800 evaluation plot.

The third investigation is concerned with the selection of the objective to be minimized during the local search. A scheme that selects the comparatively better

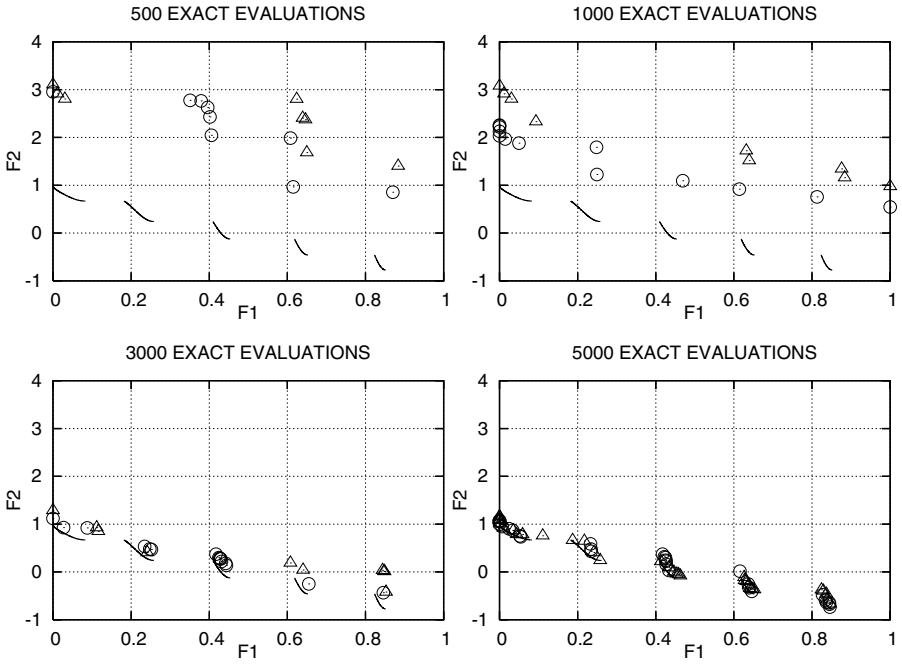


Fig. 11. *ZDT*3: Convergence history for RNG2. The non–dominated fronts computed after 500, 1000, 3000 and 5000 exact evaluations are presented for *MAMA* (○) and *MA* (△).

objective (depending on its relative location on the front) and another one which selects the objective to be refined for each new individual at random are compared, fig.6. The random selection seems to outperform the rest, though this conclusion could hardly be generalized.

A final comparison is made firstly between the proposed *MAMA* (with the already explained dual role of the *RBF* networks during both the global and local search) and a *MAEA* (in which the *RBF* networks are used only for the *IPE* of each population members) and, secondly, between the *MAMA* and a "typical" *MA*; for the latter, the local search is based on the same *RBF* networks. In the sake of fairness, all three of them are based on the same (μ,λ) *EA*, as basic search tool. The first comparison is shown in figs. 7, 8 and 9 whereas the second in figs. 10, 11 and 12. Each comparison was repeated three times, for three different RNG seed states. The first comparison leads to the conclusion that the proposed *MAMA* is much more efficient than the *MAEA*, and this is a good indication of the advantages of local search.

By considering the fronts of non–dominated solutions obtained at the cost of 5000 exact evaluations, we conclude that *MAMA* and *MA* outperform *MAEA*; the difference in performance is clear for RNG1 and RNG2 and less clear for RNG3.

In addition, for each RNG seed state, the three fronts of non–dominated solutions computed after 5000 exact evaluations are post–processed and the front of

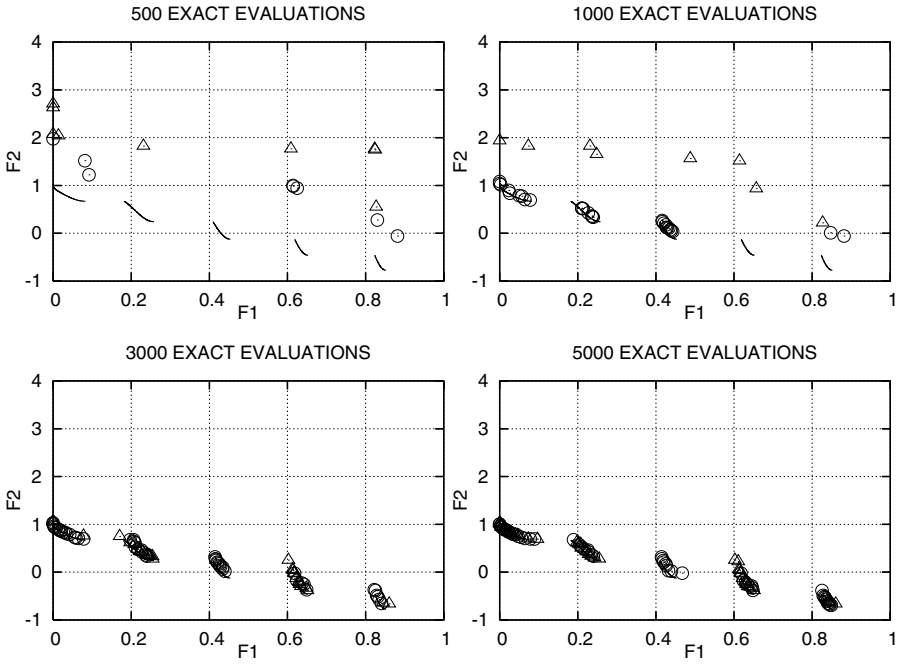


Fig. 12. *ZDT*3: Convergence history for RNG3. The non–dominated fronts computed after 500, 1000, 3000 and 5000 exact evaluations are presented for *MAMA* (○) and *MA* (△).

non–dominated solutions among them is derived. This is shown in fig.13, separately for each RNG seed state. In all cases, the final fronts consist almost exclusively of solutions computed using local search (either *MA* or *MAMA*). Between *MAMA* and *MA* there are no visible differences. Thus, to make a comparison one may count the number of non–dominated solutions on the front each method produces. This is tabulated below:

	MAMA	*MA*	*MAEA*
RNG1	46	16	0
RNG2	24	17	0
RNG3	38	30	5

and shows the superiority of *MAMA* with respect to *MA*. Some interesting statistics of the three runs (RNG1, RNG2, RNG3) of the *MAMA* algorithm are shown in fig.14. Since the Pareto front consists of five distinct parts (sub–fronts), which should all be populated by a well performing method, fig.14 (top) shows the number of elite members (after 5000 exact evaluations) *MAMA* and *MAEA*produce over the five parts of the front. From fig.14, it is clear that *MAMA* produces more elites on all these parts, which are also of better quality. This can be proved by examining the overall non–dominated front shown in fig.13. In addition, fig.14 (bottom) presents statistics on the use of local search in *MAMA*. As explained in its caption, the majority of solutions resulting from

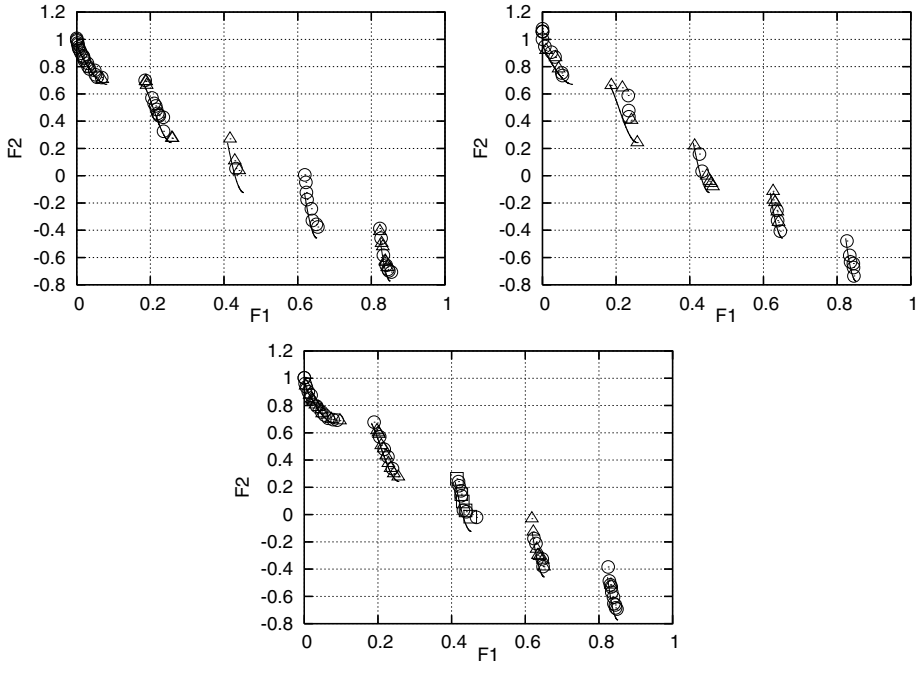


Fig. 13. *ZDT*3: The overall non–dominated members of the three non–dominated fronts computed using *MAEA* (□), *MAMA* (○) and *MA* (△), at the cost of 5000 exact evaluations

local search improve some of the objectives and produce higher cost values for the rest of them. However, there are more than 10% of these solutions that have been improved with respect to both objectives.

4.2 Design of a Combined Cycle Power Plant

The second application is concerned with the design of an optimal dual–pressure heat recovery steam generator (*HRSG*) along with the gas (*GT*) and steam (*ST*) turbines of a combined cycle power plant, as shown in fig.15. This problem involves 13 design variables (heat exchange areas, operating pressures and temperatures for the working fluids, the gas turbine operating point) with three objectives: maximum power plant efficiency, maximum produced power at the design point and minimum investment cost. A more detailed description of the case is omitted in the interest of space and since practical details of this engineering problem are beyond the scope of this chapter; the interested reader should refer to [39].

Figs.16 and 17 present the outcome of optimizations carried out using *MAMA*, *MAEA* and the *MA* which use the metamodel only to support local search and not for the inexact pre–evaluation of the population members. Fig.16 illustrates the computed Pareto fronts using *MAMA* and *MA* at 1000 exact evaluations (left) or using *MAMA* and *MAEA* at 10000 exact evaluations (right); the latter is considered to be a converged algorithm

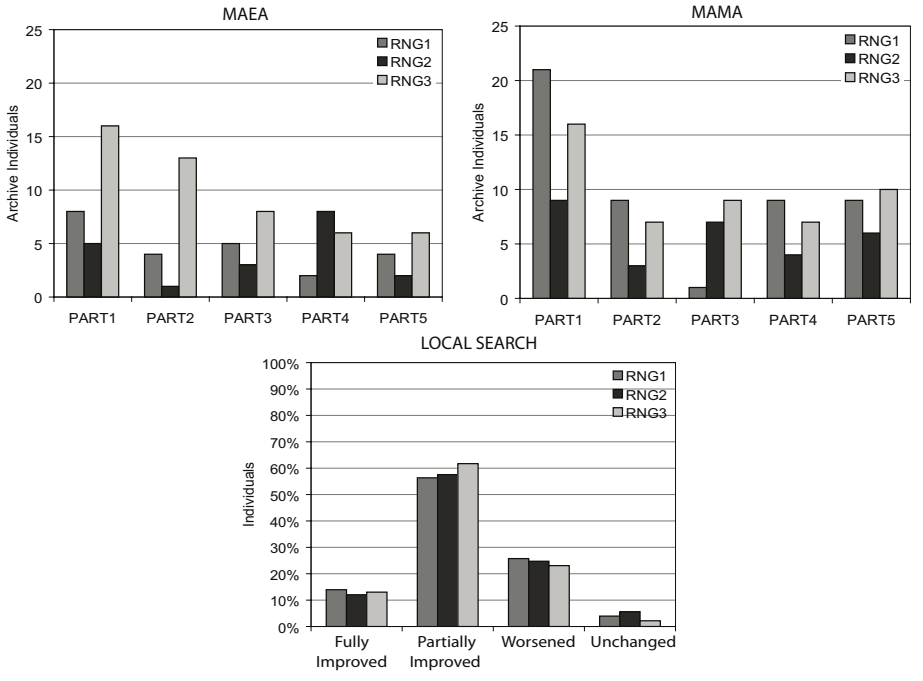


Fig. 14. *ZDT*3: Top–left: Number of elite members (at 5000 exact evaluations) per part of the discontinuous Pareto front, computed using *MAEA*. Top–right: Number of elite members (at the same number of exact evaluations) per part, computed using *MAMA*. Bottom–left: Statistics concerning the outcome of local search compared to the starting point. *Fully Improved* means that the outcome of local search dominates the starting point. The opposite stands for solutions marked as *Worsened*. *Partially improved* solutions are those which worsen at least one objective and improve at least another one. Finally, we consider as *Unchanged* those individuals which return to the starting point at the end of the local search.

from the engineering point of view whereas the former gives an indication of the algorithm's behavior at the early stages of the optimization. For the purpose of comparison, fig.16 presents only the members of the fronts of non–dominated solutions which dominate each other. By doing so, one may easily see that *MAMA* outperforms both *MA* and *MAEA*, at the early or late stages of the optimization since both 3D fronts are almost exclusively populated by solutions computed using *MAMA*. Since this figure hides details of the performance of the three algorithms tested, fig.17 aims at shedding more light into them. Using two RNG seed states (RNG1 and RNG2) for each method (six computations), the extent of each front of non–dominated solutions in the objective space after 1000 exact evaluations is presented. It is concluded that *MAMA* provides the fronts with maximum coverage. Finally, in fig.17, some statistics of the *MAMA*–based computation are also shown. From this figure, it is clear that the majority of local search actions leads to partially improved solutions (in the Pareto sense) and only about 10% of them yield improvements in all three objectives.

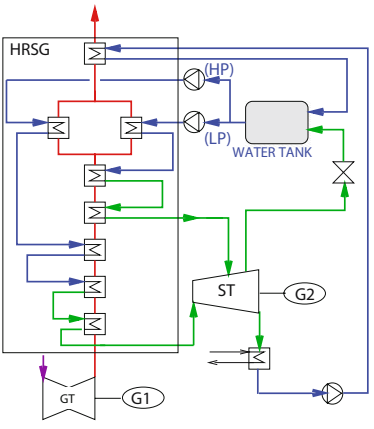


Fig. 15. Design of a combined cycle power plant: Plant configuration with a dual pressure *HRSG* (LP: Low–pressure, HP: High–pressure, G1–G2: generators). Although a single *GT* unit is shown (here, 120*MW*), in the optimal configuration this may consist of more than one units.

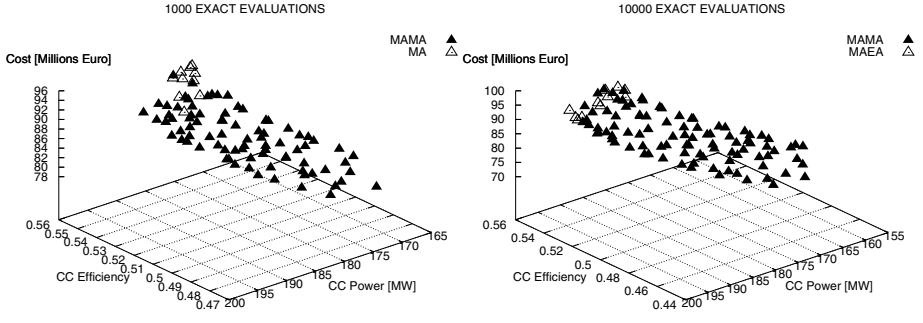


Fig. 16. Design of a combined cycle power plant: Left: The overall non–dominated solutions of the archival sets of *MAMA* and *MA*, computed after 1000 exact evaluations. Right: same plot for *MAMA* and *MAEA*, after 10000 exact evaluations.

4.3 Multi–point Design of a Compressor Cascade Airfoil

In the third example, the design–optimization of the airfoil of a compressor cascade at design and off–design flow conditions is presented. The optimization problem is defined as a two–objective one, where the first objective is to minimize the total pressure loss coefficient $F_1 = \omega = \frac{p_{tin} - p_{tout}}{p_{tin} - p_{in}}$ at the design flow conditions (Reynolds number based on the chord length $Re_c = 6 \cdot 10^5$, inlet Mach number $M_{in} = 0.5$ and inlet flow angle $\alpha_{in} = 53^o$) and the second one is to minimize the deviation of ω at off-design conditions ($\alpha_{in,1} = 50^o$, $\alpha_{in,2} = 56^o$). The second objective is cast in the form of a function $F_2 = \omega_{\alpha_{in,1}} + \omega_{\alpha_{in,2}} - 2\omega$. Here, p is the static pressure, p_t the stagnation pressure and indices *in* and *out* correspond to the cascade inlet and outlet, respectively. The exact evaluation tool is an integral boundary layer method coupled with an external flow solver (MISES,

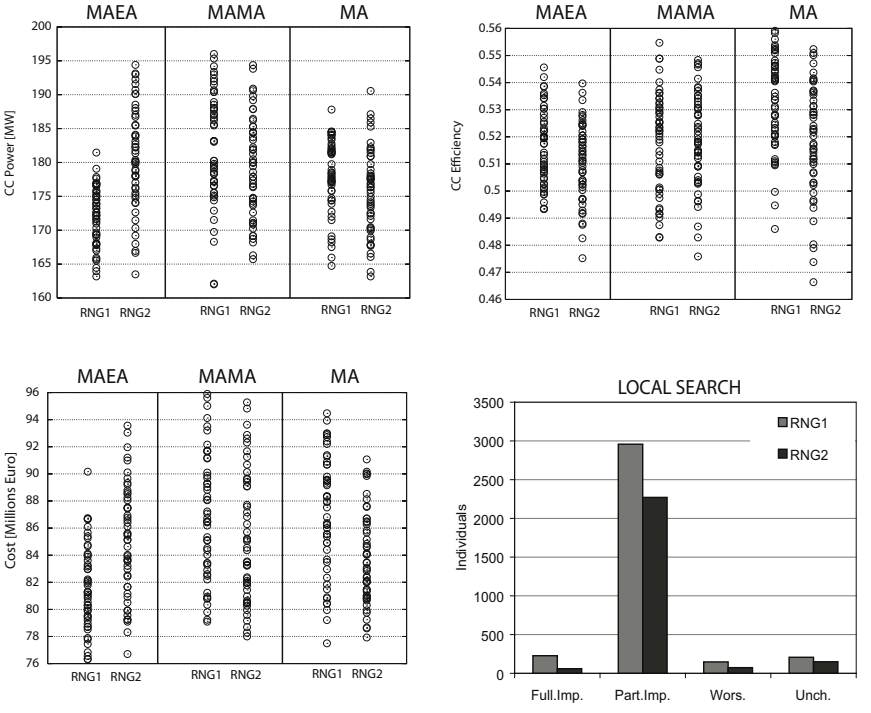


Fig. 17. Design of a combined cycle power plant: Extent (in the objectives space) of the fronts of non–dominated solutions, computed using *MAEA*, *MAMA* and *MA* after 1000 exact evaluations, for RNG1 and RNG2 (top and bottom–left). Local search performance statistics (as in fig. 14; bottom–right).

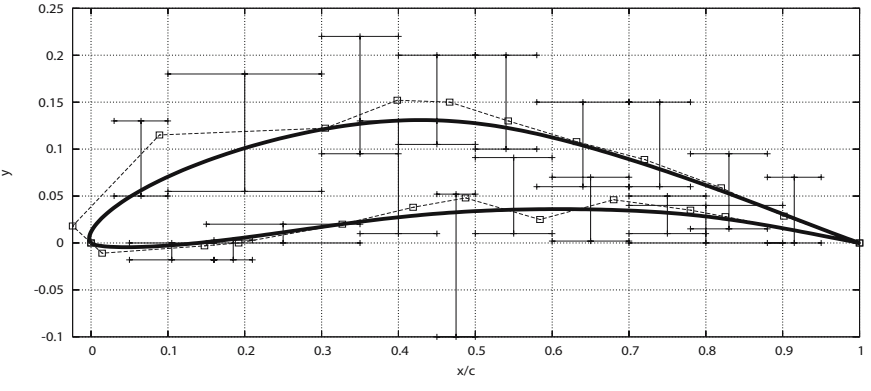


Fig. 18. Multipoint design of a compressor cascade airfoil: Airfoil contour parameterization and design variables' (Bézier control points') bounds

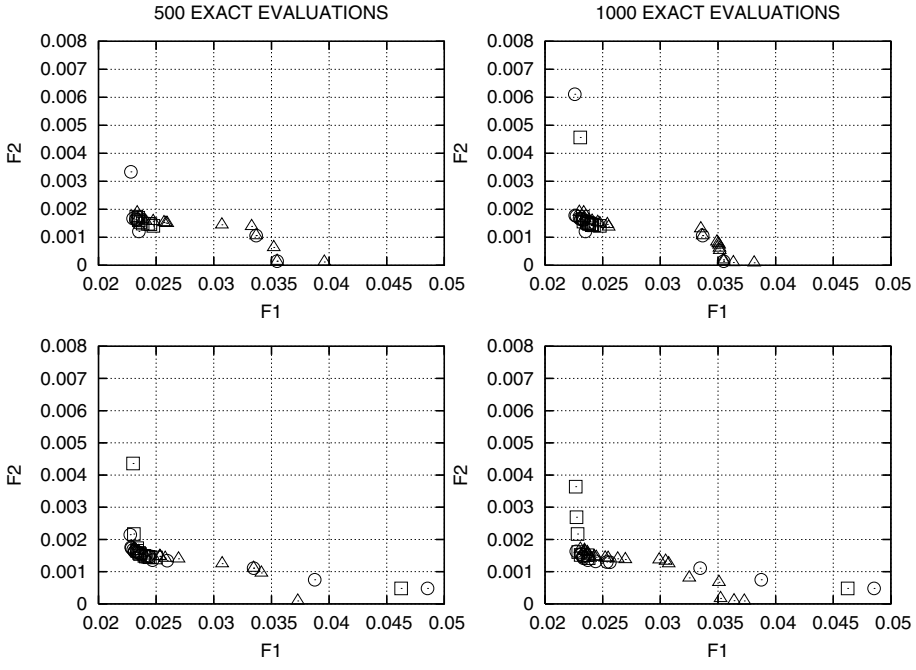


Fig. 19. Multipoint design of a compressor cascade airfoil: Convergence history with RNG1 (top) and RNG2 (bottom). The non–dominated fronts computed after 500 (left) and 1000 (right) exact evaluations are presented for *MAEA* (□), *MAMA* (○) and *MA* (△).

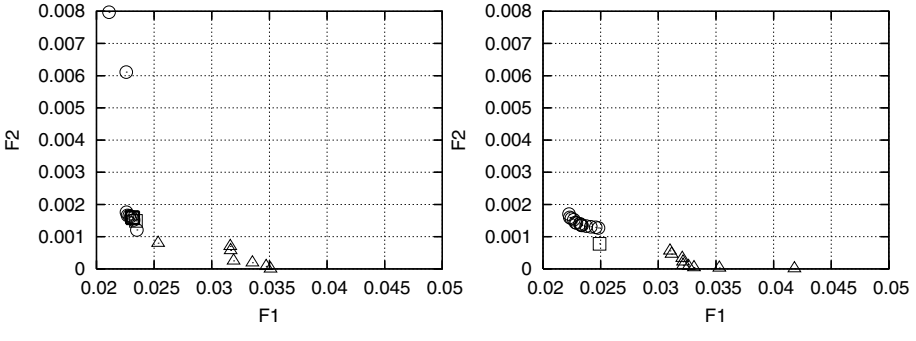


Fig. 20. Multipoint design of a compressor cascade airfoil: The overall non–dominated members of the two non–dominated fronts (with RNG1 and RNG2) computed using *MAEA* (□), *MAMA* (○) and *MA* (△), at the cost of 3000 exact evaluations

[18, 15]). The airfoil contour is parameterized using two separate Bézier polynomial curves for the suction and pressure side, with 12 control points each. The endpoints of each Bézier curve are fixed and the remaining control points are allowed to move within the bounds illustrated in fig.18. The same figure also shows an existing airfoil

with $\omega = 0.029$ at the design point. Three geometrical constraints related to the airfoil thickness (th) at certain chordwise locations, namely

$$th(0.30c) \geq 0.075c, \;\; th(0.70c) \geq 0.04c, \;\; th(0.90c) \geq 0.01c$$

(c is the chord length) and a flow–related constraint

$$\alpha_{in} - \alpha_{out} \geq 25^o$$

where $\alpha_{in} - \alpha_{out}$ is the cascade flow turning, are imposed.

Fig.19 summarizes the outcome of six computations (three methods: *MAEA*, *MAMA* and *MA*, for RNG1 and RNG2) after 500 (left) and 1000 (right) exact evaluations. Even during the early phase of the optimization, the "centre" of the front is dominated by solutions obtained using *MAMA*. It is also interesting to note that other parts of the front are dominated by the *MA*: it is clear that both *MAMA* and *MA* benefit from local search. This can be seen in fig.20 where the overall non–dominated members of the non–dominated fronts computed by the three methods (at the cost of 3000 exact evaluations) are shown.

4.4 Summary–Conclusions

The purpose of this chapter was to present a Metamodel–Assisted Memetic Algorithm, using on–line trained local metamodels supporting both the inexact pre–evaluation of evolving populations and the local search. The use of metamodels (here, *RBF* networks) to screen out non–promising individuals in each generation, so as to avoid evaluating them with the exact and costly evaluation tool, is currently a well established technique; it considerably reduces the CPU cost of evolutionary computations and makes them competitive for engineering applications with computationally expensive evaluation tools. Here, the same metamodels also support local search employed at selected population members for the purpose of refinement. The proposed *MAMA* was designed for use in *MOO* problems, cooperating with a (μ, λ) *EA*. Emphasis was put to the selection of the most appropriate set of patterns for the training of the metamodel and the classification of trained networks as dependable or questionable (depending on the density of the available information around each new individual) or the identification of outliers. Experimentation on three test problems (mathematical test case, energy system optimization and compressor airfoil design) led to the conclusion that the proposed *MAMA* outperforms *MAEAs* and standard surrogate–assisted *MAs*.

References

1. Dawkins, R.: The selfish gene. Oxford University Press, Oxford (1976)
2. Goldberg, D.E.: Genetic algorithms in search, optimization & machine learning. Addison-Wesley, Reading (1989)
3. Haykin, S.: Neural networks: A comprehensive foundation. Prentice Hall, New Jersey (1999)
4. Knuth, D.: Fundamental algorithms. Addison-Wesley, Reading (1997)
5. Moscato, P.: Memetic algorithms: A short introduction. McGraw-Hill Company, New York (1999)

6. Myers, R.H., Montgomery, D.C.: Response surface methodology: Process and product optimization using designed experiments. John Wiley & Sons, New York (2002)
7. Nocedal, J., Wright, S.: Numerical optimization. Springer, New York (1999)
8. Tikhonov, A., Arsénine, V.: Méthodes de résolution de problèmes mal posés. Editions MIR, Moscou (1976)
9. Tikhonov, A.N., Goncharsky, A.V., Stepanov, V.V., Yagola, A.G.: Numerical methods for the solution of ill-posed problems. Kluwer Academic Publishers, Dordrecht (1995)
10. Hart, W.E.: Adaptive global optimization with local search, Ph.D. Dissertation 1994. University of California, San Diego (1994)
11. Krasnogor, N.: Studies on the theory and design space of memetic algorithms. Ph.D. Dissertation 2002, University of the West of England, Bristol (2002)
12. Land, M.W.S.: Evolutionary algorithms with local search for combinatorial optimization. Ph.D. Dissertation 1991, University of California, San Diego (1998)
13. Bishop, C.: Improving the generalisation properties of radial basis neural networks. Neural Computation 3(4), 579–588 (1991)
14. Broomhead, D.S., Lowe, D.: Multivariable functional interpolation and adaptive networks. Complex Systems 2(3), 312–355 (1988)
15. Drela, M., Giles, M.B.: Viscous-inviscid analysis of transonic and low Reynolds number airfoils. AIAA Journal 25(10), 1347–1355 (1987)
16. Emmerich, M., Giannakoglou, K., Naujoks, B.: Single and multiobjective evolutionary optimization assisted by Gaussian random field metamodels. IEEE Trans. on Evolutionary Computation 10, 421–439 (2006)
17. Fritzke, B.: Fast learning with incremental RBF networks. Neural Processing Letters 1, 2–5 (1994)
18. Giles, M.B., Drela, M.: A two–dimensional transonic aerodynamic design method. AIAA Journal 25(9), 1199–1206 (1987)
19. Giannakoglou, K.: Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. International Review Journal Progress in Aerospace Sciences 38, 43–76 (2002)
20. Grierson, D.E., Pak, W.H.: Optimal sizing, geometrical and topological design using a genetic algorithm. Structural and Multidisciplinary Optimization 6(3), 151–159 (1993)
21. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Trans. on Evolutionary Computation 7, 204–223 (2003)
22. Jaszkiewicz, A.: Genetic local search for multiobjective combinatorial optimization. European Journal of Operational Research 137(1), 50–71 (2002)
23. 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)
24. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing 9, 3–12 (2003)
25. Karakasis, M., Koubogiannis, D., Giannakoglou, K.: Hierarchical distributed evolutionary algorithms in shape optimization. International Journal for Numerical Methods in Fluids 53, 455–469 (2007)
26. Karakasis, M., Giannakoglou, K.: On the use of metamodel-assisted, multiobjective evolutionary algorithms. Engineering Optimization 38(8), 941–957 (2006)
27. Karakasis, M., Giannakoglou, K.: Inexact information aided, low-cost, distributed genetic algorithms for aerodynamic shape optimization. International Journal for Numerical Methods in Fluids 43, 1149–1166 (2003)
28. Liang, K., Yao, X., Newton, C.: Evolutionary search of approximated N-dimensional landscapes. International Journal of Knowledge-Based Intelligent Engineering Systems 4, 172–183 (2000)

29. Moody, J., Darken, C.J.: Fast learning in networks of locally-tuned processing units. Neural Computation 1, 281–294 (1989)
30. Ong, Y.S., Zhou, Z.Z., Nair, P.B., Keane, A.J., Lum, K.Y.: Combining global and local surrogate models to accelerate evolutionary optimization. IEEE Trans. on Systems, Man and Cybernetics, Part C: Applications and Reviews 37, 66–76 (2007)
31. Ong, Y.S., Zhou, Z., Lim, M.H., Lee, B.S.: Memetic algorithm using multi–surrogates for computational expensive optimization problems. Journal of Soft Computing 11(10), 957–971 (2007)
32. Park, J., Sandberg, I.W.: Universal approximation using radial basis function networks. Neural Computation 3(2), 246–257 (1991)
33. Park, J., Sandberg, I.W.: Universal approximation using radial basis function networks. Neural Computation 3, 246–257 (1991)
34. Poggio, T., Girosi, F.: Networks for approximation and learning. Proceedings of the IEEE 78(9), 1481–1497 (1990)
35. 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)
36. Regis, R.G., Shoemaker, C.A.: Local function approximation in evolutionary algorithms for the optimization of costly functions. IEEE Trans. on Evolutionary Computation 8(5), 490–505 (2004)
37. Wang, G.G., Shan, S.: Review of metamodelling techniques in support of engineering design optimization. Trans. ASME, Journal of Mechanical Design 129(4), 370–380 (2006)
38. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. Evolutionary Computation 8, 173–195 (2000)
39. Bonataki, E., Georgoulis, L., Georgopoulou, C., Giannakoglou, K.: Optimal design of combined cycle power plants based on gas turbine performance data. In: ERCOFTAC Design Optimization: Methods & Applications, Athens (March 2004)
40. Dunne, R.A.: Smoothing the output of radial basis function networks. In: 5th Australian Conference on Neural Networks, Sydney (1994)
41. El-Beltagy, M.A., Nair, P.B., Keane, A.J.: Metamodeling techniques for evolutionary optimization of computationally expensive problems: Promises and limitations. In: GECCO 1999, Genetic and Evolutionary Computation Conference, Orlando (July 1999)
42. Federici, D.: Combining genes and memes to speed up evolution. In: CEC 2003, Congress on Evolutionary Computation, Canberra (December 2003)
43. Giannakoglou, K.: Designing turbomachinery blades using evolutionary methods. In: ASME Paper 99-GT-181, ASME Turbo Expo 1999, Indianapolis (June 1999)
44. Hacker, K.A.: Efficient global optimization using hybrid genetic algorithms. In: AIAA Paper 2002-5429, 9th AIAA//ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, Georgia (September 2002)
45. Ishibuchi, H., Murata, T.: Multiobjective genetic local search algorithm. In: CEC 1996, Congress on Evolutionary Computation, Nagoya, Japan (May 1996)
46. Nair, P.B., Keane, A.J.: Combining approximation concepts with genetic algorithm–based structural optimization procedures. In: AIAA Paper 1998-1912, 39th AIAA/ASMEASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Long Beach, CA (April 1998)
47. Knowles, J.D., Come, D.W.: The Pareto Archived Evolution Strategy: A new baseline algorithm for multiobjective optimisation. In: CEC 1999, Congress on Evolutionary Computation, Piscataway, NJ (July 1999)
48. Knowles, J.D., Corne, D.W.: M-PAES: A memetic algorithm for multiobjective optimization. In: CEC 1996, Congress on Evolutionary Computation, San Diego, CA (July 2000)

49. Knowles, J.D., Corne, D.W.: A comparison of diverse approaches to memetic multiobjective combinatorial optimization. In: GECCO 2000, Genetic and Evolutionary Computation Conference, Las Vegas, NV (July 2000)
50. Knowles, J.D., Corne, D.W.: A comparative assessment of memetic, evolutionary, and constructive algorithms for the multiobjective d-MST problem. In: GECCO 2001, Genetic and Evolutionary Computation Conference, San Francisco, CA (July 2001)
51. Ku, K., Mak, M.: Exploring the effects of Lamarckian and Baldwinian learning in evolving recurrent neural networks. In: IEEE International Conference on Evolutionary Computation, Indianapolis (April 1997)
52. Liang, K., Yao, X., Newton, C.: Combining landscape approximation and local search in global optimization. In: CEC 1999, Congress on Evolutionary Computation, Washington (July 1999)
53. Papila, N., Shyy, W., Fitz-Coy, N., Haftka, R.T.: Assessment of neural net and polynomial-based techniques for aerodynamic applications. In: AIAA Paper 1999-3167, 17th AIAA Applied Aerodynamics Conference, Norfolk, VA (1999)
54. Ratle, A.: Optimal sampling strategies for learning a fitness model. In: CEC 1999, Congress on Evolutionary Computation, Piscataway, NJ (July 1999)
55. Benoudjit, N., Archambeau, C., Lendasse, A., Lee, J., Verleysen, M.: Width optimization of the Gaussian kernels in radial basis function networks. In: ESANN Paper, 10th European Symposium on Artificial Neural Networks, Bruges (2002)
56. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. In: C3P 826, Pasadena, CA (1989)
57. Orr, M.J.L.: Regularised centre recruitment in radial basis function networks. In: Research Paper 59, Centre For Cognitive Science, University of Edinburgh (1993)
58. Zitzler, E., Laumans, M., Thiele, L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm for multiobjective optimization. TIK–Report 103, Zurich (2001)

A Convergence Acceleration Technique for Multiobjective Optimisation

Salem F. Adra[1], Ian Griffin[2], and Peter J. Fleming[3]

[1] Automatic Control and Systems Engineering, The University of Sheffield, UK
s.adra@sheffield.ac.uk
[2] Automatic Control and Systems Engineering, The University of Sheffield, UK
i.griffin@sheffield.ac.uk
[3] Automatic Control and Systems Engineering, The University of Sheffield, UK
p.fleming@sheffield.ac.uk

A memetic algorithm which addresses the requirement for solutions convergence towards the Pareto front of a multiobjective optimisation problem is discussed. The memetic algorithm is designed by incorporating a Convergence Accelerator Operator (CAO) in existing algorithms for evolutionary multiobjective optimisation. The discussed convergence accelerator works by suggesting improved solutions in objective space and using neural network mapping schemes to predict the corresponding solution points in decision variable space. Two leading multiobjective evolutionary algorithms have been hybridised through introduction of the CAO and tested on a variety of recognised test problems. These test problems consisted of convex, concave and discontinuous test functions, with numbers of objectives ranging from two to eight. In all cases introduction of the CAO led to improved convergence for comparable numbers of function evaluations.

1 Introduction

Real-world problems commonly require the simultaneous consideration of multiple, competing performance measures. Without loss of generality, a multiobjective optimisation problem (MOP) can be formulated as a minimization of a function $Z(X)$, where $Z(X) = Z_1(X)...Z_n(X)$ is a vector of objective functions, n is the number of objectives to be optimised and X is a vector of decision variables. For multiobjective problems in which objectives are competing, no single optimal solution exists, rather a set of candidate solutions, known as the approximation set, is sought.

More specifically, the optimisation problem consists of finding the set of decision vectors that results in the best set of solutions in objective space. The ideal set of decision vectors will be characterised by the fact that no other solution offers better objective function values across all objectives. This optimal set of candidate solutions is said to be non-dominated and is known as the Pareto optimal set, from which the decision maker ultimately selects an acceptable solution. The associated objective vectors form the trade-off surface (or Pareto front) in objective space. Figure 1 shows an optimisation problem where 3 decision variables are optimised with respect to 2 competing

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 183–205.
springerlink.com

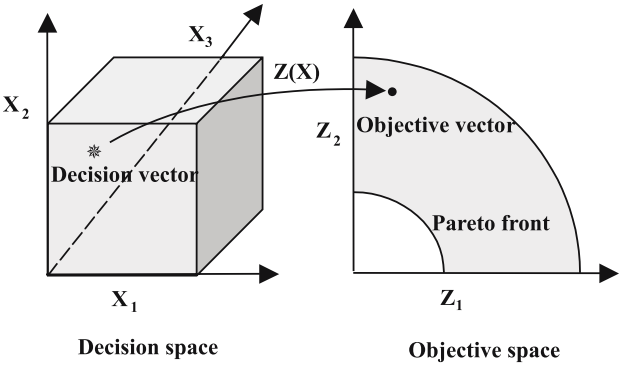


Fig. 1. The multiobjective problem domain

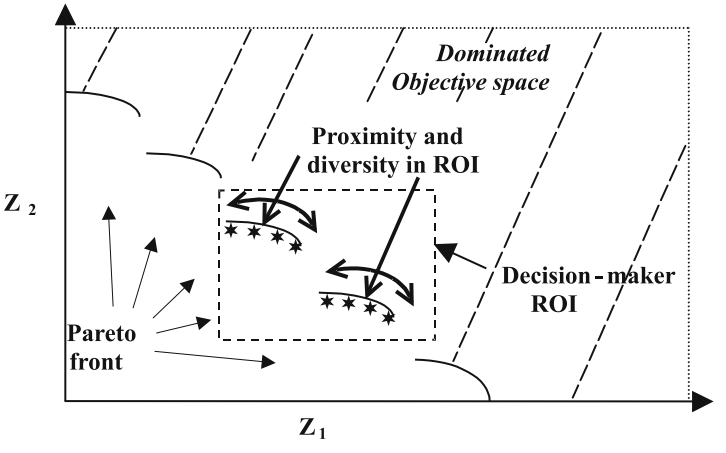


Fig. 2. A good set of solutions to a multiobjective optimisation problem in terms of proximity, diversity and relevance (i.e. location in ROI)

objectives, illustrating the mapping of a decision vector into objective space and showing the Pareto front for this idealised case.

The approximation set offered to the decision maker, is thus required to be as close as possible to the true Pareto front and well spread across objective space, presenting the decision maker with a well distributed set of solutions within the region(s) of interest (ROI) [1]. These two characteristics of an approximation set are termed proximity and diversity, respectively, and are illustrated in Figure 2.

To be of practical use, a multiobjective optimisation algorithm must produce an approximation set with acceptable proximity and diversity within acceptable computational resource. The performance of a multiobjective optimiser can then be determined by the proximity and diversity of the approximation sets produced from a given number of iterations over multiple runs of the algorithm [2]. In many application domains, calculating the true objective function may be computationally expensive. The use of

approximated models using Neural Networks (NN), metamodelling techniques, such as Kriging-based approximations, or response surface models [3], [4] provides low computational burden alternatives to full objective function evaluation [5], [6].

Evolutionary Algorithms (EAs) are stochastic, population-based, global search techniques well suited for solving MOPs. The population-based nature of EAs makes them well suited to addressing non-commensurate multiobjective problems as they simultaneously explore a family of points in the search space. The operators within these algorithms mimic Darwinian biological principles of stochastic selection followed by recombination and mutation [7] [8]. Starting either from a random population of candidate solutions or from a previously known set of solutions in decision variable space, EAs calculate the corresponding objective function values, assign them fitness scores reflecting their utility in the application domain and bias the search towards high-potential areas of the space by forcing the survival-of-the-fittest solutions. Despite their utility for solving MOPs, the use of EAs often result in a large number of objective function calculations which can be computationally expensive especially when the objective functions themselves are expensive to evaluate. Moreover, given the stochastic nature of its operators, an evolutionary algorithm offers no guarantee of finding optimal solutions within a single run. Through the variation operators operating in the decision variable space, new solutions are produced with the assumption that good parents are more likely to produce good offspring and hence should contribute more to the next generations. The addition of some straightforward determinism to such stochastic strategies is sought as a remedial measure which is believed to enhance the performance and the efficiency of EAs. This motivates further investigations and experimentations within a framework hybridizing EAs strength and structure with some innovative deterministic components; a framework commonly known as a Hybrid EA or Memetic Algorithm.

Memetic Algorithm is a concept first introduced in 1989 by [9]. The term Memetic has its roots in the word meme introduced by [10] and which denoted the unit of imitation in cultural transmission. Memetic algorithms, also called hybrid evolutionary algorithms, are increasingly thriving metaheursitics for solving multiobjective optimisation problems. The essential idea behind Memetic Algorithms is the hybridization of local search refinement techniques within a population-based strategy, such as EAs. Memetic Algorithms share most of EAs characteristics although they introduce a new improvement procedure based on local search. The main conceptual difference between EAs and memetic algorithms is the approach of the informations transmission. Whereas genetic information carried by genes is usually transmitted intact to the offspring (e.g. EAs), *memes* the base unit of memetic algorithms are typically adapted by the individual transmitting them. These hybrid algorithms were applied to a wide variety of problems such as image segmentation [11], multiobjective optimization of space allocation problems [12], radiotherapy treatment planning [13] and molecular geometry optimisation [14]. They have proved to be highly effective, outperforming similar approaches such as pure evolutionary algorithms in several application domains in terms of convergence towards Pareto optimal solutions.

As stated by [15] and re-illustrated by [16], for improving optimization results achieved by EAs one should: *'Hybridize where possible'*. The hybridization of local improvement operators among the evolutionary steps of an EA is essential to deal with

near optimal situations. This has been shown [16] in several application domains to bring improvements to the standard results achieved by stand alone genetic algorithms in terms of results quality and speed of convergence. The combination of global and local search is a strategy used by many successful global optimization approaches, and has in fact been recognized as a powerful algorithmic paradigm for evolutionary computing [17].

In this chapter, a convergence accelerator is described, which maps from objective space to decision variable space (in the reverse direction to a meta-modeling technique). This operator is a portable component that can be hybridized with any Multiobjective EA (MOEA). The purpose of this Convergence Accelerator Operator(CAO) is to enhance the performance of the host MOEA in terms of the proximity of the approximation set for a given number of objective function calculations without impeding the active diversification mechanisms of these search strategies. In this chapter, the CAO is hybridized with two widely used MOEAs, the Non-Dominated Sorting Genetic Algorithm (NSGA-II) [18] and Strength Pareto Evolutionary Algorithm (SPEA2) [19]. EAs operate in decision space and perform decision space to objective space mapping but tend to fail to exploit direct use of the objective space - a lost opportunity. In contrast to this, the CAO features an innovative direct search in objective space and then uses predictions to map from objective space to decision space; in this study this mapping is realised by an artificial neural network (NN). Performing local search in the objective space was first introduced in [20, 21] and later extended in [5]. In [20, 21, 5], the authors suggested training a NN to map in the reverse direction (i.e. objective vectors as inputs and decision vectors as outputs) and using it in a local search around the non-dominated solutions arising from the previous generation.

The chapter is organized as follows: In Section 2, the proposed CAO is introduced and described. Section 3 describes the test procedures used in the comparative testing of the standard and CAO-enhanced algorithms. Section 4 presents results of the tests described in Section 3, and concluding remarks are provided in Section 5.

2 The Convergence Acceleration Operator

2.1 Overview

The CAO is a 2-step process, which is illustrated in Figure 3.

When the CAO is launched, it starts by deterministically improving the best solutions achieved: these solutions are stored in the elitist population or the online archive of the host algorithm. This improvement takes place in objective space and produces an enhanced version of the archive. The CAO then uses a trained neural network mapping procedure to predict the corresponding decision vectors for the enhancements to the archive. A check of these new decision vectors is made, aimed at reflecting any out-of-bounds decision variables arising from the mapping back into their allowed domain. The true objective values corresponding to all of these new decision vectors are calculated. The enhanced and the original archive of solutions now compete to populate the new archive for the next generation, which will represent the pool from which solutions are selected and recombined. The two components of the CAO are described in detail in the following sections.

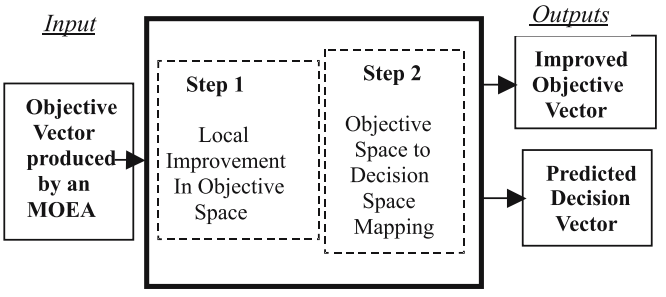


Fig. 3. The Convergence Acceleration Operator in Context

2.2 Local Improvement in Objective Space

The first CAO step is a deterministic local improvement procedure in the objective space. This is the component responsible for speeding up convergence, thereby reducing computational effort. It achieves this by steering objective values obtained by the MOEA towards an improved Pareto front. The objective space local improvement process is implemented for n objectives, and is illustrated in Figure 4 on a bi-objective problem *(n = 2)*. Note that a minimization problem is assumed, without any loss of generality.

In general, interior solutions, in terms of any specific objective *(solutions B, C and D in Figure 4)* will be improved in terms of all the performance measures by steering their objective values into a region of improved objective function values. The new improved values for the objectives are determined by linearly interpolating a new value for each objective, between its current value and the next best value(s) achieved for that objective within the population. This is described by $Z_{D'} = Z(x_D + h(x_D - x_C), y_D + h(y_D - y_E))$ where Z(x,y) represents a point in the bi-objective space, Z_D is the improved

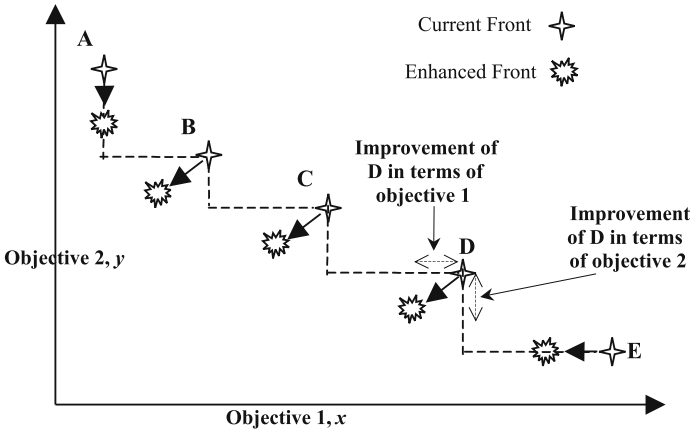


Fig. 4. Deterministic improvement of the trade-off surface in objective space

objective value and h is the interpolation step factor. This process is annotated for solution D in Figure 4. Compared to solution D, solution C has the next best value in terms of Objective 1 while solution E possesses the next best value in terms of Objective 2. The size of the step factor or objective space improvement in each dimension should be carefully chosen; ideally it should depend on the stage of the optimisation, the decision makers preferences, the regions of interests and the proximity of the population to the true Pareto front. A larger step factor is recommended for early generations of the optimisation, with its value gradually decreasing.

Boundary solutions in terms of a certain objective or axis of performance (points A and E in Figure 4) are improved in terms of the remaining objectives. In other words, solution A will be improved in the y-axis direction (Objective 2), thereby enhancing its overall quality by improving it in terms of Objective 2, and solution E will be improved in the x-axis direction (Objective 1), consequently improving its overall worth by enhancing it in terms of Objective 1.

2.3 Objective Space to Decision Space Mapping

The second component of the CAO consists of a neural network trained to map the new solutions thus generated in objective space by the first phase of the convergence accelerator back to the corresponding decision variable vectors.

Hybridizing a NN with an EA is very useful for approximating expensive objective functions. This is the meta-modeling principle [22]. By contrast, in this work, a NN is deployed in the CAO to map the proposed objective vectors back to their estimated decision variable vectors. This is achieved by training a NN, using exact objective vectors as inputs and their corresponding decision variable vectors as outputs, to approximate a mapping function from the objective space to the decision space. The training data is the exact data resulting from the objective function values derived within the cycle of a MOEA such as NSGA-II [18] or SPEA2 [19]. The ability to map objective vectors to decision variables will make it possible to search directly in objective space for desired combinations of objective values or to devise points of attractions to guide the search.

The design of the architecture of the NN (Multilayer Perceptron) was based on a trial-and-error set of experiments. The standard backpropagation algorithm [23] was used for training the NN.

Two possible approaches to training the NN component of the CAO hybridized with a MOEA are proposed: online and offline training modes. In this study, the online training mode is further elaborated and investigated. However, the interested reader is directed to [24] where the offline training mode is explored and investigated.

Neural Network Training Modes

- Online Training Mode:

The online mode consists of concurrently training and validating the NN during the execution of the MOEA. Many strategies for controlling the use of the CAO in this mode might be devised. In this mode, the CAO is a performance accelerator that can be launched upon the request of the decision maker (DM) during the execution of the optimisation process.

- Offline Training Mode:

An alternative use of the NN is to train it with data resulting from evaluations of the objective functions arising from a single run of a MOEA and then to incorporate it in subsequent runs of a MOEA when used in conjunction with the CAO. This training mode will produce a NN since trained on a richer data set. It can subsequently be hybridized with any optimiser attempting to solve the same problem. Thus, the CAO will benefit succeeding executions of the same or other optimisers solving the same problem by speeding up the search and has the potential to offer other benefits. The offline training mode restricts the usage of the CAO to specific applications and optimisation scenarios where the re-execution of a MOEA is necessary.

2.4 Summary

Figure 5 illustrates the actions of the hybridised MOEA which includes the CAO. Trajectories 2-5 describe the specific actions of the CAO. In Figure 6, a Pseudocode description of NSGA-II hybridized with the CAO is also presented.

- Trajectory 1: the mapping between a decision variables vector realised by a MOEA and its corresponding computed objective values vector.
- Trajectory 2: the resulting objective vector a member of the approximation set at generation n - is improved in the objective space.
- Trajectory 3: a prediction of the decision variables vector corresponding to the improved objective vector is made using the neural network trained with the exact data obtained during earlier evaluations of objective functions during the MOEA search.

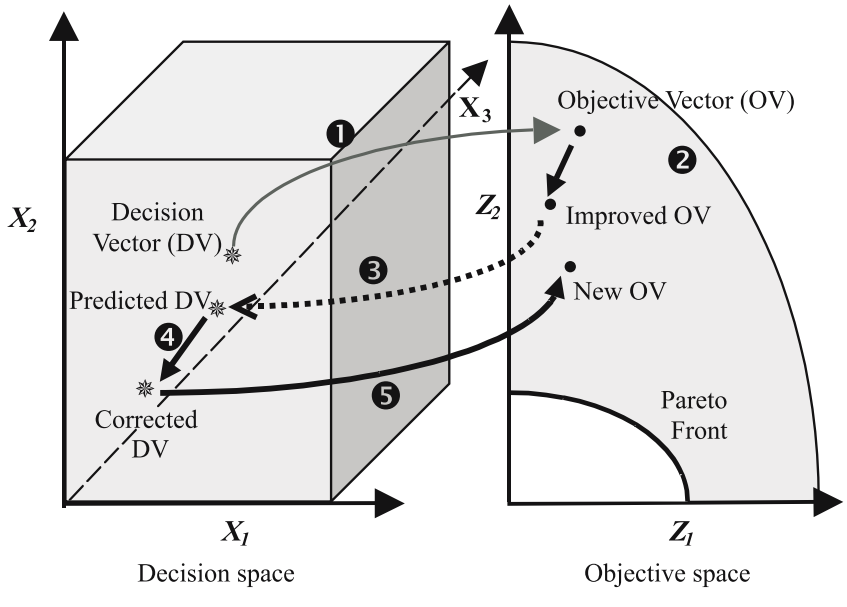


Fig. 5. CAO steps used in generating a single candidate solution

-Generate random population $\mathbf{P_0}$– size **Nind**

-Evaluate objective values

For i=1 to Gen

-Assign rank to $\mathbf{P_{i-1}}$ *(non-dominated sorting strategy)*

-Determine crowding distance between points on each front of solutions in $\mathbf{P_{i-1}}$

-Generate offspring population **Q** – size **Nind**

-Binary tournament selection

-Recombination

-Mutation

-Evaluate objective values for the offspring population **Q**

-Combine parent population $\mathbf{P_{i-1}}$ and offspring Population **Q**

-Assign rank to the combined population

-Determine crowding distance between points on each front of solutions in the combined population

-Select **Nind** solutions to form $\mathbf{P_i}$

Apply CAO

- *Component: Artificial Neural Network Training*
  - *-Initialize an RBF NN and train it with* $\mathbf{P_i}$
  - *-Input: Objective Vectors of* $\mathbf{P_i}$ *– Output: Decision Vectors of* $\mathbf{P_i}$
- *– Component: Objective Space local improvement* **- on** $\mathbf{P_i}$
- *– Component: Objective Space to Decision Space Predictions*
- *– Component: Correction Step*
- *– Update* $\mathbf{P_i}$

End loop

Fig. 6. NSGA-II/CAO Pseudocode

- Trajectory 4: any invalid decision variable vector introduced by the NN mapping is rectified by adjusting out-of-bounds values of the produced decision variables to their nearest values in their domain of definition.

- Trajectory 5: finally, the exact objective values vector for the proposed decision variables vector is calculated in the normal way. These candidate solutions will then compete for archive update and insertion with the best solutions currently stored in the online archive.

3 Test Functions and Performance Metrics

The test functions used to examine the effect of the introduced CAO scheme are: The bi-objective functions:

- ZDT1 (convex test function),
- ZDT2 (non-convex test function)
- ZDT3 (discontinuous test function).

(These test functions belong to a set of test functions [25] that are widely used in Evolutionary Multiobjective Optimisation (EMO) research for testing multiobjective optimisers.)

Further, 3- objective, 5-objective and 8-objective versions of DTLZ2, a scalable test function introduced in [26] to test the effectiveness of MOEAs in dealing with increasing number of objectives, are also used.

NSGA-II and SPEA2 are the comparison benchmark optimisers. Each is also hybridised - NSGA-II/CAO, SPEA2/CAO - with the introduction of the CAO into their cycles to test its effect. Optimiser configurations used in the experiments involving these four optimisers are given in Table 1.

The number of individual objective function evaluations in NSGA-II/CAO and SPEA2/CAO increases from 1 to 2 evaluations per solution for each generation that the CAO is executed. In this study, the CAO is introduced from the 26th to the 50th (last) generation allowing the neural network to be trained during generations 1-25 of the optimisation process. Note that the training of the NN continues throughout the entire optimisation process.

In order to compare the algorithms for the same number of objective function evaluations, the population size of the CAO-hybridized optimisers is reduced to 90 individuals while SPEA2 and NSGAII operate on a population of 135 individuals. For fairness in the comparison, all algorithms are executed for the same number of generations (50), thus maintaining the same level of global search.

The following methods are used to analyze the performance of the optimisers and their CAO-hybridized versions on the bi-objective functions:

- The Pareto fronts achieved by the investigated optimisers and the true Pareto fronts for all of the bi-objective functions used are visually inspected and the average performance over the 10 executions of the algorithms is visualised.
- The generational distance metric [27] is deployed to assess the degree of convergence of solutions by measuring the closeness of the achieved approximation sets to their corresponding true Pareto front.
- The spread metric [18] is used to assess the diversity of the approximation sets achieved by each optimiser.

Table 1. Optimiser Configurations

Size of Population	NSGA-II: 135-200 NSGA-II/CAO: 90 SPEA2: 135-200 SPEA2/CAO: 90
Crossover operator	Simulated Binary Crossover (SBX) Probability: 0.8
Mutation operator	Gaussian Mutation Probability: 1/number of decision variables
Number of Generations	50
Number of Runs	10

- The mean values for the generational distance and spread metrics are calculated for each of the 10 runs of each optimiser. The significance of the observed results is assessed using a randomization testing technique [28], described by [29], whose central concept is that an observed result which had arisen by chance would not appear unusual in a distribution of results obtained through many random relabellings of the samples (in this case, the generational distance and spread metric values).

The effectiveness of the CAO when tackling the DTLZ test functions with 3, 5 and 8 objectives is assessed by using two well-established binary metrics:

- The dominated distance metric (DD-Metric), which computes the dominated distance between two sets of objective vectors [30].
- The C-metric of [25], which calculates the percentage of solutions in a certain approximation set that are dominated or equal to any solution in another competing approximation set.

In [31] it is shown that binary indicators such as these that compare the quality of one approximation set in terms of a certain criterion with another approximation set are suitable metrics to use in order to conclude that a certain approximation set is better than another.

4 Results

The performance and utility of the CAO is investigated in this section. The effect of the introduced operator is examined by comparing the results achieved by NSGA-II and SPEA2 (operating on a population of 135 individuals) with the results achieved by their hybridised versions, NSGA-II/CAO and SPEA2/CAO (operating on a population of 90 individuals). The averaged Pareto fronts achieved by NSGA-II and NSGA-II/CAO after 10 runs (50 generations each run) are illustrated for the test functions ZDT1 (Fig.6), ZDT2 (Fig. 8) and ZDT3 (Fig. 10), together with their true Pareto fronts. Figures 8, 10 and 12 show the results of the randomization testing technique which illustrates the significance of the spread metric values (diversity) and generational distance values (convergence) achieved over the 10 runs of the algorithms. It is clear from Figures 7,

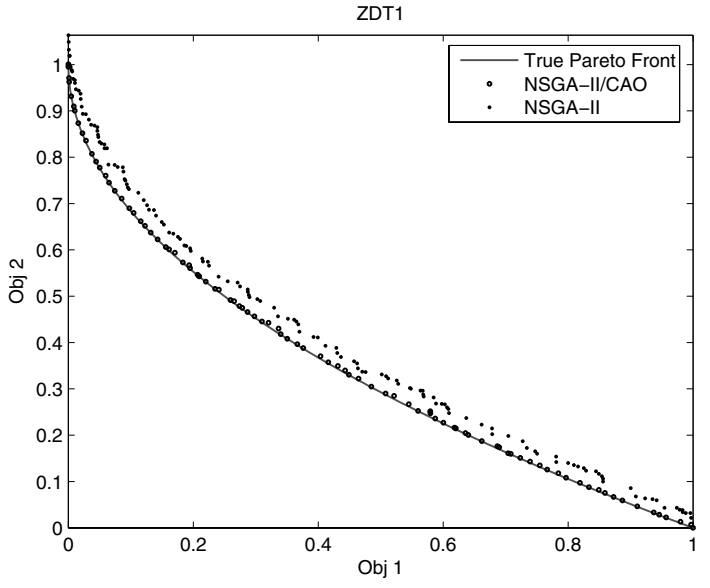


Fig. 7. Results achieved by NSGA-II and NSGA-II/CAO on ZDT1

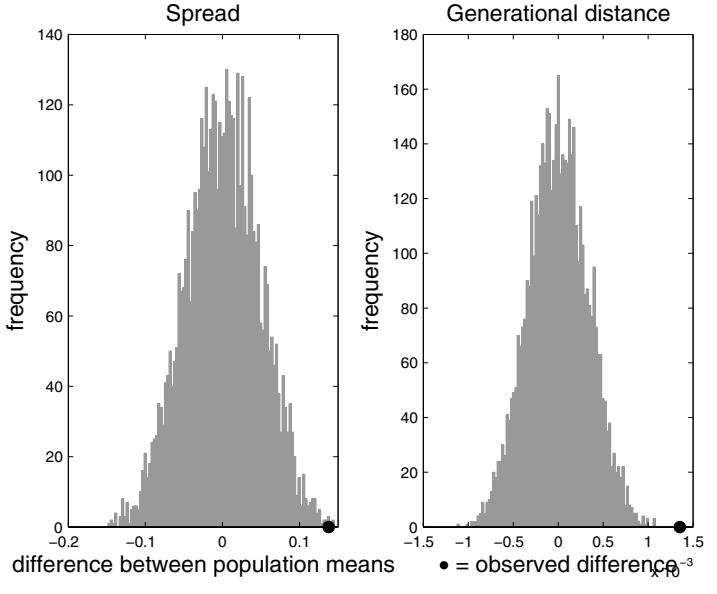


Fig. 8. Randomisation testing of the spread and generational distance metrics on ZDT1 (for NSGA-II and NSGA-II/CAO)

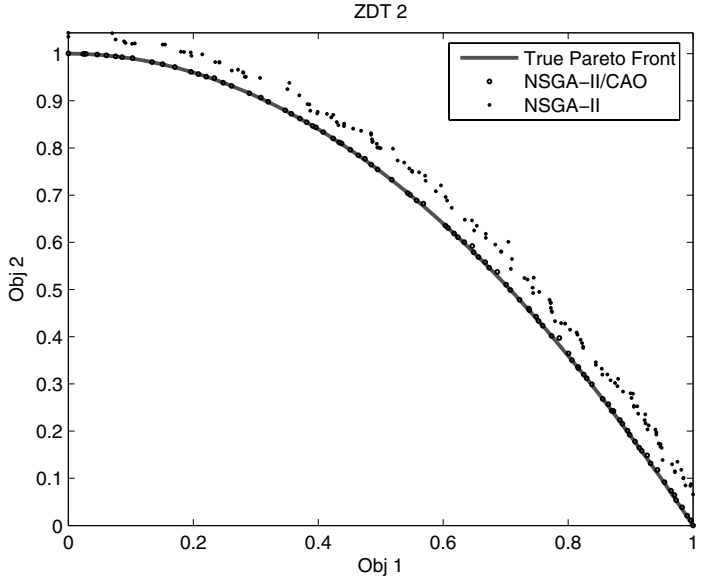


Fig. 9. Results achieved by NSGA-II and NSGA-II/CAO on ZDT2

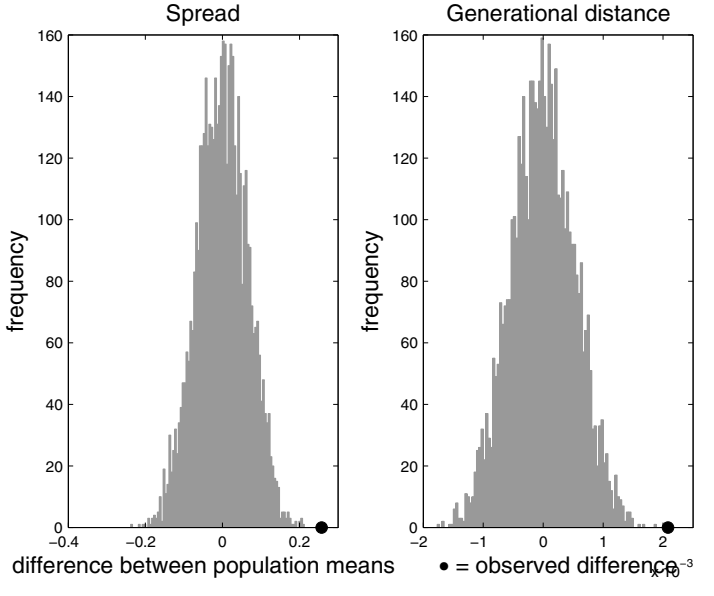


Fig. 10. Randomisation testing of the spread and generational distance metrics on ZDT2 (for NSGA-II and NSGA-II/CAO)

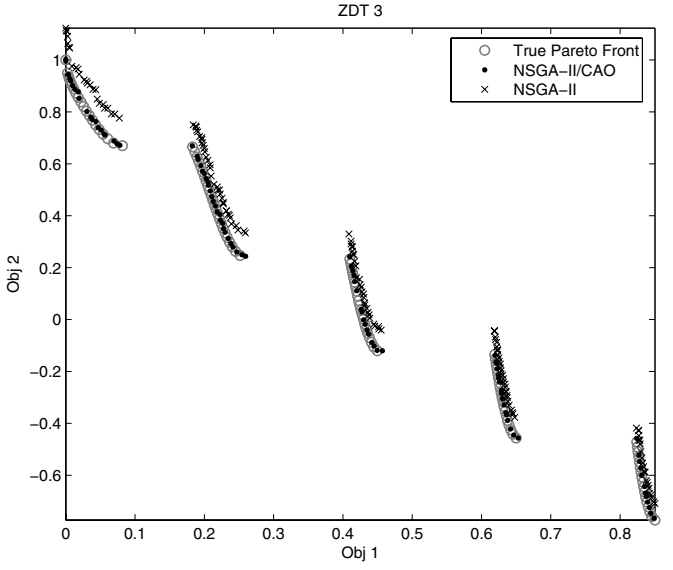


Fig. 11. Results achieved by NSGA-II and NSGA-II/CAO on ZDT3

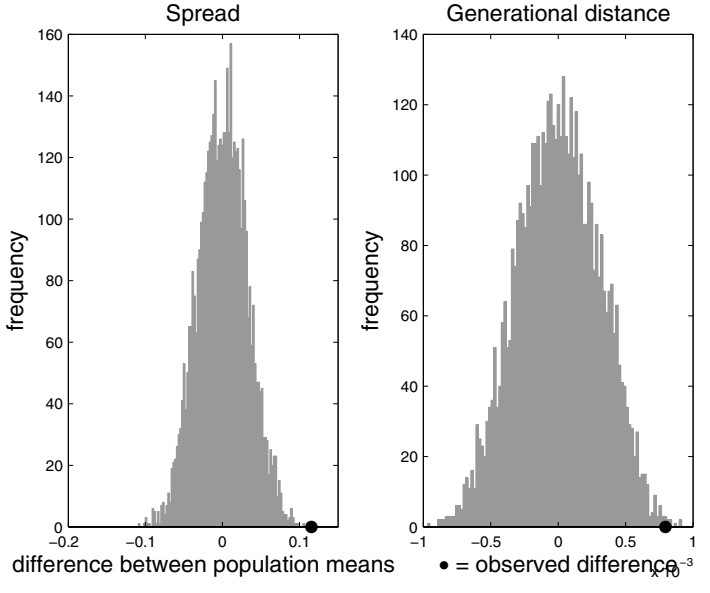


Fig. 12. Randomisation testing of the spread and generational distance metrics on ZDT3 (for NSGA-II and NSGA-II/CAO)

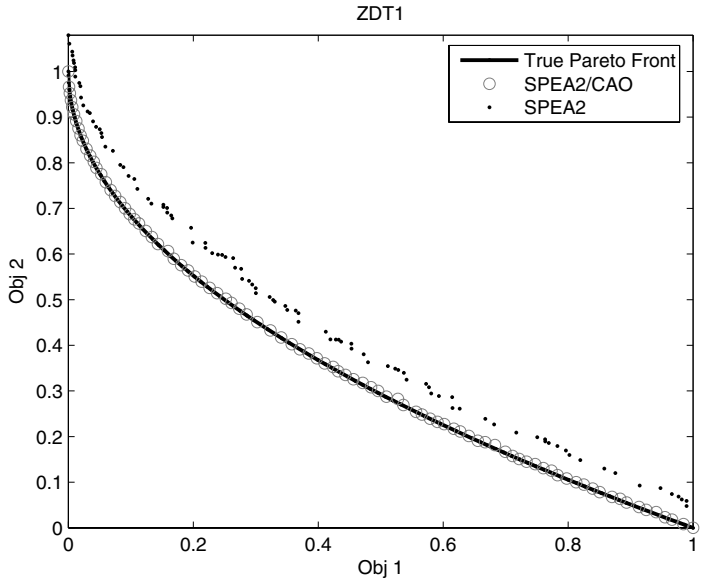


Fig. 13. Results achieved by SPEA2 and SPEA2/CAO on ZDT1

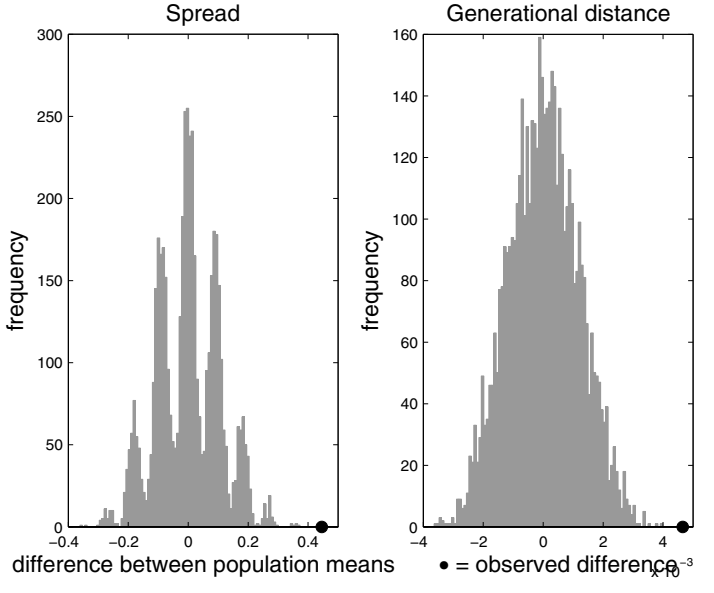


Fig. 14. Randomisation testing of the spread and generational distance metrics on ZDT1 (for SPEA2 and SPEA2/CAO)

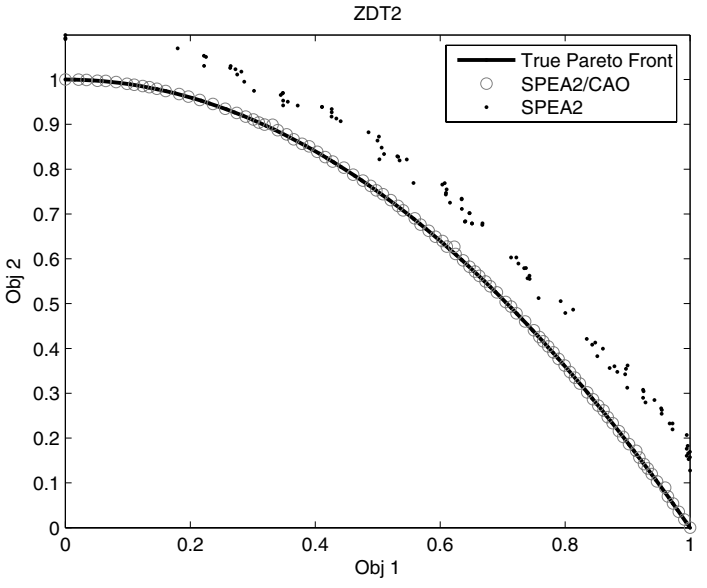


Fig. 15. Results achieved by SPEA2 and SPEA2/CAO on ZDT2

9 and 11 that NSGA-II/CAO outperforms NSGA-II in approximating the true Pareto front of the convex, concave and discontinuous test function.

In Figures 8, 10 and 12, the observed difference between the average calculations of the generational distance values -*measuring the convergence of the solutions and their closeness to the true Pareto front*- achieved by NSGA-II/CAO and NSGA-II, over the 10 runs was calculated and illustrated by a black circle.

The observed difference between the average values of the spread metric measurements depicting the diversity of the solutions achieved by NSGA-II/CAO and NSGA-II at every run of the algorithms is also calculated and presented as a black circle. The grey histograms illustrate the occurrence frequency of the resulting differences between average values of spread metric values and generational distance values shuffled and randomly allocated to the two optimisers. This shuffling and random allocation of the two metrics values was repeated 5000 times to test the significance of the observed results. A smaller spread metric value or generational distance value corresponds, respectively, to a better diversity and closeness to the true Pareto front. The real observations (black circles) lying to the right of the histograms denote a positive difference which favours the CAO hybridized optimiser. (In this work, B is the CAO hybridized optimiser, in the expressions: *Mean (SpreadValues (A)) - Mean (SpreadValues (B)) and Mean (GenerationlDistanceValues (A)) - Mean (GenerationlDistanceValues (B))*.

The randomization testing method has demonstrated a significant and consistent improvement in the performance of the NSGA-II/CAO in terms of convergence towards the true Pareto front and diversity over the 10 executions of the two algorithms. Note

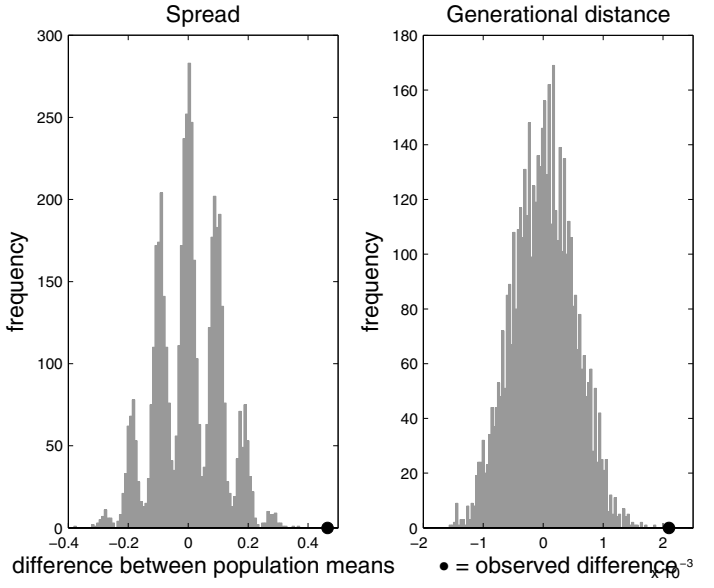


Fig. 16. Randomisation testing of the spread and generational distance metrics on ZDT2 (for SPEA2 and SPEA2/CAO)

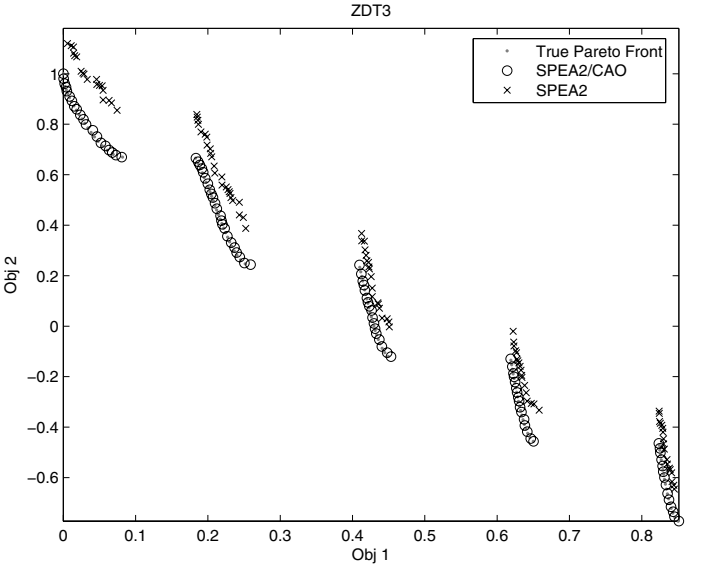


Fig. 17. Results achieved by SPEA2 and SPEA2/CAO on ZDT3

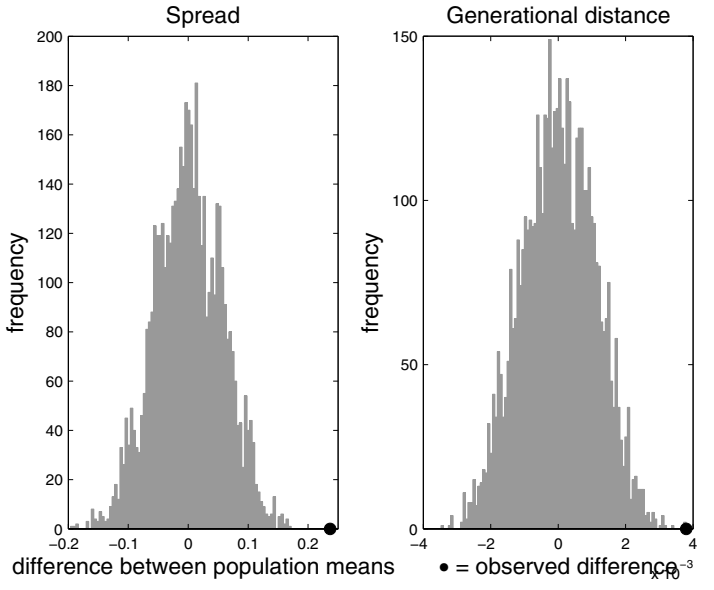


Fig. 18. Randomisation testing of the spread and generational distance metrics on ZDT3 (for SPEA2 and SPEA2/CAO)

Table 2. DD-METRIC AND C-METRIC RESULTS FOR DTLZ2(3)

	DTLZ2(3) A = NSGA-II/CAO and B = NSGA-II		
Run No:	DD-metric(A,B).10^{-3}	C-metric(A,B)	C-metric(B,A)
1	-3.823	8%	2.2%
2	-1.782	4%	2.2%
3	-4.436	10%	0%
4	-4.919	12%	2.2%
5	-3.156	7%	0%
6	-7.297	12%	0%
7	-5.791	13%	0%
8	-0.554	2%	2.2%
9	-9.837	16%	0%
10	-5.320	15%	0%
Mean Value	-4.690	9.9%	0.88%

that NSGA-II/CAO consistently produces a more diversified approximation set compared to NSGA-II, which was operating on a larger population size.

Similar observations are made when CAO is integrated in SPEA2. In fact, the CAO seemed to introduce even more benefits to the performance of SPEA2, which can be seen in Figures 13, 15 and 17 for the bi-objective scenarios. The results of the

Table 3. DD-METRIC AND C-METRIC RESULTS FOR DTLZ2(5)

	DTLZ2(5) A = NSGA-II/CAO and B = NSGA-II		
Run No:	DD-metric(A,B).10^{-3}	C-metric(A,B)	C-metric(B,A)
1	-438.29	53%	4.4%
2	-347.56	47%	3.3%
3	-499.27	55%	2.2%
4	-965.20	83%	0%
5	-786.39	73%	1.1%
6	-535.95	56%	1.1%
7	-775.34	71%	2.2%
8	-295.53	42%	3.3%
9	-417.08	59%	3.3%
10	-473.30	62%	2.2%
Mean Value	-553.39	60.1%	2.3%

Table 4. DD-METRIC AND C-METRIC RESULTS FOR DTLZ2(8)

	DTLZ2(8) A = NSGA-II/CAO and B = NSGA-II		
Run No:	DD-metric(A,B).10^{-3}	C-metric(A,B)	C-metric(B,A)
1	-120.47	7%	0%
2	-271.31	16%	0%
3	-28.98	2%	0%
4	-59.70	5%	2.2%
5	-253.59	16%	0%
6	-37.55	6%	0%
7	-44.95	3%	0%
8	-424.82	24%	0%
9	-59.54	5%	0%
10	-112.40	8%	0%
Mean Value	-141.33	9.2%	0.22%

randomization testing are illustrated in Figures 14, 16, and 18, and, again, demonstrate the impact of the CAO on one of the best-performing MOEAs.

Tables 2 - 7 illustrate the results highlighting the effect of the CAO on optimisation problems with a larger number of objectives. The scalable test function DTLZ2, with 3, 5 and 8 objectives, was chosen to investigate the performance of the CAO. In a similar manner to the experimentations carried out on the bi-objective problems, the effect of the CAO is investigated by contrasting NSGA-II and SPEA2 with their CAO hybridized counterparts. In Tables 2 and 7, the DD-Metric and the C-metric are computed and the results are shown for each run of the algorithms. These metrics are binary metrics that highlight whether an approximation set resulting from an algorithm A is better than another approximation set resulting from an algorithm B. A negative DD-metric value

Table 5. DD-METRIC AND C-METRIC RESULTS FOR DTLZ2(3)

	DTLZ2(3) A = SPEA2/CAO and B = SPEA2		
Run No:	DD-metric(A,B).10^{-3}	C-metric(A,B)	C-metric(B,A)
1	-3.376	7%	1.1%
2	-3.729	9%	0%
3	-3.067	3%	0%
4	-0.324	4%	1.1%
5	-2.992	15%	1.1%
6	-3.992	10%	0%
7	-1.714	6%	0%
8	-3.931	11%	1.1%
9	-1.144	3%	1.1%
10	-1.136	4%	1.1%
Mean Value	-2.540	7.2%	0.66%

Table 6. DD-METRIC AND C-METRIC RESULTS FOR DTLZ2(5)

	DTLZ2(5) A = SPEA2/CAO and B = SPEA2		
Run No:	DD-metric(A,B).10^{-3}	C-metric(A,B)	C-metric(B,A)
1	-12.91	22%	0%
2	-100.15	22%	0%
3	-39.58	16%	0%
4	-103.68	22%	1.1%
5	-186.58	34%	3.3%
6	-253.11	4%	0%
7	-198.26	31%	0%
8	-238.80	37%	0%
9	-6.31	9%	0%
10	-117.82	5%	3.3%
Mean Value	-125.72	20.22%	0.77%

denotes that the first input of the metric (e.g. Algorithm A in DD-Metric (A, B)) is better than and dominates most or part of its second input (e.g. Algorithm B).

These experiments demonstrate that the fronts achieved by SPEA2 and NSGA-II are consistently improved by their counterparts deploying the CAO. For all the dimensions of the problems investigated, the DD-metric reveals results favoring NSGA-II/CAO and SPEA2/CAO over NSGA-II and SPEA2. Over the 10 executions of the algorithms, and despite operating on smaller population sizes, the solutions achieved by NSGA-II/CAO cover an average of 9.9% of the solutions achieved by NSGA-II for the 3-objectives problem.

On the other hand, NSGA-II only scores an average of 0.88% coverage of the results achieved by NSGA-II/CAO, including several runs with 0% coverage. Similar C-metric

Table 7. DD-METRIC AND C-METRIC RESULTS FOR DTLZ2(8)

	DTLZ2(3) A = SPEA2/CAO and B = SPEA2		
Run No:	DD-metric(A,B).10^{-3}	C-metric(A,B)	C-metric(B,A)
1	-28.92	2%	0%
2	-389.58	17%	0%
3	-243.64	11%	0%
4	-127.51	7%	1.1%
5	-8.10	1%	0%
6	-260.31	12%	0%
7	-246.03	13%	0%
8	-38.08	2%	0%
9	-76.08	4%	0%
10	-52.06	2%	1%
Mean Value	-147.03	7.1%	0.21%

observations are made for the 5- and 8-objectives versions of DTLZ2, with a remarkable average of 60.1% solutions coverage favouring NSGA-II/CAO over NSGA-II for the 5-objectives test problem. SPEA2/CAO has out-performs SPEA2 on all three versions of DTLZ (Table 2). The highest coverage achieved by SPEA2 of the solutions obtained by SPEA2/CAO is 0.77%, while SPEA2/CAO covers at least an average of 7.1% of the approximation sets achieved by SPEA2. Again, in the 5-objectives version of DTLZ, SPEA2/CAO exhibits the most significant improvement in coverage over SPEA2. This feature deserves further study, using tools such as the heat maps of [32], in order to understand why the performance on the 5-objectives version might be significant for this dimension of problem.

Further experiments were undertaken in an attempt to quantify the extent of superiority of the CAO hybridized optimisers. It was noted that, on average, the population size of NSGA-II and SPEA2 must be increased to a minimum of 200 individuals (more than twice the population size of NSGA-II/CAO and SPEA2/CAO) in order to match the quality of the fronts achieved by their hybridized counterparts. Thus, SPEA2 and NSGA-II require more objective function evaluations (around 750 more evaluations) to match the performance of their CAO hybridised equivalent optimiser. This conclusion holds for all the test functions used in this work. The set of experiments conducted in this Section demonstrate the benefits of the CAO and the improvement it confers to two of the most established MOEAs.

5 Conclusions and Future Work

A Convergence Accelerator Operator is proposed for incorporation in existing algorithms for evolutionary multiobjective optimisation. This operator works by suggesting improved solutions in objective space and using neural network mapping schemes to predict the corresponding solution points in decision variable space. Two leading MOEAs have been hybridised through introduction of the CAO and tested on a variety

of recognised test problems. These test problems consisted of convex, concave and discontinuous test functions, with numbers of objectives ranging from two to eight. In all cases introduction of the CAO led to improved convergence and solution diversity for comparable numbers of function evaluations. Indeed, for the bi-objective test problems, improved performance in diversity is achieved by the hybridised algorithms for smaller population sizes than those used by the standard algorithms.

It is important to recognise that the CAO introduces additional computational effort through the requirement to train the neural network. This computational effort is substantial when compared with the execution time associated with computing a ZDT function, for example, since these functions are trivially simple to compute. The CAO is designed for use in real-world problems where objective function computation is non-trivial. For example, NN training time proved to be approximately 1500 times that of computing the two ZDT1 functions. Clearly, one would not advocate use of CAO in such situations. However, in a real-world problem such as the ALSTOM gasifier problem [33], it was found that NN training time proved to be approximately one-hundredth of the time required to compute the ALSTOM gasifier problem objectives. Moreover, here we have not sought to optimise performance of the NN mapping methodology.

Thus, a portable operator has been described that can be incorporated into any MOEA to improve its convergence. Its value is in application to real-world problems where there is a substantial computational cost for objective function evaluation. Future work should focus on interactively executing the CAO on request by the DM and in the deployment of the operator in a progressive preference articulation technique, for example [34], to assist in guiding the search towards specific regions of interest (ROI). Further, the interpolation step factor used for objective space improvement is an application-dependent parameter and will be influenced by the landscape of the objective space. In the experiments presented in here, step factors ranging from 0.01 up to 0.2 were investigated before settling for h=0.1 as the step factor to be used for the tests. There is scope to explore the use of adaptive step factors as MOEAs explore the objective space.

References

1. Fleming, P., Purshouse, R.C., Lygoe, R.J.: Many-objective optimization: An engineering design perspective. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 14–32. Springer, Heidelberg (2005)
2. Purshouse, R.C.: On the Evolutionary Optimisation of Many Objectives. PhD thesis, Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, UK (September 2003)
3. Farina, M.: A neural network based generalized response surface multiobjective evolutionary algorithm. In: Proceedings of the 2002 Congress on Evolutionary Computation. IEEE Service Center, Piscataway (2002)
4. El-Beltagy, M., Nair, P., Keane, A.: Metamodeling techniques for evolutionary optimization of computationally expensive problems: Promises and limitations. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999). Morgan Kaufmann, San Francisco (1999)

5. Adra, S.F., Hamody, A.I., Griffin, I., Fleming, P.J.: A Hybrid Multi-Objective Evolutionary Algorithm Using an Inverse Neural Network for Aircraft Control System Design. In: 2005 IEEE Congress on Evolutionary Computation (CEC 2005), vol. 1, pp. 1–8. IEEE Service Center, Edinburgh (2005)
6. Nariman-Zadeh, N., Atashkari, K., Jamali, A., Pilechi, A., Yao, X.: Inverse modelling of multi-objective thermodynamically optimized turbojet engines using GMDH-type neural networks and evolutionary algorithms. Engineering Optimization 37, 437–462 (2005)
7. Darwin, C.: The Origin of Species. John Murray, London (1859)
8. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor (1975)
9. Moscato, P.: On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. Tech. Rep. Caltech Concurrent Computation Program, Report. 826, California Institute of Technology, Pasadena, California, USA (1989)
10. Dawkins, R.: The Selfish Gene. Oxford University Press, Oxford (1976)
11. Bhanu, B., Lee, S., Das, S.: Adapting image segmentation using genetic and hybrid search methods. IEEE Transactions on Aerospace and Electronic Systems 31(4), 1268–1291 (1995)
12. E, E.B., Cowling, P., Silva, J., Petrovic, S.: Combining hybrid metaheuristics and populations for the multiobjective optimisation of space allocation problems. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001). Morgan Kaufmann, San Francisco (2001)
13. Haas, O., Burnham, K., Mills, J.: Hybrid optimisation technique for radiotherapy treatment planning. In: Proceedings of the 1998 IEEE International Conference on Control Applications. Trieste, Italy (1998)
14. Hodgson, R.: Memetic algorithms and the molecular geometry optimization problem. In: Proceedings of the 2000 Congress on Evolutionary Computation, pp. 625–632. IEEE Service Center, Piscataway (2000)
15. Davis, L.: Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York (1991)
16. Knowles, J.D.: Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization. PhD thesis, The University of Reading, Reading, UK (2002)
17. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Reading (1989)
18. 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)
19. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength Pareto evolutionary algorithm. TIK Report 103, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2001)
20. Gaspar-Cunha, A., Vieira, A.: A hybrid multi-objective evolutionary algorithm using an inverse neural network. In: Hybrid Metaheuristics, First International Workshop, HM. Valencia, Spain (2004)
21. Gaspar-Cunha, A., Vieira, A., Fonseca, C.: Multi-objective optimization: Hybridization of an evolutionary algorithm with artificial neural networks for fast convergence. In: Workshop on Design and Evaluation of Advanced Hybrid Meta-Heuristics, Nottingham, UK (November 2004)
22. Jin, R., Chen, W., Simpson, T.: Comparative studies of metamodelling techniques under multiple modelling criteria. Structural and Multidisciplinary Optimization 23(1), 1–13 (2001)
23. Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (1995)
24. Adra, S.F., Griffin, I., Fleming, P.J.: An informed convergence accelerator for evolutionary multiobjective optimisers. In: 2007 Genetic and Evolutionary Computation Conference(GECCO 2007), London, UK (July 2007)

25. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8(2), 173–195 (2000)
26. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002) (IEEE Neural Networks Council, ed.), vol. 1, pp. 825–830. IEEE Service Center, Piscataway (2002)
27. Veldhuizen, D.A.V.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio (1999)
28. Manly, B.F.J.: Randomization and Monte Carlo Methods in Biology. Chapman and Hall, London (1991)
29. Purshouse, R.C., Fleming, P.J.: An adaptive divide-and-conquer methodology for evolutionary multi-criterion optimisation. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 133–147. Springer, Heidelberg (2003)
30. Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (1999)
31. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance assessment of multiobjective optimizers: An analysis and review. IEEE Transactions on Evolutionary Computation 7(2), 117–132 (2003)
32. Pryke, A., Mostaghim, S., Nazemi, A.: Heatmap visualization of population based multi objective algorithms. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 361–375. Springer, Heidelberg (2007)
33. Griffin, I.A., Schroder, P., Chipperfield, A.J., Fleming, P.J.: Multi-objective optimization approach to the ALSTOM gasifier problem. Proceedings of the Institution of Mechanical Engineers Part I: Journal of Systems and Control 214, 453–468 (2000)
34. Fonseca, C.M., Fleming, P.J.: Multiobjective optimization and multiple constraint handling with evolutionary algorithms — Part I: A unified formulation. IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans 28(1), 26–37 (1998)

Part III

Knowledge Propagation through Cultural Evolution

Risk and Cost Tradeoff in Economic Dispatch Including Wind Power Penetration Based on Multi-Objective Memetic Particle Swarm Optimization

Lingfeng Wang and Chanan Singh

Department of Electrical and Computer Engineering, Texas A&M University,
College Station, United States
l.f.wang@ieee.org, singh@ece.tamu.edu

Utilization of renewable energy resources such as wind energy for electric power generation has assumed great significance in recent years. Wind power is a source of clean energy and is able to spur the reductions of both consumption of depleting fuel reserves and emissions of pollutants. However, since the availability of wind power is highly dependent on the weather conditions, the penetration of wind power into traditional utility grids may incur certain security implications. Therefore, in economic power dispatch including wind power penetration, a reasonable tradeoff between system risk and operational cost is desired. In this chapter, a bi-objective economic dispatch problem considering wind penetration is first formulated, which treats operational costs and security impacts as conflicting objectives. Different fuzzy membership functions are used to reflect the dispatcher's attitude toward the wind power penetration. A multi-objective memetic particle swarm optimization (MOMPSO) algorithm is adopted to develop a power dispatch scheme which is able to achieve compromise between economic and security requirements. Numerical simulations including comparative studies are reported based on a typical IEEE test power system to show the validity and applicability of the proposed approach.

1 Introduction

The major objective of Economic Dispatch (ED) is to schedule the power generation in an appropriate manner in order to satisfy the load demand while minimizing the total operational cost [1-4]. These ED problems are usually highly nonlinear and the metaheuristic methods have turned out to be more effective than the traditional analytical methods. In recent years, renewable energy resources such as wind power have shown great prospects in decreasing fuel consumption as well as reducing pollutants emission [5-11]. Unfortunately, the expected generation output from a wind park is difficult to predict accurately, primarily due to the intermittent nature of the wind coupled with the highly nonlinear wind energy conversion. This unpredictability may incur the security problems when the penetration of wind power in the traditional power system exceeds a certain level. For instance, the dynamic system stability may be lost due to excessive wind fluctuations. As a result, for achieving the tradeoff between system risk and total

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 209–230.
springerlink.com

running cost, it is desirable to examine how to dispatch the power properly for the power system taking into account the impacts of wind power penetration. In this paper, the ED model is first constructed as a bi-objective optimization problem through simultaneous minimization of both risk level and operational cost. For this purpose, an effective optimization prodecure is needed. Particle swarm optimization (PSO) is a salient metaheuristics, which turns out to be capable of resolving a wide variety of highly nonlinear and complex engineering optimization problems with outstanding convergence performance. Meanwhile, it has strong ability to avoid premature convergence. In this study, a multi-objective memetic particle swarm optimization (MOMPSO) algorithm is proposed to derive the Pareto-optimal solutions for economic dispatch including wind power penetration. Moreover, considering the different attitudes of dispatchers towards wind power penetration, we used several fuzzy membership functions to indicate the system security level in terms of wind power penetration and wind power cost. Different fuzzy representations including linear and quadratic functions can be used to reflect the dispatcher's optimistic, neutral, or pessimistic attitude toward wind power penetration. Furthermore, minimization of pollutant emissions is treated as the third design objective and the tradeoff surface among the three design objectives is also derived.

The remainder of the chapter is organized as follows. Section 2 presents the wind penetration model described by different fuzzy membership functions. Section 3 formulates the economic dispatch problem including its multiple objectives and a set of design constraints imposed. Section 4 introduces the inner workings of particle swarm optimization algorithms. The MOMPSO algorithm adopted is discussed in Section 5. Simulation results and analysis are presented in Section 6. Finally, conclusions are drawn and future research is suggested.

2 Wind Power Penetration Model

Wind power integration is an important issue to address for achieving a reliable power system including wind power source. Because of the unpredictable and variable characteristics of wind power, its integration into the traditional thermal generation systems will incur the operator's concern on system security. Fuzzy definition regarding wind penetration is a viable way to represent the penetration level of the wind power, since it is usually difficult to determine the optimal wind power that should be integrated into the conventional power grids [12].

As shown in Figure 1, a fuzzy membership function μ regarding the wind penetration is defined to indicate the system security level. It can be mathematically expressed in the following form [12]:

$$\mu = \begin{cases} 1, & W \le W(P_D)_{min} \\ \frac{W(P_D)_{max} - W}{W(P_D)_{max} - W(P_D)_{min}}, & W_{min} \le W \le W_{max} \\ 0, & W \ge W(P_D)_{max} \end{cases} \tag{1}$$

where W is the wind power incorporated in economic dispatch; $W(P_D)_{min}$ is the lower bound of wind power penetration, below which the system is deemed secure; $W(P_D)_{max}$ is the upper bound of wind power penetration, above which the system is considered as

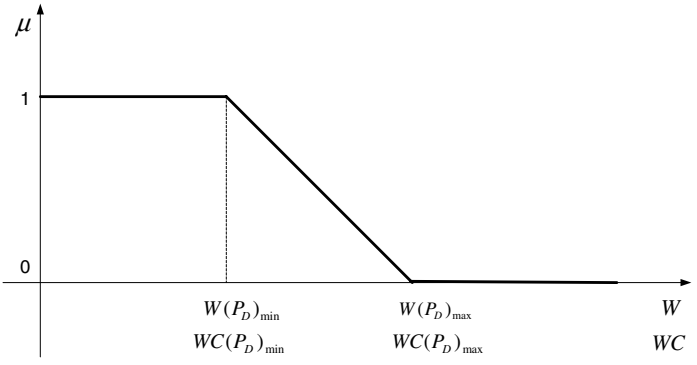


Fig. 1. Fuzzy linear representation of the security level in terms of wind penetration and wind power cost

insecure due to the wind perturbations. Both $W(P_D)_{min}$ and $W(P_D)_{max}$ are dependent on the total load demand in the power dispatch.

The above defined membership function can also be represented in terms of the operational cost for incorporating wind power:

$$\mu = \begin{cases} 1, & WC \le WC(P_D)_{min} \\ \frac{WC_{max} - WC}{WC_{max} - WC_{min}}, & WC_{min} \le WC \le WC_{max} \\ 0, & WC \ge WC(P_D)_{max} \end{cases} \tag{2}$$

where WC is the running cost of wind power in the power dispatch; $WC(P_D)_{min}$ is the lower bound cost for producing wind power, below which the system is seen as secure; $W(P_D)_{max}$ is the upper bound cost for including wind power, above which the system is considered as insecure due to the wind intermittency. In a similar fashion, both $WC(P_D)_{min}$ and $WC(P_D)_{max}$ are dependent on the total load demand in the power dispatch. In this study, sensitivity studies are also carried out to illustrate the impact of different allowable ranges of wind power penetration as well as different running costs of wind power on the final solutions obtained.

To reflect dispatcher's differing attitudes toward wind power penetration, a quadratic membership function can be defined as follows [12]:

$$\mu = \begin{cases} 1, & W \le W(P_D)_{min} \\ a_w W^2 + b_w W + c_w, & W_{min} \le W \le W_{max} \\ 0, & W \ge W(P_D)_{max} \end{cases} \tag{3}$$

where a_w, b_w, and c_w are the coefficients of the quadratic function, which determine its shape reflecting the dispatcher's attitude toward wind power. As shown in Figure 2, by selecting different coefficients a_w, b_w, and c_w, different shapes of the quadratic function can be defined. For the identical security level μ_0, the penetration levels of wind power differ for different defined functions $w_1 < w_2 < w_3$. The curves corresponding to these three values reflect the pessimistic, neutral, and optimistic attitudes of the dispatcher toward the wind power integration, respectively.

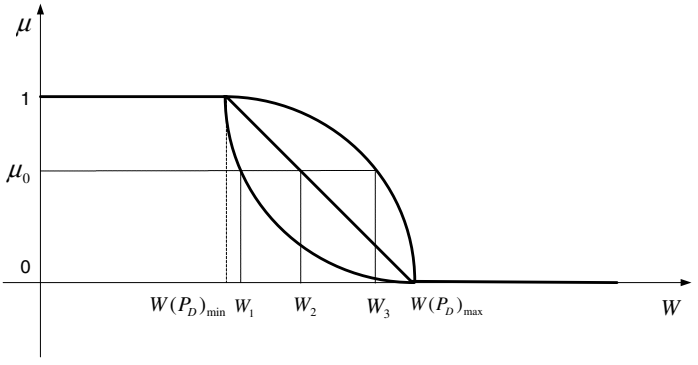


Fig. 2. Fuzzy quadratic representation of the security level in terms of wind power penetration

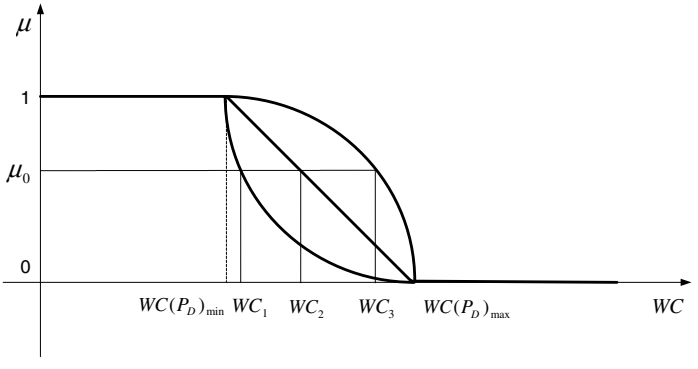


Fig. 3. Fuzzy quadratic representation of the security level in terms of wind power cost

In a similar fashion, the security level can also be defined in terms of the operational cost of wind power. Its function shape is shown in Figure 3.

$$\mu = \begin{cases} 1, & WC \le WC(P_D)_{min} \\ a_c WC^2 + b_c WC + c_c, & WC_{min} \le WC \le WC_{max} \\ 0, & WC \ge WC(P_D)_{max} \end{cases} \tag{4}$$

where a_c, b_c, and c_c determine the curve shape of the quadratic function defined in terms of the running cost of wind power.

3 Problem Formulation

The problem of economic power dispatch with wind penetration consideration can be formulated as a bi-criteria optimization model. The two conflicting objectives, i.e., total operational cost and system risk level, should be minimized simultaneously while fulfilling certain system constraints. This bi-objective optimization problem is formulated mathematically in this section.

3.1 Problem Objectives

There are two objectives that should be minimized simultaneously, that is, system risk level and the total operational cost.

- Objective 1: Minimization of system risk level
  From the security level function defined in (1) and (2), we know that the larger the value of membership function μ is, the more secure the system will become. If the wind penetration is restricted under a certain level, the system can be considered as secure. On the contrary, if excessive wind penetration is introduced into the power dispatch, the system may become insecure. Here we define an objective function which should be minimized in order to ensure system security:
  $$R(\mu) = \frac{1}{\mu} \tag{5}$$
- Objective 2: Minimization of operational cost
  The cost curves of different generators are represented by quadratic functions with sine components. The superimposed sine components represent the rippling effects produced by the steam admission valve openings. The total \$/h fuel cost $FC(P_G)$ can be represented as follows:
  $$FC(P_G) = \sum_{i=1}^{M} a_i + b_i P_{Gi} + c_i P_{Gi}^2 + |d_i \sin[e_i(P_{Gi}^{min} - P_{Gi})]| \tag{6}$$
  where M is the number of generators committed to the operating system, a_i, b_i, c_i, d_i, e_i are the cost coefficients of the i-th generator, and P_{Gi} is the real power output of the ith generator. P_G is the vector of real power outputs of generators and defined as
  $$P_G = [P_{G1}, P_{G2}, \ldots, P_{GM}] \tag{7}$$
  The running cost of wind power can be represented in terms of the value of membership function μ which indicates the system security level. For the linear membership function case,
  $$WC(P_G, \mu) = C_w(W_{av} - (P_D + P_L - \sum_{i}^{M} P_{Gi})) - \mu * \Delta WC + WC_{max} \tag{8}$$
  where W_{av} is the available wind power from the wind farm, C_w is the coefficient of penalty cost for not using all the available wind power, P_D is the load demand, P_L is the transmission loss, and
  $$\Delta WC = WC_{max} - WC_{min}. \tag{9}$$
  For the quadratic membership function case,
  $$WC(P_G, \mu) = C_w(W_{av} - (P_D + P_L - \sum_{i}^{M} P_{Gi})) - \frac{b_c}{2a_c} \pm \sqrt{\frac{\mu - (c_c - \frac{b_c^2}{4a_c})}{a_c}} \tag{10}$$
  The sign of the last term in (10) is determined by the curve shape of the defined quadratic function. Thus, the total operational cost TOC can be calculated as
  $$TOC(P_G, \mu) = FC(P_G) + WC(P_G, \mu) \tag{11}$$

3.2 Problem Constraints

Due to the physical or operational limits in practical systems, there is a set of constraints that should be satisfied throughout the system operations for a feasible solution.

- Constraint 1: Generation capacity constraint
  For normal system operations, real power output of each generator is restricted by lower and upper bounds as follows:

$$P_{Gi}^{min} \leq P_{Gi} \leq P_{Gi}^{max} \tag{12}$$

  where P_{Gi}^{min} and P_{Gi}^{max} are the minimum and maximum power from generator i, respectively.
- Constraint 2: Power balance constraint
  The total power generation and the wind power must cover the total demand P_D and the real power loss in transmission lines P_L. For the linear membership function, this relation can be represented by

$$\sum_{i=1}^{M} P_{Gi} + W_{max} - \mu * \Delta W = P_D + P_L \tag{13}$$

  For the quadratic membership function, the relation can be expressed by

$$\sum_{i=1}^{M} P_{Gi} - \frac{b_w}{2a_w} \pm \sqrt{\frac{\mu - (c_w - \frac{b_w^2}{4a_w})}{a_w}} = P_D + P_L \tag{14}$$

  The sign of the last term in (14) is determined by the curve shape of the defined quadratic function. The transmission losses can be calculated based on the Kron's loss formula as follows:

$$P_L = \sum_{i=1}^{M} \sum_{j=1}^{M} P_{Gi} B_{ij} P_{Gj} + \sum_{i=1}^{M} B_{0i} P_{Gi} + B_{00} \tag{15}$$

  where B_{ij}, B_{0i}, B_{00} are the transmission network power loss **B**-coefficients. It should be noted that the transfer loss of the wind power is not considered in this study.
- Constraint 3: Available wind power constraint
  The wind power used for dispatch should not exceed the available wind power from the wind park:

$$0 \leq P_D + P_L - \sum_{i}^{M} P_{Gi} \leq W_{av} \tag{16}$$

- Constraint 4: Security level constraint
  From the definition of membership function shown from (1) to (4), the values of μ should be within the interval of $[0,1]$:

$$0 \leq \mu \leq 1. \tag{17}$$

3.3 Problem Statement

In summary, the objective of economic power dispatch optimization considering wind penetration is to minimize $R(\mu)$ and $TOC(P_G, \mu)$ simultaneously subject to the constraints (12)–(17).

4 Mechanism of Particle Swarm Optimization

Particle swarm optimization (PSO) is a population-based stochastic optimization technique, which was inspired by the movement pattern in a bird flock or fish school [13, 14]. In PSO, individuals (i.e., particles) move around in a multidimensional search space to approach the optima, where each point represents a solution to the target problem. Initially a bunch of particles are randomly created and set into motion through this space. In their movement, each particle adjusts its position based on its own experience as well as the experience of a neighboring particle by utilizing the best position encountered by itself and its neighbors. At each generation, they observe the "fitness" of themselves and their neighbors and move toward those with a better position. In this way, PSO combines both local and global search methods together in order to improve its search effectiveness and efficiency. Unlike other evolutionary computation algorithms including Genetic Algorithms (GA), PSO has no evolution operators such as crossover and mutation. The optima is obtained via following the current optimum particles by the potential particles. This simple algorithm turns out to be highly effective in a diverse set of optimization problems.

Let x and v denote a particle coordinates (position) and its corresponding flight speed (velocity) in the search space. Therefore, the i-th particle is represented as $x_i = [x_{i1}, x_{i1}, \ldots, x_{id}, \ldots, x_{iM}, x_{i,M+1}]$ in the $(M+1)$-dimensional space. Each particle keeps track of its coordinates in the solution space which are associated with the best solution it has achieved so far. This fitness value is called *pbest*. The best previous position of the i-th particle is recorded and represented as $pbest_i = [pbest_{i1}, pbest_{i2}, \ldots, pbest_{id}, \ldots, pbest_{iM}, pbest_{i,M+1}]$. Another "best" value that is tracked by the particle swarm optimizer is the best value obtained so far by any particle in the neighbors of the particle. When a particle takes all the population as its topological neighbors, the best value is a global best and is called *gbest*. The index of the best particle among all the particles in the group is represented by the $gbest_d$. The rate of the velocity for particle i is represented as $v_i = (v_{i1}, v_{i2}, \ldots, v_{id}, \ldots, v_{iM}, v_{i,M+1})$. The modified velocity and position of each particle can be calculated using the current velocity and the distance from $pbest_{id}$ to $gbest_d$ as shown in the following formulas:

$$v_{id}^{(t+1)} = \chi * (w * v_{id}^{(t)} + c_1 * \text{rand}() * (pbest_{id} - x_{id}^{(t)}) + c_2 * \text{Rand}() * (gbest_d - x_{id}^{(t)})), \tag{18}$$

$$x_{id}^{(t+1)} = x_{id}^{(t)} + v_{id}^{(t+1)}, i = 1, 2, \ldots, N, d = 1, 2, \ldots, M+1. \tag{19}$$

where N is the number of particles in a group, $M+1$ is the number of members in a particle, t is the pointer of generations, $\chi \in [0,1]$ is the constriction factor which controls the velocity magnitude, w is the inertia weight factor, c_1 and c_2 are acceleration constants, rand() and Rand() are uniform random values in a range $[0,1]$, $v_i^{(t)}$ is the velocity of particle i at generation t, and $x_i^{(t)}$ is the current position of particle i at generation t. The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its *pbest* and *gbest* locations. Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *gbest* locations.

As we can see, in PSO fewer parameters need to be adjusted as compared with other meta-heuristics such as genetic algorithms. PSO has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a particular requirement.

Usually, traditional gradient-based optimization methods deal with multi-objective optimization problems by allocating weights to each of the objectives, which indicate their importance in the overall problem. However, it is often hard to find weights which can accurately reflect the real-life situation. Moreover, it is highly possible that these methods are not capable of detecting solutions lying in concave regions of the Pareto front [15]. Meta-heuristics such as evolutionary optimization techniques are especially suited to handle the multi-objective optimization problems since they are able to search simultaneously for multiple Pareto-optimal solutions. A set of Pareto-optimal solutions is derived during the optimization, in which each cost is found so that the whole set is no worse than any other set of solutions. In this research, an enhanced multi-objective optimization particle swarm optimization algorithm is designed to resolve the target power dispatch problem.

5 The Proposed Approach

The standard PSO algorithm is not suited to resolve multi-objective optimization problems in that no absolute global optimum exists there, but rather a set of non-dominated solutions. Thus, to render the PSO algorithm capable of dealing with MO problems, some modifications become necessary. Parsopoulos and Vrahatis [15] used the weighted aggregation approach to handle MO problems by converting multiple objectives into a single one. The weights can be fixed or adaptive in the optimization. To approximate the Pareto front, the algorithm needs to run multiple times. Hu and Eberhart [16] proposed a dynamic neighborhood strategy which uses one-dimension optimization to cope with multiple objectives. Later, Hu, Eberhart, and Shi [17] modified this method by introducing an extended memory which stores the global Paerto-optimal solutions in order to reduce the computational cost. Coello Coello and Lechuga [18] present an MOPSO which maintains the previously found non-dominated solutions. These solutions serve as the guides for the flight of particles. Mostaghim and Teich [19] proposed a sigma method, which adopts the best local guides for each particle to promote the convergence and diversity of the MOPSO approach.

When using stochastic search based algorithms to optimize multi-objective problems, two key issues usually arise in the algorithm design. First, the fitness evaluation should be suitably designed to guide the search toward the set of Pareto-optimal solutions. Second, the diversity of the population should be maintained by refraining the search from premature convergence. In this study, the classic PSO algorithm is revised accordingly to facilitate a multi-objective optimization approach. Meanwhile, local search and other mechanisms are incorporated to improve its performance which leads to a memetic algorithm termed multi-objective memetic particle swarm optimization (MOMPSO) [20]. A “meme” refers to a unit of cultural evolution capable of conducting local refinements. That is, individual could improve itself prior to communicating with the population it is in. Combining local search into traditional heuristic

optimization methods has turned out to be able to achieve orders of magnitude faster search for some problem domains.

5.1 Archiving

The Pareto-dominance concept is used to evaluate the fitness of each particle and thus determine which particles should be selected to store in the archive of non-dominated solutions. Similar to the elitism used in evolutionary algorithms and the tabu list used in tabu searches, the best historical solutions found by the population are recorded continuously in the archive in order to serve as the non-dominated solutions generated in the past. The major function of the archive is to store a historical record of the non-dominated solutions found along the heuristic search process. The archive interacts with the generational population in each iteration so as to absorb superior current non-dominated solutions and eliminate inferior solutions currently stored in the archive. The non-dominated solutions obtained at every iteration in the generational population (swarm) are compared with the contents of archive on a one-per-one basis. A candidate solution can be added to the archive if it meets any of the following conditions:

- The archive is empty;
- The archive is not full and the candidate solution is not dominated by or equal to any solution currently stored in the archive;
- The candidate solution dominates any existing solution in the archive;
- The archive is full but the candidate solution is non-dominated and is in a less crowded region than at least one solution.

Furthermore, due to the global attraction mechanism in PSO, the historical archive of previously found non-dominated solutions would make the search converge toward globally non-dominated solutions highly possible.

5.2 Global Best Selection

In MOPSO, gbest plays an important role in directing the whole swarm move toward the Pareto front. Very often, the rapid swarm converges within the intermediate vicinity of the gbest may lead to the diversity loss and premature convergence. To resolve this, Fuzzy Global Best (f-gbest) scheme [20] is adopted in this study, which is based on the concept of possibility measure to model the lack of information about the true optimality of the gbest. In this scheme, the gbest refers to the possibility of a particle at a certain location, rather than a sharp location as defined in traditional PSO algorithms. In this way, the particle velocity can be calculated as follows:

$$p_{c,d}^{k} = N(g_{g,d}^{k}, \delta) \tag{20}$$

$$\delta = f(k) \tag{21}$$

$$v_{i,d}^{k+1} = w * v_{i,d}^{k} + c_1 * r_1^{k} * (p_{i,d}^{k} - x_{i,d}^{k}) + c_2 * r_2^{k} * (p_{c,d}^{k} - x_{i,d}^{k}) \tag{22}$$

where $p_{c,d}^{k}$ is the dth dimension of f-gbest in cycle k. The f-gbest is represented by a normal distribution $N(p_{g,d}^{k}, \delta)$, where δ indicates the degree of uncertainty regarding the

optimality of the gbest position. To reflect the reduction of this uncertainty as the search proceeds, δ can be defined as a nonincreasing function of the number of iterations. For instance, here $f(k)$ is defined as a simple function

$$f(k) = \begin{cases} \delta_{max}, & \text{cycles} < \xi * \text{max\_cycles} \\ \delta_{min}, & \text{otherwise} \end{cases} \tag{23}$$

where ξ is a user-specified parameter which affects the change of δ. We can see that the f-gbest function is designed to enable the particles to explore a region beyond that defined by the search trajectory of original PSO. f-gbest encourages global exploration at the early search stage when δ is large, and facilitates local fine-tuning at the late stage when δ decreases. Thus, this scheme tends to reduce the possibility of premature convergence as well as enhance the population diversity.

5.3 Local Search

During the heuristic multi-objective optimization process, since the MO optimization algorithm is attempting to build up a discrete picture of a possibly continuous Pareto front, it is often desired to distribute the solutions as diversely as possible on the discovered tradeoff curve. Furthermore, the uniformity among the distributed solutions is also crucial so as to achieve consistent and smooth transition among the solution points when searching for the best compromise solution based on the particular requirements of the target problem. Therefore, to accomplish these challenges, it is highly necessary to preserve the diversity of solutions distribution during the optimization process. In this investigation, the combination of a local search termed Synchronous Particle Local Search (SPLS) [20] into MOPSO can be regarded as an effective measure for preserving distribution diversity and uniformity as well as speeding up the search process.

SPLS carries out guided local fine-tuning so as to promote the distribution of non-dominated solutions, whose computational procedure is laid out in the following [20]:

- Choose S_{LS} individuals randomly from the population.
- Choose N_{LS} non-dominated individuals with the best niche count from the archive and store them in the selection pool.
- Allocate an arbitrary non-dominated individual from the selection pool to each of the S_{LS} individuals as *gbest*.
- Allocate an arbitrary search space dimension for each of the S_{LS} individuals.
- Assimilation operation: With the exception of the assigned dimension, update the position of S_{LS} individuals in the search space with the selected *gbest* position.
- Update the position of all S_{LS} assimilated individuals using (20)-(22) along the pre-assigned dimension only.

5.4 Constraints Handling

Because the standard PSO does not take into account how to deal with the constraints, the constraints handling mechanism should be added to ensure the solution feasibility in constrained optimization problems such as power dispatch. In the proposed MOMPSO,

a simple constraint checking procedure called rejecting strategy is incorporated. When an individual is evaluated, the constraints are first checked to determine if it is a feasible candidate solution. If it satisfies all of the constraints, it is then compared with the non-dominated solutions in the archive. Or else, it is dumped. The dumped individual is then replaced by a randomly created one. Here the concept of Pareto dominance is applied to determine if it is eligible to be chosen to store in the archive of non-dominated solutions. The constraint satisfaction checking scheme used in the proposed algorithm proves to be quite effective in ensuring the feasibility of the non-dominated solutions.

5.5 Individual (particle) Representation

It is crucial to appropriately encode the individuals of the population in PSO for handling the economic dispatch application. The power output of each generating unit and the value of membership function are chosen to represent the particle position in each dimension, and positions in different dimensions constitute an individual (particle), which is a candidate solution for the target problem. The position in each dimension is real-coded. The i-th individual P_{Gi} can be represented as follows:

$$P_{Gi} = [P_{Gi1}, P_{Gi2}, \ldots, P_{Gid}, \ldots, P_{GiM}, \mu_i], i = 1, 2, \ldots, N \tag{24}$$

where M is the number of generators and N is the population size; P_{Gid} is the power generated by the d-th unit in i-th individual; and μ_i is the value of the membership function in i-th individual. Thus, the dimension of a population is $N \times (M+1)$.

5.6 Algorithm Steps

In principle, an archive-based MOPSO algorithm can be illustrated in Figure 4. As seen from the figure, initially the population is randomly created, and then the selection pressure from the PSO algorithm drives the population move towards the better positions. At each iteration, the generational population is updated and certain elite individuals from it are chosen to store in the elite population (archive) based on the Pareto dominance concept. It should be noted that each individual also maintains a memory which records the best positions the individual has encountered so far. The personal best for each particle is selected from this memory. Meanwhile, among the individuals stored in the archive, the global best is singled out according to our fuzzified global guide selection strategy. Both guides are then used by the PSO to steer the search to promising regions. The procedure is repeated until the maximum number of iterations is reached or the solutions cease improving for a certain number of generations. Under this framework, other multi-objective optimization algorithms can also be developed based on different meta-heuristics such as evolutionary computation, simulated annealing, and tabu search.

The proposed MOMPSO is applied to the constrained power dispatch problem in order to derive the optimal or near-optimal solutions. Its computational steps include:

- Step 1: Specify the lower and upper bounds of generation power of each unit as well as the range of security level.

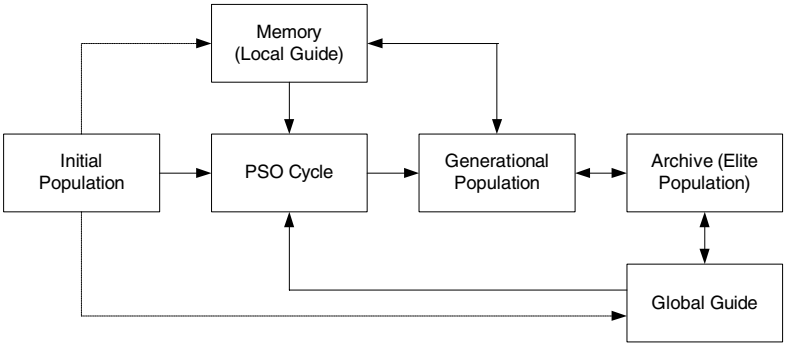


Fig. 4. Data flow diagram of the proposed algorithm

- Step 2: Randomly initialize the individuals of the population. Note that the speed and position of each particle should be initialized such that each candidate solution (particle) locates within the feasible search space.
- Step 3: For each individual P_{Gi} of the population, the transmission loss P_{Li} is calculated based on **B**-coefficient loss formula.
- Step 4: Evaluate each individual P_{Gi} in the population based on the concept of Pareto-dominance.
- Step 5: Store the non-dominated members found thus far in the archive.
- Step 6: Initialize the memory of each particle in which a single local best *pbest* is stored. The memory is contained in another archive.
- Step 7: Increment iteration counter.
- Step 8: Choose the personal best position *pbest* for each particle based on the memory record; Choose the global best *gbest* according to the aforementioned f-gbest selection mechanism. Meanwhile, local search based on SPLS is carried out. The niching and fitness sharing mechanism is also applied throughout this process for enhancing the diversity of solutions.
- Step 9: Update the member velocity v of each individual P_{Gi} based on (18). For the output of each generator,

$$\begin{aligned} v_{id}^{(t+1)} &= \chi * (w * v_i^{(t)} + c_1 * \text{rand}() * (pbest_{id} - P_{Gid}^{(t)}) \\ &\quad + c_2 * \text{Rand}() * (gbest_d - P_{Gid}^{(t)})), \\ &\quad i = 1, \ldots, N; d = 1, \ldots, M \end{aligned} \tag{25}$$

where N is the population size, and M is the number of generating units. For the value of membership function μ,

$$\begin{aligned} v_{i,M+1}^{(t+1)} &= \chi * (w * v_i^{(t)} + c_1 * \text{rand}() * (pbest_{i,M+1} - \mu_i^{(t)}) \\ &\quad + c_2 * \text{Rand}() * (gbest_{M+1} - \mu_i^{(t)})), \end{aligned} \tag{26}$$

- Step 10: Modify the member position of each individual P_{Gi} based on (19). For the output of each generator,

$$P_{Gid}^{(t+1)} = P_{Gid}^{(t)} + v_{id}^{(t+1)} \tag{27}$$

For the value of membership function μ,

$$\mu_i^{(t+1)} = \mu_i^{(t)} + v_{i,M+1}^{(t+1)} \tag{28}$$

Following this, add the turbulence factor into the current position. For the output of each generator,

$$P_{Gid}^{(t+1)} = P_{Gid}^{(t+1)} + R_T P_{Gid}^{(t+1)} \tag{29}$$

For the value of membership function μ,

$$\mu_i^{(t+1)} = \mu_i^{(t+1)} + R_T \mu_i^{(t+1)} \tag{30}$$

where R_T is the turbulence factor. The turbulence term is used here to enhance the diversity of solutions.

- Step 11: Update the archive which stores non-dominated solutions according to the aforementioned four Pareto-optimality based selection criteria.
- Step 12: If the current individual is dominated by the *pbest* in the memory, then keep the *pbest* in the memory; Otherwise, replace the *pbest* in the memory with the current individual.
- Step 13: If the maximum iterations are reached, then go to Step 14. Otherwise, go to Step 7.
- Step 14: Output a set of Pareto-optimal solutions from the archive as the final solutions.

6 Simulation and Evaluation of the Proposed Approach

In this study, a typical IEEE 30-bus test system with 6-generators is used to investigate the effectiveness of the proposed MOMPSO approach. The system configuration is shown in Figure 5. The system parameters including fuel cost coefficients and generator capacities are listed in Table 1. The sinusoidal term in (6) is not considered in this study due to its relatively minor impact on the total fuel costs. The **B**-coefficients are shown in (31). The load demand P_D used in the simulations is 2.834 p.u., the available wind power W_{av} is 0.5668 p.u., and the coefficient of penalty cost C_w is set 20 \$/p.u.

Table 1. Fuel cost coefficients and generator capacities

Generator i	a_i	b_i	c_i	P_{Gi}^{min}	P_{Gi}^{max}
G_1	10	200	100	0.05	0.50
G_2	10	150	120	0.05	0.60
G_3	20	180	40	0.05	1.00
G_4	10	100	60	0.05	1.20
G_5	20	180	40	0.05	1.00
G_6	10	150	100	0.05	0.60

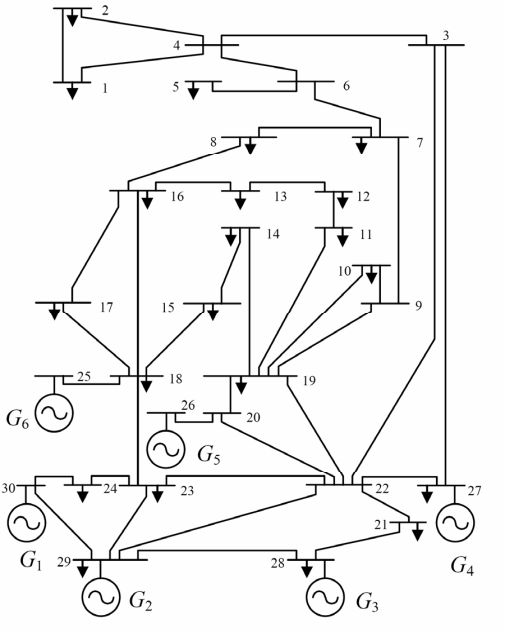


Fig. 5. IEEE 30-bus test power system

$$B_{ij} = \begin{bmatrix} 0.1382 & -0.0299 & 0.0044 & -0.0022 & -0.0010 & -0.0008 \\ -0.0299 & 0.0487 & -0.0025 & 0.0004 & 0.0016 & 0.0041 \\ 0.0044 & -0.0025 & 0.0182 & -0.0070 & -0.0066 & -0.0066 \\ -0.0022 & 0.0004 & -0.0070 & 0.0137 & 0.0050 & 0.0033 \\ -0.0010 & 0.0016 & -0.0066 & 0.0050 & 0.0109 & 0.0005 \\ -0.0008 & 0.0041 & -0.0066 & 0.0033 & 0.0005 & 0.0244 \end{bmatrix} \tag{31}$$

6.1 Comparison of Different Design Scenarios

Since PSO algorithms are sometimes quite sensitive to certain parameters, the simulation parameters should be appropriately chosen. In the simulations, both the population size and archive size are set to 100, and the number of generations is set to 500. The acceleration constants c_1 and c_2 are chosen as 1. Both turbulence factor and niche radius are set to 0.02. The inertia weight factor w decreases when the number of generations increases:

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times iter \tag{32}$$

where $iter_{max}$ is the number of generations and $iter$ is the current number of iterations. From the above equation, we can appreciate that the value of w will decrease as the iteration number increases. In the search process, the most efficient way to locate the optimal or near-optimal solutions in a complex large search space is first to move to the smaller solution space as promptly as possible, and then seek out the desired solution in this space via thorough search. The parameter w is defined to regulate the size of

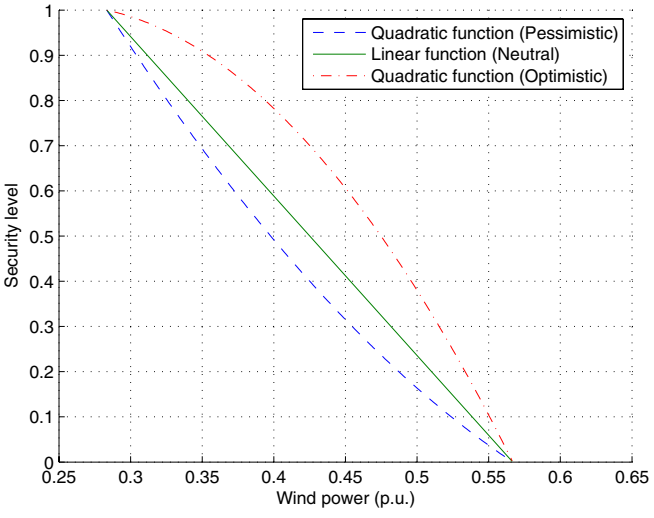


Fig. 6. Different curve shapes of membership functions

Table 2. Example solutions for different design scenarios

Generators/objectives	Pessimistic	Linear	Optimistic
P_{G1}	0.0678	0.0500	0.0605
P_{G2}	0.2463	0.2430	0.2425
P_{G3}	0.3863	0.4029	0.4221
P_{G4}	0.9164	0.9300	0.9354
P_{G5}	0.4034	0.3990	0.3490
P_{G6}	0.3166	0.2929	0.2897
W	0.5043	0.5232	0.5408
Cost ($/hour)	518.893	515.436	512.147
Risk level	6.5864	6.49894	6.31094

search step of each particle. At first, the value of w is set relatively large in order to drive the particle to the solution area quickly. Then, when the particle approaches the desired solution, the size of each search step becomes smaller in order to prevent the particle from flying past the target position during the flight. In this way, the desired solutions can be sought through gradual refinement. For the f-gbest parameters, δ_{max}, δ_{min}, and ξ are chosen as 0.15, 0.0001, and 0.4, respectively. The simulation program is coded using C++ and executed in a 2.20 GHz Pentium-4 processor with 512 MB of RAM. In simulations, the minimum and maximum allowable wind power penetrations are set as 10% and 20% of the total load demand, respectively. The running cost of wind power is calculated based on its linear relationship with the amount of wind power integrated, i.e., $WC = \sigma W$. The coefficient σ indicating the running cost of wind power is set 50 \$/p.u. in the simulation. The parameters used in the simulations are listed below and different function curves are shown in Figure 6.

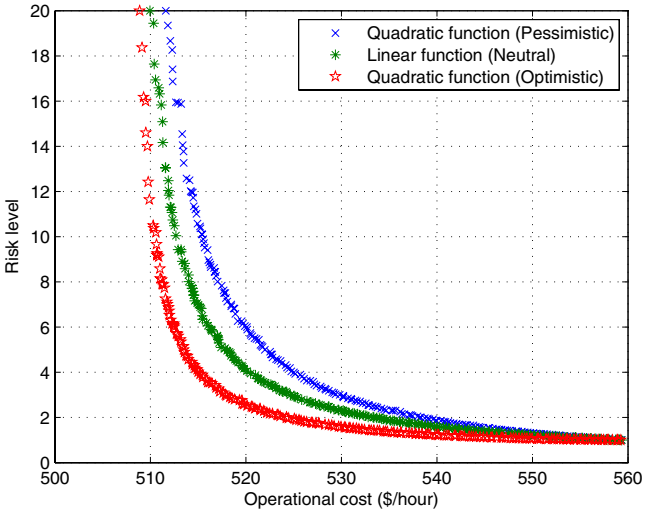


Fig. 7. Pareto fronts obtained based on different membership functions

- Quadratic representation (optimistic design): $a_w = -9.9607$, $b_w = 4.94$, $c_w = 0.4$;
- Linear representation (neutral design): W_{min}=0.2834, W_{max}=0.5668;
- Quadratic representation (pessimistic design): $a_w = 4.9803$, $b_w = -7.7629$, $c_w = 2.8$.

The illustrative non-dominated solutions derived in different design scenarios are listed in Table 2 and the Pareto-optimal fronts evolved using the proposed MOMPSO are shown in Figure 7.

As shown in the figure, the Pareto-optimal solutions are widely distributed on the tradeoff surface due to the diversity preserving mechanisms used in the proposed MOMPSO algorithm. Unlike the single-objective optimization, in multi-objective optimization the decision maker can choose a suitable solution based on his/her preference from a pool of non-dominated solutions. We can also appreciate that for the same risk level calculated from different membership functions, the optimistic design has the lowest operational cost since it includes the largest amount of wind power among all of the three designs. Wind power has the lowest operational cost as compared with the same amount of electric power from fuel-based generation.

6.2 Sensitivity Analysis

Sensitivity analysis has been carried out in order to illustrate the impacts of different allowable ranges of wind power penetration as well as different running costs of wind power on the final non-dominated solutions derived. Here the linear membership function is used. We herein quantify the impact of wind penetration through numerical simulations by changing the permissible ranges of wind power penetration $[W(P_D)_{min}, W(P_D)_{max}]$. In the simulations for determining the impact of different allowable wind penetration ranges, the running cost of wind power is kept unchanged,

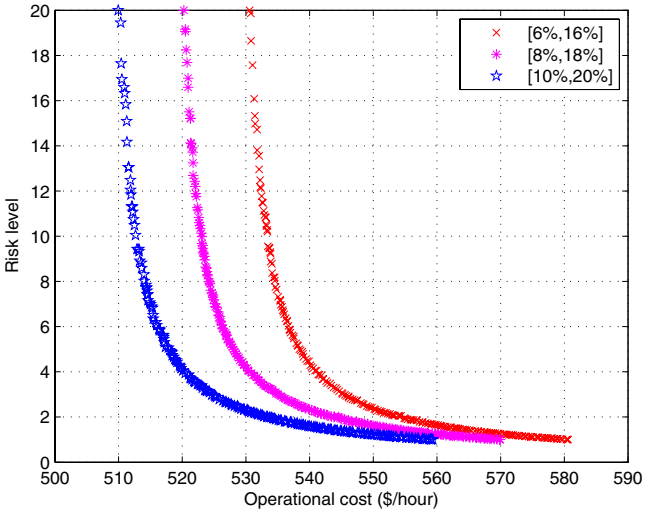


Fig. 8. Pareto fronts obtained for different wind penetration ranges

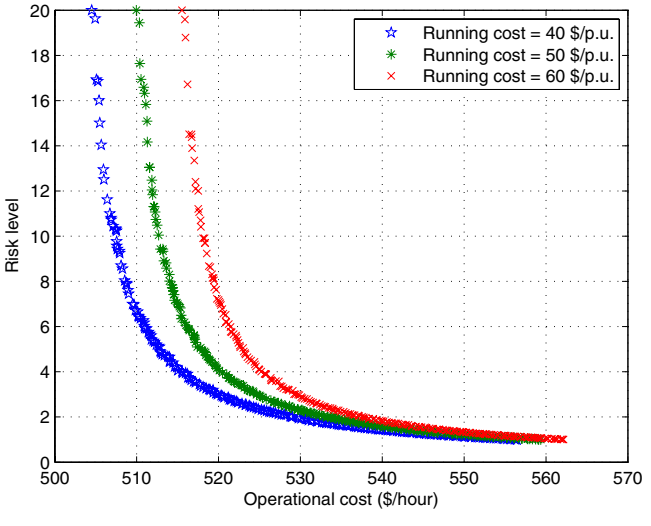


Fig. 9. Pareto fronts obtained for different running costs of wind power

i.e., $\sigma = 50\$/p.u.$ In a similar fashion, in the simulations for examining the impact of running costs of wind power, the penetration range of wind power is fixed, i.e., $[W(P_D)_{min}, W(P_D)_{max}] = [10\% * P_D, 20\% * P_D]$. The derived Pareto fronts are shown in Figure 8 and Figure 9, respectively. From the figures, we can appreciate that the non-dominated solutions vary with the different ranges of allowable wind penetration as well as the running costs of wind power. In Figure 8, at the same risk level, the design

scenario with the largest value of maximum allowable wind penetration W_{max} has the lowest cost since the most portion of wind power is integrated. In Figure 9, at the identical risk level, the scenario with the lowest running cost of wind power results in the lowest overall cost since the same amount of wind power is integrated with the lowest cost.

6.3 Comparative Studies

In order to conduct a quantitative assessment of the performance of a multi-objective optimization algorithm, normally four major issues should be considered:

- Minimize the distance of the approximated Pareto front obtained from the target algorithm with respect to the true Pareto front. Since the true Pareto front is unknown in most practical problems, this requirement can be interpreted as increasing the accuracy of the obtained final solutions (i.e., to render the obtained Pareto front as close to the true one as possible).
- Maximize the spread of the obtained solutions. It means that the diversity of the solutions should be maximized in the optimization run by rendering the distribution of the solutions as smooth and even as possible. Thus, the decision-maker will have more choices for different demands on decision.
- Maximize the number of elements of the Pareto-optimal set. The extent of the estimated Pareto front should be increased, i.e., a wide range of non-dominated solutions in objective space should be derived by the optimizer. For each objective, a sufficiently wide range of values should be covered by the non-dominated solutions.
- Minimize the computational cost. Most of the MO optimization algorithms are computationally expensive, thus the computational time is an important criterion for measuring the efficacy of an algorithm in dealing with such problems.

Thus, certain quantitative metrics need to be defined in order to compare the performance of different MO algorithms in a more objective fashion. A comparative study is conducted to examine how competitive the proposed approach is in dealing with the target problem. Comparison of Pareto front estimates is not a easy task since it is unlikely to evaluate the quality of a Pareto front using any single measure. Here, we use four measures to comprehensively assess the performance of different optimizers in terms of accuracy, diversity, extent, and computational economy. Note that all the following comparisons are conducted based on the neutral design scenario with the original design parameters.

- **C-metric:** C-metric is quite often used to examine the quality of the Pareto fronts obtained [21]. Let $S_1, S_2 \subseteq S$ be two sets of decision solutions. The C-metric is defined as the mapping between the ordered pair (S_1, S_2) and the interval $[0,1]$:

$$\mathsf{C}(S_1, S_2) = \frac{|\{a_2 \in S_2; \exists a_1 \in S_1 : a_1 \preceq a_2\}|}{|S_2|} \tag{33}$$

  Provided that $\mathsf{C}(S_1, S_2) = 1$, all solutions in S_2 are dominated by or equal to solutions in S_1. If $\mathsf{C}(S_1, S_2) = 0$, then none of the solutions in S_2 are covered by S_1. Both $\mathsf{C}(S_1, S_2)$ and $\mathsf{C}(S_2, S_1)$ should be checked in the comparison since C-metric is not

Table 3. Comparison of C-metric for different algorithms

	C(M, O)	C(O, M)	C(M, F)	C(F, M)	C(M, S)	C(S, M)
Best	1.0	0.1001	1.0	0.1427	1.0	0.1503
Worst	0.9899	0.0	0.9890	0.0	0.9811	0.0
Average	0.9993	0.0001	0.9905	0.0002	0.9880	0.0005
Median	0.9995	0.0001	0.9928	0.0002	0.9882	0.0005
Std. Dev.	0.0001	0.0001	0.0009	0.0001	0.0010	0.0002

Table 4. Comparison of the spacing metric for different algorithms

Spacing	MOPSO	f-gbest	SPLS	MOMPSO
Best	0.0489	0.0403	0.0397	0.0389
Worst	0.1077	0.0996	0.0988	0.0767
Average	0.0526	0.0422	0.0418	0.0402
Median	0.0521	0.0412	0.0410	0.0375
Std. Dev.	0.2562	0.1131	0.0988	0.0612

symmetrical in its arguments, i.e., the equation $\mathsf{C}(S_1,S_2) = 1 - \mathsf{C}(S_2,S_1)$ does not necessarily always hold. Table 3 illustrates the comparison of C-metric for different algorithms, where "O", "F", "S", and "M" indicate the original MOPSO, MOPSO only with f-gbest selection, MOPSO only with SPLS, and MOMPSO, respectively.

- **Spacing:** The spacing metric is defined to measure the spread of the non-dominated solutions. For instance, one of such metrics is to measure the range variance of neighboring individuals in the set of non-dominated solutions [22]. It is defined as

$$SP = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\overline{d} - d_i)^2} \tag{34}$$

where $d_i = \min_j(|f_1^i(\overrightarrow{x}) - f_1^j(\overrightarrow{x})| + |f_2^i(\overrightarrow{x}) - f_2^j(\overrightarrow{x})|), i, j = 1, \ldots, n$, $\overline{d}$ is the mean of all d_i, and n is the number of non-dominated solutions found so far. The larger the value is, the more unevenly the solution members are spaced. If the value of SP is zero, all members in the Pareto front are equidistantly distributed. Table 4 illustrates the comparison of the spacing metric for different algorithms. For simplicity of notation, here f-gbest is used to denote the MOPSO only with f-gbest and SPLS represents the MOPSO only with SPLS.

- **Error ratio:** The metric termed error ratio was proposed by Van Veldhuizen [23] to indicate the percentage of solutions that are not members of the Pareto-optimal set. It is given as

$$ER = \frac{\sum_{i=1}^{n} e_i}{n} \tag{35}$$

where n is the number of solutions in the current set of non-dominated solutions. If vector i belongs to the Pareto-optimal set, then $e_i = 0$; or else $e_i = 1$. It is evident that if $ER = 0$, all the solutions derived from the optimizer are members of the Pareto-optimal set. Table 5 illustrates the comparison of the error ratio metric for different algorithms.

Table 5. Comparison of the error ratio metric for different algorithms

Error ratio	MOPSO	f-gbest	SPLS	MOMPSO
Best	0.0051	0.0	0.0	0.0
Worst	0.3698	0.2899	0.2640	0.2486
Average	0.2305	0.1196	0.1181	0.1030
Median	0.2209	0.1090	0.1086	0.0980
Std. Dev.	0.2253	0.1643	0.1904	0.1088

Table 6. Comparison of computational time for different algorithms (in seconds)

CPU time	MOPSO	f-gbest	SPLS	MOMPSO
Best	15.2	14.9	12.8	11.2
Worst	19.9	17.9	16.4	12.5
Average	17.5	16.0	13.1	11.7
Median	17.9	16.4	13.4	11.6
Std. Dev.	0.1024	0.0549	0.0602	0.0121

- **Computational time:** Under the exactly identical environments including both hardware and software platforms, different algorithms are compared in terms of the CPU time consumed in obtaining their corresponding Pareto fronts. Table 6 illustrates the time needed for obtaining 100 mutually non-dominated solutions for different algorithms.

From the above comparative studies, we can find the following three major advantages of the solutions obtained by MOMPSO with respect to those derived from the other three algorithms:

- Higher quality solutions are obtained using the MOMPSO. In the set of solutions obtained using MOMPSO, most of them have better objective function values than those derived from other approaches.
- The solutions of MOMPSO have better diversity characteristics. This is primarily due to the several diversity retention mechanisms used:
  - A useful distribution preservation measure adopted in this study is the fuzzification mechanism used during the selection of global guide for each particle in every generation;
  - A local search scheme termed Synchronous Particle Local Search (SPLS) is integrated into the original MOPSO in order to enhance the population diversity and expedite the convergence speed;
  - Niching and fitness sharing is used to preserve the population diversity by preventing the population from falling into the detrimental genetic drift;
  - In the archiving process, the individuals lying in less populated regions of the objective have higher chance to be chosen than those in the highly populated areas;
  - A turbulence factor is added to the PSO operations, which may increase the solution diversity by enhancing the randomness in a particle's movements.

The diversity is preserved by MOMPSO while there is no guarantee that the solutions obtained by objective aggregation approach will span over the entire tradeoff surface.

- The computational time of MOMPSO is less than the other methods. The SPLS mechanism incorporated reduces the computational expense due to its capability of facilitating convergence speedup.

7 Concluding Remarks

This chapter investigates the integration of wind power into conventional power networks and its impact on generation resource management. Wind power is environmentally friendly since it is able to reduce the fossil fuel and natural gas consumption. Also, wind power needs less operational cost since it does not consume fossil fuels and natural gases. However, due to the intermittent and variable nature of the wind power, it is usually quite difficult to determine how much wind power should be integrated to ensure both power system security and operational cost reduction. In this chapter, fuzzy representations of system security in terms of wind power penetration level and operational costs are adopted in constructing economic dispatch models. A multi-objective memetic particle swarm optimization (MOMPSO) algorithm is developed to derive the non-dominated Pareto-optimal solutions in terms of the specified multiple design objectives. Different design scenarios can be formulated according to dispatcher's attitudes toward wind power integration with respect to risk and cost. A numerical application example is used to illustrate the validity and applicability of the developed optimization procedure. In the further investigations, probabilistic methods may also be adopted to handle various uncertainties in power systems including wind power penetration.

Acknowledgment

The authors would like to thank the financial support from NSF for this research (Grant No. ECS0406794: Exploring the Future of Distributed Generation and Micro-Grid Networks).

References

1. Hota, P.K., Dash, S.K.: Multiobjective generation dispatch through a neuro-fuzzy technique. Electric Power Components and Systems 32, 1191–1206 (2004)
2. Jayabarathi, T., Jayabarathi, K., Jeyakumar, D.N., et al.: Evolutionary programming techniques for different kinds of economic dispatch problems. Electric Power System Research 73, 169–176 (2005)
3. Lee, K.Y., Yome, A.S., Park, J.H.: Adaptive Hopfield neural networks for economic load dispatch. IEEE Transactions on Power Systems 13(2), 519–526 (1998)
4. Gaing, Z.-L.: Particle swarm optimization to solving the economic dispatch considering the generator constraints. IEEE Transactions on Power Systems 18(3), 1187–1195 (2003)
5. DeMeo, E.A., Grant, W., Milligan, M.R., Schuerger, M.J.: Wind plant integration: costs, status, and issues. IEEE Power & Energy Magazine, 38–46 (November/December 2005)

6. Douglas, J.: Putting wind on the grid. EPRI Journal, 6–15 (Spring 2006)
7. Eriksen, P.B., Ackermann, T., Abildgaard, H., Smith, P., Winter, W., Garcia, R.: System operation with high wind penetration. IEEE Power & Energy Magazine, 65–74 (November/December 2005)
8. Jenkins, N., et al.: Embedded Generation, IEE Power and Energy Series 31, The Institution of Electrical Engineers (2000)
9. Piwko, R., Osborn, D., Gramlich, R., Jordan, G., Hawkins, D., Porter, K.: Wind energy delivery issues. IEEE Power & Energy Magazine, 67–56 (November/December 2005)
10. Smith, J.C.: Wind of change: issues in utility wind integration. IEEE Power & Energy Magazine, 20–25 (November/December 2005)
11. Zavadil, R., Miller, N., Ellis, A., Muljadi, E.: Making connections: wind generation challenges and progress. IEEE Power & Energy Magazine, 26–37 (November/December 2005)
12. Miranda, V., Hang, P.S.: Economic dispatch model with fuzzy constraints and attitudes of dispatchers. IEEE Transactions on Power Systems 20(4), 2143–2145 (2005)
13. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE Proceedings of the International Conference on Neural Networks, Perth, Australia, pp. 1942–1948 (1995)
14. Kennedy, J., Eberhart, R.: Swarm Intelligence. Morgan Kaufmann Publishers, San Francisco (2001)
15. Parsopoulos, K.E., Vrahatis, M.N.: Particle swarm optimization method in multi-objective problems. In: Proceedings of the ACM Symposium on Applied Computing, Madrid, Spain, pp. 603–607 (2002)
16. Hu, X., Eberhart, R.: Multiobjective optimization using dynamic neighborhood particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, Honolulu, Hawaii, pp. 1677–1681 (2002)
17. Hu, X., Eberhart, R.C., Shi, Y.: Particle swarm with extended memory for multiobjective optimization. In: Proceedings of the IEEE Swarm Intelligence Symposium, Indianapolis, Indiana, pp. 193–197 (2003)
18. Coello Coello, C.A., Lechuga, M.S.: MOPSO: A proposal for multiple objective particle swarm optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, Honolulu, HI, pp. 1051–1056 (2002)
19. Mostaghim, S., Teich, J.: Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO). In: IEEE Proceedings of the, Swarm Intelligence Symposium, Indianapolis, IN, pp. 26–33 (2003)
20. Liu, D.S., Tan, K.C., Goh, C.K., Ho, W.K.: A multiobjective memetic algorithm based on particle swarm optimization. IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics 37(1), 42–50 (2007)
21. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8(2), 173–195 (2000)
22. Schott, J.R.: Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization, M. S. Thesis, Dept. Aeronautics and Astronautics, Massachusetts Inst. Technol., Cambridge, MA (May 1995)
23. Van Veldhuizen, D.A.: Multiobjective Evolutionary Algorithms: Classifications, Analyzes, and New Innovation, Ph.D Dissertation, Dept. Elec. Comput. Eng., Air Force Inst. Technol., Wright-Patterson AFB, OH (May 1999)

Hybrid Behavioral-Based Multiobjective Space Trajectory Optimization

Massimiliano Vasile

Department of Aerospace Engineering, University of Glasgow, James Watt South Building, G12 8QQ, Glasgow, UK
m.vasile@aero.gla.ac.uk

In this chapter we present a hybridization of a stochastic based search approach for multi-objective optimization with a deterministic domain decomposition of the solution space. Prior to the presentation of the algorithm we introduce a general formulation of the optimization problem that is suitable to describe both single and multi-objective problems. The stochastic approach, based on behaviorism, combined with the decomposition of the solutions pace was tested on a set of standard multi-objective optimization problems and on a simple but representative case of space trajectory design.

1 Introduction

The design of a space mission steps through different phases of increasing complexity; generally, the first step is a mission feasibility study. In order to be successful, the feasibility study phase has to analyze, in a reasonably short time, a large number of different mission options. Each mission option requires the design of one or more trajectories that have to be optimal with respect to one or more criteria. In mathematical terms, the problem can be formulated as a search for multiple local minima, or as a multi-objective optimization problem.

In both cases, it is desirable to have a collection of several optimal solutions. Normally in literature, single objective and multi-objective optimization are treated as two distinct problems with different algorithms developed to address one or the other (see [1, 3, 4, 5, 6, 7, 8] for some examples of algorithms for global single objective optimization and [9, 10, 11, 12] for some examples of algorithms for multi-objective optimization). In most of the cases Evolutionary Algorithms seem to be the preferred method and many examples exist of their use to address both single objective and multi-objective problems in space trajectory design: Gage et al. has shown the effectiveness of genetic algorithms with niching technique compared to a simple grid search for the optimization of bi-impulsive transfers [13], Coverstone et al. used genetic algorithms for low-thrust trajectory design [15, 16], Gurfil et al. used niching genetic algorithms for the characterization of geocentric orbits [14], Vasile proposed a hybridization of evolutionary algorithms with SQP methods for the design of weak stability transfers [17] and an hybridisation with branch and bound for low-thrust trajectory design [18], and Dachwald et al. proposed the combination of a neurocontroller and of Evolutionary Algorithms for

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 231–253.
springerlink.com

the design of low-thrust trajectories[19]. More recently, an comparison of several global optimization methods applied to the optimization fo space trajectories showed that Differential Evolution outperforms GAs on some trajectory design problems [20, 21].

In general, all evolutionary-based approaches for global optimization implement some heuristic derived from nature. From the very basic evolutionary paradigms to the more complex behaviors of ant colonies or bird flocks, each one of these heuristics can be interpreted as basic behaviors (like reproduction, feeding or trail following) associated to individual agents. This chapter presents a generalization of this concept: a population of agents is endowed with a set of individualistic and social behaviors, in order to explore a virtual environment composed of the solution space. The combination of individualistic and social behaviors aims at an optimal balance between global search and local convergence (or exploration versus exploitation).

Furthermore, a unified formulation is proposed that can be applied to the solution of both multi-objective and single objective problems in which the aim is to find a set of optimal solutions, rather than a single one. In order to improve the exploration of the search space and to collect as many local minima as possible, the proposed metaheuristic was hybridized with a domain decomposition technique.

2 General Problem Formulation

The general problem both for single and multi-objective optimization is to find a set X, contained in a given domain D, of solutions $\mathbf{x}$ such that the property $P(\mathbf{x})$ is true for all $\mathbf{x} \in X \subseteq D$,

$$X = \{\mathbf{x} \in D \mid P(\mathbf{x})\} \tag{1}$$

where the domain D is a hyper-rectangle defined by the upper and lower bounds on the components of the vector $\mathbf{x}$,

$$D = \left\{x_i \mid x_i \in [b_i^l \; b_i^u] \subseteq \Re, \; i = 1, ..., n\right\} \tag{2}$$

All the solutions satisfying property P are here defined to be optimal with respect to P, or P-optimal, and X can be said to be a P-optimal set. Now, the property P might not identify a unique set, for example if P is Pareto optimality, X can collect all the points belonging to a local Pareto front. Therefore we can define a global optimal set X_{opt} such that all the elements of X_{opt} dominate the elements of any other X,

$$X_{opt} = \left\{\mathbf{x}^* \in D \mid P(\mathbf{x}^*) \wedge \forall \mathbf{x} \in X \Rightarrow \mathbf{x}^* \prec \mathbf{x}\right\} \tag{3}$$

where $\mathbf{x}^* \prec \mathbf{x}$ represents the dominance of the $\mathbf{x}^*$ solution over the $\mathbf{x}$ solution.

If we are looking for local minima, the property P is to be a local minimiser or a solution $\mathbf{x}^*$ can be said to dominate solution $\mathbf{x}$ if the associated value of the objective function $f(\mathbf{x}^*) < f(\mathbf{x})$. In this case X_{opt} would contain the global optimum or a set of global optima all with the same value of f.

In the case of multiple objective problems, given a set of solution vectors we can associate to each one of them a scalar dominance index I_d such that:

$$I_d(\mathbf{x}_j) = |\{i \mid i \wedge j \in N_p \wedge \mathbf{x}_i \prec \mathbf{x}_j\}| \tag{4}$$

where the symbol $|.|$ is used to denote the cardinality of a set and N_p is the set of the indices of all the given solution vectors. Here and in the following, a solution vector $\mathbf{x}_i$ is said to be dominating a solution vector $\mathbf{x}_j$ if the values of all the components of the objective vector $\mathbf{f}(\mathbf{x}_i)$ are lower than or equal to the values of all the components of the objective vector $\mathbf{f}(\mathbf{x}_j)$ and at least one component is strictly lower. In this case, for the $j-th$ solution, $P(\mathbf{x}_j)$ simply defines the property of being not-dominated by any other solution in the set N_p, thus:

$$X = \{\mathbf{x}_j \in D \mid I_d(\mathbf{x}_j) = 0\} \tag{5}$$

For constrained problems, the property P is to be optimal, either locally or Pareto, and feasible at the same time. The property P can be expressed through a single scalar value or through a set of values and relationships. It can be a bolean value, a real number or a fuzzy expression (e.g. bad, average, good).

3 A Behavioral Prospective

The search for a set of solutions can be broken down to a three steps process: collecting information, making decision, taking action. For a black-box problem the collection of information is generally performed by sampling the solution space. The aim in this respect is to minimize the number of samples required to find the desired solution or set of solutions. The decision making process consists of deciding what action to take at every step of the search, selecting who does what in the case of multiple entities and deciding when to stop the search. The action step consists of implementing the selected actions by the selected entity. The three steps are required to be automatic with minimum human intervention, which means that the decision to orient the search in one or another direction should not require the human judgment (e.g. restrict the search space, increase the number of samples in a specific region).

Let us assume that a virtual agent is endowed with the ability of collecting pieces of information, making decisions and implementing actions. The decision making process could involve a long term planning of actions a closed-loop control mechanism or some sort of action-selection process in response to stimuli. For example in particle swarm optimization (PSO)[6] the velocity of each particle i is computed with a close-loop control mechanism:

$$\mathbf{v}_{i+1} = w\mathbf{v}_i + \mathbf{u}_i \tag{6}$$

where w is a weight and given the random numbers r_1,r_2 and the weights a_1 and a_2 the control $\mathbf{u}_i$ has the form:

$$\mathbf{u}_i = a_1 r_1(\mathbf{x}_i - \mathbf{x}_{gi}) + a_2 r_2(\mathbf{x}_i - \mathbf{x}_{go}) \tag{7}$$

The search is continued till the decision to stop is taken. The control function requires a piece of information collected by the particle $\mathbf{x}_{gi}$ and one collected by another particle $\mathbf{x}_{go}$. In this case, the decision making process includes the selection of the particle in $\mathbf{x}_{go}$.

Let us assume that the virtual agent is equipped with a set of actions and an action-selection process. If more than one agent exists then some of the actions can be regarded

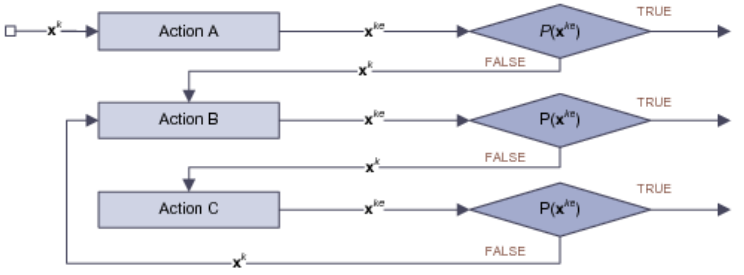


Fig. 1. Example of action selection mechanism

as individualistic, since involve only one agent at time, while others as social since require interaction among multiple agents. A collection of actions and the action selection process define a behavior. When at step k an agent $\mathbf{x}^k$ implements an action, it produces an outcome $\mathbf{x}^{k_e}$ according to:

$$\mathbf{x}^{k_e} = \mathbf{x}^k + \beta(\mathbf{x}^k, \mathbf{x}^{k-1}, \Pi_k, A_l) \tag{8}$$

where β is a function of the current state of the agent $\mathbf{x}^k$, the current state of the population Π_k, the past state of the agent $\mathbf{x}^{k-1}$ and the information about the past state of the population stored in an archive A_l. Note that β is not analytical but is an algorithm that selects an action, assigns a value and return a variation $\Delta\mathbf{x}^k$.

For example, for individualistic behaviors (see Fig. 1 and the next section), each agent can perform three types of actions, **A**, **B** and **C**. This general scheme accommodates two types of heuristics: Action **A** generates always the same outcome every time is performed once a solution vector **x** is given (e.g. inertia in PSO), while Actions **B** and **C** generate different values for the same **x** every time they are performed (e.g. mutation in EA[1]). These last two actions are repeated until an improvement is registered or a maximum number of attempts is reached. The index k_e is increased by one every time an action is performed, and every action makes use of the agent status **x**, the status of other agents and of the outcome of the proceeding actions.

4 MultiAgent Collaborative Search

A population of virtual agents (i.e., points within H)is deployed in the search space:

$$H = [a_1, b_1] \times \cdots \times [a_n, b_n]$$

Each agent is associated to a solution vector **x** and endowed with a set of basic actions forming a behavior. The entire population evolves, through a number of steps, toward the set X. At each step, the agents collect clues about the environment and implement actions according to an action selection mechanism. Some of the actions are devoted to acquiring new information (sampling the solution space), others to displacing agents, other actions are instead to exchange information among the agents. We implement a set

of actions, derived from PSO, EA and Differential Evolution (DE)[4] and very simple, basic action selection mechanisms. The general scheme both for a single agent and for a group is: select actions, implement actions, evaluate actions, make decision.

Given an agent $\mathbf{x} \in H$, a hyperrectangle

$$S^{\mathbf{x}} = S_1^{\mathbf{x}} \times \cdots S_n^{\mathbf{x}}$$

is associated to it, where each $S_i^{\mathbf{x}}$ is an interval centered at the corresponding component $\mathbf{x}[i]$ of the agent. The *size* of $S^{\mathbf{x}}$ is specified by the value $\rho(\mathbf{x})$: the i-th edge of $S^{\mathbf{x}}$ has length

$$2\rho(\mathbf{x}) \max\{b_i - \mathbf{x}[i], \mathbf{x}[i] - a_i).$$

As we will see, the ρ value associated to an agent is updated at each iteration according to some rule (see Section 4.2.1). The intersection $S^{\mathbf{x}} \cap H$ basically represents the local region around agent $\mathbf{x}$ which we want to explore. We also associate an *effort* value $s(\mathbf{x})$ to each agent $\mathbf{x}$, which specifies the amount of computational effort we want to dedicate to the exploration of $S^{\mathbf{x}}$. This value is updated at each iteration (see, again, Section 4.2.1).

The subdomain H is explored locally by acquiring information about the landscape within each region $S^{\mathbf{x}}$ and explored globally by evolving a population of agents which are also allowed to collaborate with each other. Moreover, an archive A_l of solutions over the domain H is maintained during the search. The archive is maintained in order to have a set of solutions for the problem at hand (see the discussion in the Introduction). The proposed approach, called Multiagent Collaborative Search (MACS), is outlined in the following, while the details will be specified in the following subsections.

Multiagent Collaborative Search

Step 0. Initialization. Generate an initial population of agents Π_0 within H through a Latin Hypercube (i.e., a non-collapsing design where points/agents are evenly spread even when projected along a single parameter axis; for a more detailed description and a justification of the use of Latin Hypercubes we refer, e.g., to [22]). A hyperrectangle $S^{\mathbf{x}_j^0}$ is associated to the j-th agent $\mathbf{x}_j^0 \in \Pi_0$. The initial *size* $\rho(\mathbf{x}_j^0)$ of each region $S^{\mathbf{x}_j^0}$ is fixed to 1 (i.e., the initial local region of each agent corresponds to the whole set H). The *effort* $s(\mathbf{x}_j^0)$ dedicated to agent $\mathbf{x}_j^0 \in \Pi_0$ is fixed to the same value $s_{\max}$ (equal to n in the computations) for all agents in Π_0. Set $k = 0$.

Step 1. Social Behavior. A set of social actions, specified by a *social behavior* are applied to the population Π_k. In particular two sets of social actions are implemented:

- Collaboration. The agents exchange information with each other. Each set of communication actions gives rise to new sampled points. Some of them identify the new location of a subset of the collaborating agents. See Section 4.1.1.
- Repulsion. If two or more agents are too crowded one or more are reallocated in the search space. See Section 4.1.2.

Step 2. Filtering. A *filter* partitions population Π_k into two subsets Π_k^{in}, the population within the filter, and Π_k^{out}, the population outside the filter. See Section 4.3.

Step 3. Individualistic Behavior. A set of individualistic actions, specified by an *individualistic behavior*, are applied to each agent $\mathbf{x} \in \Pi_k$.

- Local Exploration. These set of actions allows local exploration (within $S^{\mathbf{x}}$) of the region around the agent. They are repeatedly applied until either an improvement is observed or the number $s(\mathbf{x})$ of actions is reached. If an agent $\mathbf{x}$ generates an improvement, population Π_k is updated by replacing $\mathbf{x}$ with its improvement. See Section 4.2.
- Hyperrectangle and effort update. The size parameter ρ and the effort parameter s associated to each agent within the filter are updated according to some rule. See Section 4.2.1.

Step 4. Archive update. Apply filtering and update archive A_l (see Section 4.4).

Step 5. Stopping rule. A stopping rule is checked (see Section 4.5). If it is not satisfied, then set $k = k+1$ and go back to Step 1. If it is satisfied, then update the archive A_l by adding the current population, i.e., set $A_l = A_l \cup P_k$.

4.1 Social Behavior

Social behavior is defined through a set of communication actions (collaboration), a decision making process to select the outcome of the communication actions, a repulsion mechanism to limit crowding and increase diversity and a shared memory mechanism to exploit the social knowledge acquired during the search.

4.1.1 Collaboration

Collaboration defines operations through which information is exchanged between pairs of agents. Given a pair of agents $\mathbf{x}_1$ and $\mathbf{x}_2$, with $\mathbf{x}_1$ considered to be the better one according to property P, three different actions are defined. Two of them are defined by adding to $\mathbf{x}_1$ a step $\Delta\xi$ defined as follows

$$\Delta\xi = \alpha_2 r^t (\mathbf{x}_2 - \mathbf{x}_1) + \alpha_1 (\mathbf{x}_2 - \mathbf{x}_1),$$

and correspond to: *extrapolation* on the side of $\mathbf{x}_1$ ($\alpha_1 = 0$, $\alpha_2 = -1$, $t = 1$), with the further constraint that the result must belong to the domain H (i.e., if the step $\Delta\xi$ leads out of H, its size is reduced until we get back to H); *interpolation* ($\alpha_1 = 0, \alpha_2 = 1$), where a random point between $\mathbf{x}_1$ and $\mathbf{x}_2$ is sampled. In the latter case, the shape parameter t is defined as follows:

$$t = 0.75\frac{s(\mathbf{x}_1) - s(\mathbf{x}_2)}{s_{\max}} + 1.25$$

The rationale behind this definition is that we are favoring moves which are closer to the agent with a higher fitness value if the two agents have the same s value, while in the case where the agent with highest fitness value has a s value much lower than that of the other agent, we try to move away from it because a small s value indicates that improvements close to the agent are difficult to detect.

The third operation is the *recombination* operator, a *single-point crossover*, where, given the two agents: we randomly select a component i; split the two agents into two parts, one from component 1 to component i and the other from component $i+1$ to component n; and then we combine the two parts of each of the agents in order to

generate two new solutions. Note that the three operations give rise to four new samples, denoted by $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3, \mathbf{y}_4$.

The first of the two parent agents is selected at random in the worst half of the current population (from the point of view of the property P), while the second parent is selected at random from the whole population. Selecting the first parent from the best half generally reduces the diversity of the population and may cause premature convergence.

Note that here all the communication actions are selected and implemented sequentially. However, according to the general scheme mentioned above, a different action selection mechanism can adaptively choose the most appropriate subset of actions at every step. The decision making process is used to select which of the four samples will be used. Each pair of parent agents $\mathbf{x}_1$ and $\mathbf{x}_2$ generates four samples $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$ and $\mathbf{y}_4$. Then, a tournament, based on the property P, is started between the worst of the two parents and the best of the four samples. The winner of the tournament will be the new location in the solution space of the worst parent agent. When an agent is displaced no update of ρ and s is performed.

4.1.2 Repulsion

When the distance between two agents drops below a given threshold, a repulsion action is applied to the one with the worst P. More precisely, consider agent $\mathbf{x}_j$ and let

$$M_j = \{i \,:\, S^{\mathbf{x}_j} \cap S^{\mathbf{x}_i} \neq \emptyset\}$$

be the set of agents whose box has a nonempty intersection with the one of $\mathbf{x}_j$. Let $n_c(j)$ denote the cardinality of M_j. Then, for each $i \in M_j$ we check the following condition

$$w_c n_c(j) \rho(\mathbf{x}_j) > \rho_{ij},$$

where ρ_{ij} denotes the normalized distance[1] between $\mathbf{x}_i$ and $\mathbf{x}_j$ and w_c is a small positive parameter called *crowding factor*. If the condition is satisfied, then the worse between agents $\mathbf{x}_i$ and $\mathbf{x}_j$ is repelled (note that $w_c = 0$ corresponds to no repulsion). Repulsion is basically an interpolation between the agent to be repelled and one vertex of the current domain chosen at random. The idea behind repulsion is to avoid convergence of different agents to the same subregion with a consequent waste of computational effort.

4.1.3 Shared Memory

The archive A_l is used to direct the movements of those agents for which P is false. For all agents for which the property P is not true at step k the inertia component is recomputed as:

$$\Delta\xi = r(\mathbf{x}_{A_l} - \mathbf{x}^k) \tag{9}$$

where $\mathbf{x}_{A_l}$ is an element taken from the archive. The elements in the archive are ranked according to their relative distance or crowding factor. Then, every agent for which P is false picks the least crowded element $\mathbf{x}_{A_l}$ not already picked by any other agent.

[1] By normalized distance we mean the distance between the two agents once H has been transformed into the unit hypercube through the appropriate affine transformation.

4.2 Individualistic Behavior

The individualistic behavior is defined through a set of local exploration actions, an action selection mechanism, a decision making process to select the outcome of the exploration and an adaptive update of the resources and of the regions $S^{\mathbf{x}}$.

At every generation, a behavior β is used to generate the set of exploration actions. In particular, given agent j at generation k, denoted by $\mathbf{x}_j^k$, a behavior is a collection of displacement vectors $\Delta\xi$ generated by some function z_β:

$$\beta = \{\Delta\xi \ | \ \mathbf{x}_j^k + \Delta\xi \in H \text{ and } \Delta\xi = z_\beta(\mathbf{x}_j^k, \mathbf{x}_j^{k-1}, \mathbf{w}, \mathbf{r}, \Pi_k)\} \tag{10}$$

where z_β is a function of the current and past state $\mathbf{x}_j^k$ and $\mathbf{x}_j^{k-1}$ of agent j, of a set of weights $\mathbf{w}$, of a set of random numbers $\mathbf{r}$ and of the current population P_k. Every point $\mathbf{x}_j^k + \Delta\xi$ is called a *child* of agent j. In what follows we describe the different kinds of actions employed in this chapter.

Inertia. This action is performed at most once at each generation. If agent j has improved from generation $k-1$ to generation k, then we follow the direction of the improvement (possibly until we reach the border of the hyperrectangle associated to the agent), i.e., we perform the following step:

$$\Delta\xi = \bar{\lambda}(\mathbf{x}_j^k - \mathbf{x}_j^{k-1}) \tag{11}$$

where

$$\bar{\lambda} = \min\{1, \max\{\lambda \ : \ \mathbf{x}_j^k + \lambda(\mathbf{x}_j^k - \mathbf{x}_j^{k-1}) \in S^{\mathbf{x}_j}\}\}.$$

Follow-the-trail. This step is inspired by Differential Evolution (see, e.g., [8, 4]). It is defined as follows: let $\mathbf{x}_{i_1}^k, \mathbf{x}_{i_2}^k, \mathbf{x}_{i_3}^k$ be three randomly selected agents; then

$$\Delta\xi = \mathbf{x}_j^k - (\mathbf{x}_{i_1}^k + (\mathbf{x}_{i_3}^k - \mathbf{x}_{i_2}^k)) \tag{12}$$

(if the step leads out of $S^{\mathbf{x}_j}$, then its length is reduced until we reach the border of $S^{\mathbf{x}_j}$).

Random-Walk. Given the agent $\mathbf{x}$ and its associated hyperrectangle $S^{\mathbf{x}}$, four different kinds of mutation actions are performed all arising from the following displacement of a component i of the agent:

$$\Delta\xi_i = w_1 r^t(\ell_i - x_i) + (1 - w_1) r^t(u_i - x_i) \tag{13}$$

where ℓ_i and u_i are respectively the lower and upper limits of x_i within $S^{\mathbf{x}} \cap H$, r is a uniform random number in $[0,1]$, $w_1 = 1$ with some probability p_i and $w_1 = 0$ with probability $1 - p_i$, and $t \geq 0$ is a shape parameter ($t = 1$ corresponds to uniform sampling, while $t > 1$ favors more local moves). The four mutation actions are the following:

- all components i are perturbed according to (13) with $t = 1$ and $p_i = 0.5$;
- a component i is selected at random and perturbed according to (13) with $t = 1$ and $p_i = 0.5$;
- a component i is selected at random and perturbed according to (13) with $t = 0.5$ and $p_i = 0.5$;

- a component i is selected at random and randomly fixed either at its lower limit or its upper limit in the region $S^{\mathbf{x}} \cap H$, i.e., $t = 0$ and $p_i = 0.5$.

Linear blending. Once a mutation action on agent $\mathbf{x}$ has been performed, its result, denoted by $\mathbf{y}$, is further refined through blending procedures. Linear blending corresponds to the following displacement:

$$\Delta\xi = \alpha_2 r^t(\mathbf{y}-\mathbf{x}) + \alpha_1(\mathbf{y}-\mathbf{x}). \tag{14}$$

where $\alpha_1, \alpha_2 \in \{-1, 0, 1\}$, $r \in [0,1]$ is a random number, and t a shaping parameter which controls the magnitude of the displacement. Here we use the parameter values $\alpha_1 = 0$, $\alpha_2 = -1$, $t = 1$, which corresponds to *extrapolation* on the side of $\mathbf{x}$, and $\alpha_1 = \alpha_2 = 1$, $t = 1$, which corresponds to *extrapolation* on the side of $\mathbf{y}$. If the displacement defined by an extrapolation action is too large, i.e., the resulting point is outside the hyperrectangle associated with the current agent, then it is reduced until the resulting point is within the hyperrectangle.

Quadratic blending. The outcome of the linear blending can be used to construct a second order local model of the fitness function. We can define a second order blending operator that generates a displacement using the agent $\mathbf{x}$, the perturbation $\mathbf{y}$ obtained by mutation, and the new point $\mathbf{z}$ generated by the linear blending operator. A second order one-dimensional model of the fitness function along the line with direction $\mathbf{x}-\mathbf{z}$ is obtained by fitting the fitness values in the three points $\mathbf{x}$, $\mathbf{y}$ and $\mathbf{z}$. Then, the new point is the minimum of the second-order model along the intersection of the line with the hyperrectangle associated with the agent.

As already pointed out, the inertia action is performed at most once. All the other actions are cyclically performed until either an improvement is observed or the number $s(\mathbf{x}_j^k)$ of actions is reached. Note that in each cycle only one of the four mutation actions is performed in turn.

4.2.1 Size and Effort Update

Given an agent $\mathbf{x}_j^k \in \Pi_k^{in}$, its size parameter $\rho(\mathbf{x}_j^k)$, defining the hyperrectangle $S^{\mathbf{x}_j^k}$ centered at $\mathbf{x}_j^k$, and its effort parameter $s(\mathbf{x}_j^k)$, giving the maximum number of actions applied to it, are updated at each generation. Both are reduced or enlarged depending on whether an improvement has been observed or not in the previous generation.

If $\mathbf{x}_j^{k+1} \neq \mathbf{x}_j^k$, i.e., an improvement has been observed for agent j at iteration k, then the effort is updated according to the following formula:

$$s(\mathbf{x}_j^{k+1}) = \max\{s(\mathbf{x}_j^k) + 1, s_{\max}\},$$

i.e., it is increased by 1, unless the maximum allowed number of actions has been already reached (recall that in the computations $s_{\max}$ has been fixed to the dimension n of the problem). Basically, we are increasing the effort if the agent is able to improve. In the same case the size is increased by the following formula:

$$\rho(\mathbf{x}_j^{k+1}) = \max\{\rho(\mathbf{x}_j^k)\ln(e + rank(\mathbf{x}_j^{k+1})), 1\}$$

where $rank(\mathbf{x}_j^{k+1})$ is the ranking of the agent $\mathbf{x}_j^{k+1}$ within the population Π_k (the best individual has rank equal to 1, the second best equal to 2, and so on). Basically, the worse the ranking of an individual, the greater the possible increase of the radius will be. The increase is limited from above by 1 (when $\rho = 1$ the local region around the agent to be explored is equal to the whole domain H). The idea is that for low ranked individuals it makes sense to look for larger improvements and then to try to find a better point in larger regions making the search more global.

If no improvement is observed, then the effort is updated according to the following formula:

$$s(\mathbf{x}_j^{k+1}) = \max\{s(\mathbf{x}_j^k) - 1, 1\},$$

i.e., it is decreased by 1, unless the minimum allowed number of actions has been already reached.

In the same case the size is reduced according to the following rule. Let $\rho_{min}(\mathbf{x}_j^k)$ be the smallest possible reduction of the size parameter such that the child $\mathbf{y}^*$ of $\mathbf{x}_j^k$ with best fitness value is still contained in the hyperrectangle. Then:

$$\rho(\mathbf{x}_j^{k+1}) = \begin{cases} \rho_{min}(\mathbf{x}_j^k) & \text{if } \rho_{min}(\mathbf{x}_j^k) \geq 0.5\rho(\mathbf{x}_j^k) \\ 0.5\rho(\mathbf{x}_j^k) & \text{otherwise} \end{cases}$$

i.e., the size parameter is reduced to $\rho_{min}(\mathbf{x}_j^k)$ unless this is smaller than $0.5\rho(\mathbf{x}_j^k)$, in which case we only halve the size parameter.

4.3 Filtering

Given a population Π_k, a filter simply subdivides the population into two parts, Π_k^{in} and Π_k^{out}. Π_k^{in} contains the best members of the population Π_k, i.e., those with the best fitness values, while Π_k^{out} contains all the other individuals in Π_k. The main difference between agents inside and outside the filter is that on agents outside the filter, only mutation actions (see equation (13) below) over the whole subdomain H are performed (for each agent outside the filter the number of these mutation actions is a random one between 1 and the size of Π_k^{out}), while also other actions, allowing a deeper local exploration, are performed on agents inside the filter (see the following Section 4.2). Moreover, values ρ and s are only updated for agents in Π_k^{in} (see the following Section 4.2.1). In the case an agent outside the filter at iteration k enters the filter at iteration $k+1$, its ρ and s values are initialized as specified in Step 0.

4.4 Archive Update

Let ρ_{tol} be a small threshold value, $\mathbf{x}_i$ be an agent whose size parameter is below the threshold value, i.e., $\rho(\mathbf{x}_i) < \rho_{tol}$ and L_i be the set of agents whose normalized distance from $\mathbf{x}_i$ is below the threshold ρ_{tol} (including $\mathbf{x}_i$ itself). If agent $\mathbf{x}_i$ is the best one in L_i (from the point of view of P), then all agents in L_i are randomly regenerated within the current domain H and $\mathbf{x}_i$ is inserted in archive A_l, while if it is not, only agent $\mathbf{x}_i$ is randomly regenerated within H. At termination of the MACS algorithm we insert the whole final population into the archive.

4.5 Stopping Rule

The stopping rule is quite simple: the search within a subdomain is stopped when a prefixed number N_e of function evaluations is reached.

4.6 Definition of P for Multiobjective Optimization

Although the problem formulation through the definition of P is general and applicable to both single objective and multiple objective optimization problems, either constrained or not, the actual property is substantially different depending on the type of problem.

For box constrained multi-objective problems the property P can be defined by the value of the scalar dominance index I_d, thus:

$$I_d(\mathbf{x}) > I_d(\mathbf{x}_{k_e}) + \varepsilon \Rightarrow P(\mathbf{x}_{k_e}) = true \tag{15}$$

where ε is now the minimum expected improvement in the computation of the dominance. Note that this easily accommodates the concept of ε dominance.

Now, when multiple outcomes with the same dominance index are generated by either social or individualistic actions, the one that corresponds to the longest vector difference in the criteria space with respect to $\mathbf{x}$ is considered. Note that in many situations the action selection scheme in Fig. 1 generates a number of solutions that are dominated by the agent $\mathbf{x}$. Many of them can have the same dominance value; therefore in order to rank them, we use the modified dominance index:

$$I_d(\mathbf{x}) = \left|\left\{j \mid f_j(\mathbf{x}_{k_e}) = f_j(\mathbf{x})\right\}\right| \kappa + \left|\left\{j \mid f_j(\mathbf{x}_{k_e}) > f_j(\mathbf{x})\right\}\right| \tag{16}$$

where κ is equal to one if there is at least one component of $\mathbf{f}(\mathbf{x}) = [f_1, f_2, ..., f_{N_f}]^T$ which is better than the corresponding component of $\mathbf{f}(\mathbf{x}_{k_e})$, and is equal to zero otherwise.

Now, if for the ${k_e}^{th}$ outcome, the dominance index in Eq. 16 is not zero but is lower than the number of components of the objective vector, then the agent $\mathbf{x}$ is only partially dominating the ${k_e}^{th}$ outcome. Among all the partially dominated outcome with the same dominance index we chose the one that satisfies the condition:

$$\min_{k_e} \left\langle \left(\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{x}_{k_e})\right), \mathbf{e} \right\rangle \tag{17}$$

where $\mathbf{e}$ is the unit vector of dimension N_f, $\mathbf{e} = \frac{[1,1,1,...,1]^T}{\sqrt{N_f}}$, and N_f is the number of objective functions.

Since the partially dominated outcomes of one agent could dominate other agents or the outcomes of other agents at the end of every evolution cycle all the outcomes are added to the archive. Then, the dominance index in Eq. 4 is computed for all the elements in A_l and only the non-dominated ones are preserved.

4.7 Hybridization with Domain Decomposition

The Multiagent Collaborative Search described in the previous section is a stochastic process. In order to improve its robustness (repeatability of the result) and increase the

exhaustiveness of the search, MACS is combined with a deterministic domain decomposition technique. The search space D, is a hyperrectangle and the subdomains into which it is subdivided are also hyperrectangles. The stochastic algorithm searches on the subdomains in order to evaluate them. In this section we will give the details of the deterministic method.

Below we give the description of a generic branching procedure for GO problems.

BRANCHING PROCEDURE
Step 0. Initialization Let $\mathscr{F} = \{D\}$.
Step 1. Node selection Let θ be a function which associates a value to each node $H \in \mathscr{F}$. Then, select a node $\overline{H} \in \mathscr{F}$ such that

$$\overline{H} \in \arg\min_{H \in \mathscr{F}} \theta(H), \tag{18}$$

Step 2. Evaluation Evaluate the selected node $\overline{H}$ through some procedure.
Step 3. Node branching Subdivide $\overline{H}$ into η nodes $H_i,\ i = 1, \ldots, \eta$, for some integer $\eta \geq 2$, and update $\mathscr{F}$ as follows:

$$\mathscr{F} = (\mathscr{F} \setminus \{\overline{H}\}) \cup \{H_1, \ldots, H_\eta\}.$$

Step 4. Node deletion Delete nodes from $\mathscr{F}$ according to some rule.
Step 5. Stopping rule If $\mathscr{F} = \emptyset$, then STOP. Otherwise, go back to Step 1.

Note that in the scheme above each node corresponds to a subdomain, and in what follows the two terms will be used as synonymous. Such scheme is quite typical for branch-and-bound methods. For these methods θ delivers a lower bound for each subdomain; each node is evaluated by evaluating feasible points within the corresponding subdomain (if any) and possibly updating the upper bound; node branching can be performed in several ways; node deletion is done through standard fathoming rules. However, what is missing in our context is an easy way to obtain bounds. Therefore, while we retain the branching structure, we need some other ways to define a function θ and to evaluate, branch and delete nodes. All this will be specified in the following subsections.

4.7.1 Node Evaluation

The evaluation of a subdomain H is done by running the MultiAgent Collaborative Search (MACS) algorithm within H. The MACS algorithm explores the subdomain H and stores in a local archive A_l all the promising points in H. The local archive A_l is then compared to the global archive A_g containing all the points in the search space for which P is true. The points in $E_v(H) = (A_l \cap A_g) \cap H$ represent the evaluation of the subdomain.

4.7.2 Node Branching

First we recall that each subdomain is a hyperrectangle. Branching is done through the standard bisection method: the (relative) largest edge of the domain to be subdivided is selected and two new subdomains (i.e, $\eta = 2$) are obtained by splitting with respect to a point midpoint $\tilde{\mathbf{x}}$. More formally, let

$$H = [a_1, b_1] \times \cdots \times [a_n, b_n]$$

be the domain to be subdivided. Let

$$j \in \arg\max_{i=1,\ldots,n} \frac{b_i - a_i}{D_i - d_i},$$

where d_i and D_i denote respectively the lower and upper bounds for variable x_i in the original domain D. We can now subdivide the interval $b_i - a_i$ into a number n_j of subintervals and look for the one that contains the majority of the points in H. Now if the subinterval has boundaries b_{ik} and a_{ik} with $k = 1, \ldots, n_j$, the cutting point $\tilde{x}_j$ is defined as:

$$\tilde{x}_j = \begin{cases} b_{ik} \text{if } (b_i - b_{ik}) > (a_{ik} - a_i) \\ a_{ik} \text{if } (b_i - b_{ik}) \leq (a_{ik} - a_i) \end{cases}$$

Then, we define the two new subdomains

$$H_1 = [a_1, b_1] \times \cdots \times [a_j, \tilde{x}_j] \times \cdots \times [a_n, b_n],$$

$$H_2 = [a_1, b_1] \times \cdots \times [\tilde{x}_j, b_j] \times \cdots \times [a_n, b_n].$$

4.7.3 The Game of Exploration

The selection of the subdomain H on which to perform a new search with MACS depends on the outcome of a simple game between two players: explorer and exploiter.

Before presenting the game and selection process, we need to introduce two other functions ω and φ. Let H be a given subdomain and $\tilde{H}$ be its father. Function ω is defined as follows for H:

$$\omega(H) = \frac{\max\{N(H), 1\}}{N} \frac{\ell(D)}{\ell(H)} \tag{19}$$

where $\ell(\cdot)$ denotes the geometric mean of the edge lengths of an n-dimensional hyperrectangle, N is the number of points in $E_v(\tilde{H})$, obtained through the evaluation of the father node $\tilde{H}$ by the MACS algorithm, and $N(H)$ is equal to the number of points in $E_v(\tilde{H})$ which also belong to H, i.e., $N(H) = \mid E_v(\tilde{H}) \cap H \mid$ (for the root node D we simply set $N(D) = N$). We also remark that in (19) the ratio between geometric means of the edge lengths can also be viewed as the nth root of a ratio between volumes:

$$\frac{\ell(D)}{\ell(H)} = \left(\frac{Vol(D)}{Vol(H)} \right)^{\frac{1}{n}}.$$

Then, small values for ω are obtained for subdomains with small $N(H)$ and large volume, i.e., for subdomains with a low density of observed points. Function φ is defined as follows:

$$\varphi(H) = |\{i | I(\mathbf{x}_i) = 0\}| \tag{20}$$

Now, the player *explore* always tries to maximize the volume of the explored space and therefore selects the subdomain with the lowest value of ω. The player *exploiter*

always tries to maximize convergence and thus selects always the subdomain with the highest value of φ. If both players play explore (explore-explore strategy), then the algorithm will select the subdomain with the smallest ω for further exploration, if both players play converge (converge-converge strategy), then the algorithm will select the subdomain with the highest φ. If *explorer* plays explore first and *exploiter* plays converge (explore-converge strategy) then, the algorithm will rank the subdomain according to ω and then, among the first n_ω of them will take the one with the largest number of elements in A_l. Vice versa, if *exploiter* plays first (converge-explore strategy), the algorithm will rank the subdomains according to φ and among the first n_φ will select the one with the lowest ω. The selection of the strategy to play depends on the outcome of the game.

If both players play converge then exploiter gets a reward only if it finds an improvement while explorer gets no reward whatsoever. Since we are interested in collecting as many different elements of a set as possible, this strategy is not convenient for any of the two players. If both players play explore then the explorer gets a reward while the exploiter gets half of the reward of the explorer only if an improvement is registered. This strategy is convenient if no improvements are registered with any other strategy or if no information is available. Thus, it is the first strategy that both players play at the beginning. The outcome of the exploration of a subdomain could be: a) a number of points belonging to the set X higher than the one already available, b) a number of points belonging to the set X lower or equal to the number already available, c) no points belonging to the set X. In the last case the algorithm plays explore-explore, in case a) though the subdomain looks promising the higher number of points could suggest an over-exploitation of the subdomain and the algorithm then plays explore-converge. Finally, in case b) the algorithm plays converge-explore.

4.8 Discussion

The hybrid behavioral-based approach presented in the previous sections is one possible implementation of the concept expressed by Eq.8. In particular we used a very simple action selection mechanism for both social and individualistic actions. More sophisticated mechanisms may allow for a reduction of the number of function evaluations or could include a learning mechanism. Furthermore, in the present implementation there is a limited use of the past history of the search process. The behavior Eq.8 depends on the archive A_l, which works as a repository of the social knowledge, and on the state of the agent at step $k-1$, therefore only a partial history is preserved and used.

4.9 Preliminary Optimization Test Cases

The proposed optimization approach, combining MACS with deterministic domain decomposition, was implemented in a software code in Matlab called EPIC, and tested on a number of standard problems, well known in literature. In a previous work [24, 25], EPIC was tested on single objective optimization problems related to space trajectory design, showing good performances.

Here we initially used a set of test functionsthat can be found in [10, 9]. In the next section EPIC will be tested on a typical space trajectory optimization problem.

Table 1. Multiobjective test functions

Scha	$f_2 = (x-5)^2 \,; f_1 = \begin{cases} -x & if\ x \le 1 \\ -2+x & if\ 1 < x < 3 \\ 4-x & if\ 3 < x \le 4 \\ -4+x & if\ x > 4 \end{cases}$	$x \in [-5,10]$
Deb	$f_1 = x_1$ $f_2 = (1+10x_2)\left[1-\left(\frac{x_1}{1+10x_2}\right)^{\alpha} - \frac{x_1}{1+10x_2} sin(2\pi q x_1)\right]$	$x_1, x_2 \in [0,1]$ $\alpha = 2;.\ q = 4$
T4	$g = 1+10(n-1)+\sum_{i=2}^{n}[x_i^2 - 10cos(2\pi q x_i)];$ $h = 1-\sqrt{\frac{f_1}{g}}$ $f_1 = x_1; f_2 = gh$	$x_1 \in [0,1];$ $x_i \in [-5,5];$ $i = 2,\ldots,n$

The test case, *T4*, is commonly recognized as one of the most challenging problems since it has 21^9 different local Pareto fronts of which only one corresponds to the global Pareto-optimal front. In this case the exploration capabilities of each single agents are enough to locate the correct front with a very limited effort. In fact even with just five agents it was possible to reconstruct (see Fig. 2b) the correct Pareto front 20 times over 20 different runs. The total number of function evaluations was fixed to 20000 for each of the runs, though already after 10000 function evaluations EPIC was always able to locate the global front (see Fig. 2a).

Despite the small number of agents the sampled points of the Pareto are quite well distributed with just few and limited interruptions. The use of a limited number of agents instead of a large population is related also to the complexity of the algorithm. In fact the complexity of the procedure for the management of the global archive is of order $n_A(n_p+n_A)$, where n_A is the archive size and n_p is the population size, while the complexity of the exploration-perception mechanism is of order $n_p(n+n_p)$, therefore, even

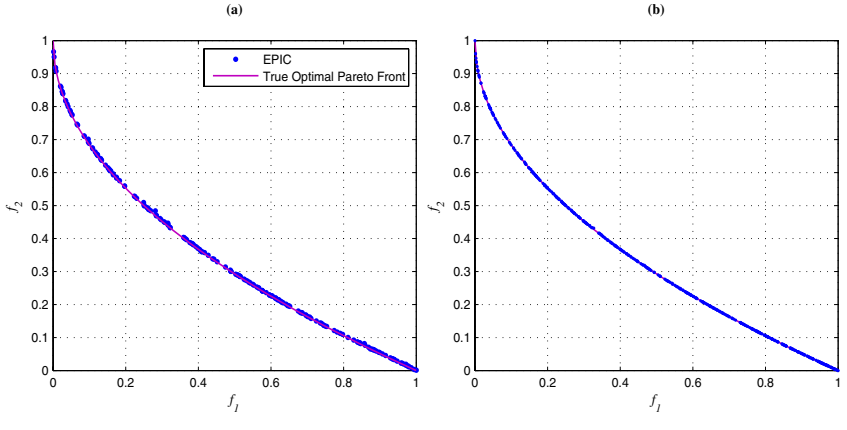


Fig. 2. Pareto front for the test case *T4*: a) 10000 function evaluations, b) 20000 function evaluations

Table 2. Comparison of the average Euclidian distances between 500 uniformly space points on the optimal Pareto front for various optimization algorithms

Approach	T4	Scha	Deb
EPIC	1.542e-3 (5.19e-4)	5.4179e-4 (1.7e-4)	1.4567e-4 (3.61e-4)
NSGA-II	0.513053 (0.118460)	0.002536 (0.000138)	0.001594 (0.000122)
PAES	0.854816 (0.527238)	0.002881 (0.00213)	0.070003 (0.158081)
MOPSO	0.0011611 (0.0007205)	0.002057 (0.000286)	0.00147396 (0.00020178)

if the algorithm is overall polynomial in population dimension, the computational cost would increase quadratically with the number of agents.

As an additional proof of the effectiveness of MACS, we compare the average Euclidean distance of 500 uniformly spaced points on the true optimal Pareto front from an equal number of points belonging to the solution found by EPIC, NSGA-II, PAES and MOPSO (see Table 2).

5 Application to Space Trajectory Design

In this section we present an apparently very simple example of space trajectory optimization. It is a two impulse transfer from the Earth to the asteroid Apophis. As it often happens, the goal is to minimize the propellant consumption and the time of flight. The cost of the mission, in fact, increases proportionally to both quantities.

The propellant consumption is a function of the velocity change, or Δv[23], required to depart from the Earth and to rendezvous with Apophis. Both the Earth and Apophis are point masses, with the only source of gravity attraction being the Sun. Therefore, the spacecraft is assumed to be initially at the Earth, flying along its orbit. The first velocity change, or Δv_1, is used to leave the orbit of the Earth and put the spacecraft into a transfer orbit to Apophis. The second change in velocity, or Δv_2, is then used to inject the spacecraft into Apophis' orbit.

The two Δv's are a function of the positions of the Earth and Apophis at the time of departure t_0 and at the time of arrival $t_f = t_0 + T$, where T is the time of flight. The contour lines of the sum of the two Δv is represented in Fig. 3 for $t_0 \in [3675, 10500]^T$ MJD2000 and $T \in [50, 900]$days.

As can be seen in the specified solution space D there is a large number of local minima. Each minimum has a different value but some of them are nested, very close to each other with similar values. For each local minimum, there can be a different front of locally Pareto optimal solutions. The global Pareto front should contain the best transfer with minimum total Δv and the fastest transfer with minimum TOF.

The best known approximation of the global Pareto front is represented in Fig. 4. It is a disjoint front corresponding to two basins of attraction of two minima as can be seen in Fig. 5. The lower front is made of solutions with a very low transfer time, the upper front, instead, is made of solutions with a much longer transfer time but a total Δv similar to the one of the solutions belonging to the lower front.

Besides containing local minima with similar Δv, the two basins of attraction present similar values of the first objective function. Converging to the upper front is therefore

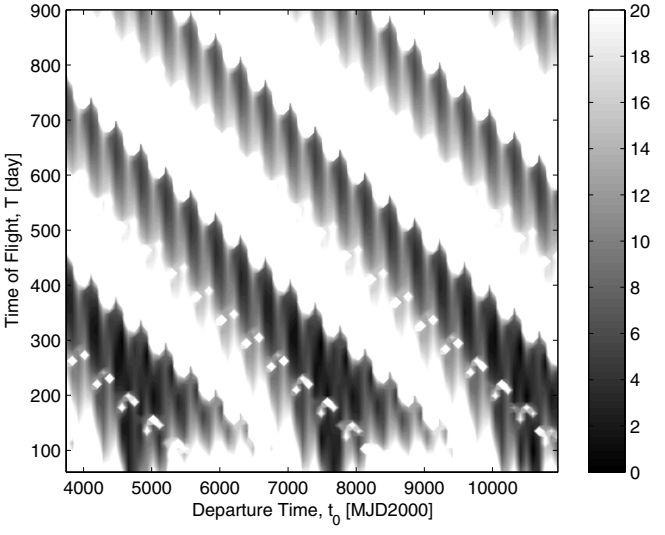


Fig. 3. Earth-Apophis search space

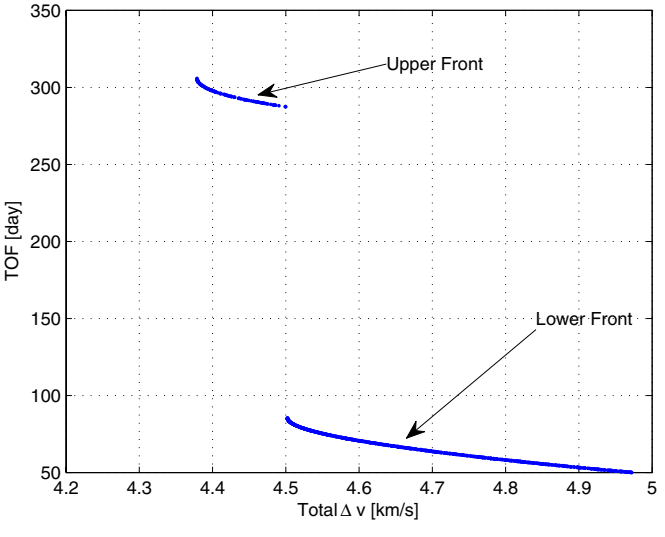


Fig. 4. Earth-Apophis transfer: dual front

quite a challenge since the lower front has a significantly lower value of the second objective function. It is only when the optimizer converges to the a vicinity of the local minimum of the upper front that the latter becomes not dominated by the lower front. The upper front contains the global minimum with a total $\Delta v = 4.3786$ k/s while the lower front contains only a local minimum.

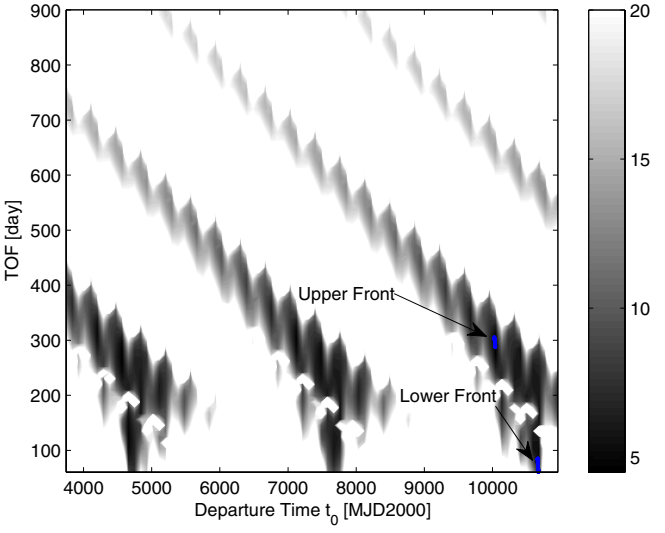


Fig. 5. Earth-Apophis transfer: distribution of the solutions in the search space

Table 3. Comparison of the metrics M_1 and M_2 ont he Earth-Apophis case

Approach	Metric	3000	6000	9000
EPIC	M_1	4% (39.00)	38% (22.28)	51% (17.53)
	M_2	1.89 (5.9)	1.66 (5.14)	0.66 (3.22)
NSGA-II	M_1	10% (26.51)	19% (27.54)	24% (25.63)
	M_2	20.96 (27.78)	11.99 (16.20)	10.12 (12.82)

In order to test the multi-objective optimizers with this simple but typical space trajectory design problem, we define two metrics:

$$M_1 = \frac{1}{M_p}\sum_{i=1}^{M_p}\min_{j\in N_p} 100\left\|\frac{\mathbf{f}_j-\mathbf{f}_i}{\mathbf{f}_i}\right\| \tag{21}$$

$$M_2 = \frac{1}{N_p}\sum_{i=1}^{N_p}\min_{j\in M_p} 100\left\|\frac{\mathbf{f}_j-\mathbf{f}_i}{\mathbf{f}_j}\right\| \tag{22}$$

Although similar, the two metrics are measuring two different things: M_1 is the sum, over all the elements in the global Pareto front, of the minimum distance of all the elements in the Pareto front N_p from the the i^th element in the global Pareto front. M_2, instead, is the sum, over all the elements in the Pareto front N_p, of the minimum distance of the elements in the global Pareto front from the i^th element in the Pareto front N_p.

Therefore, if N_p is only a partial representation of the global Pareto front but is a very accurate partial representation, then metrics M_1 would give a high value and metrics M_2

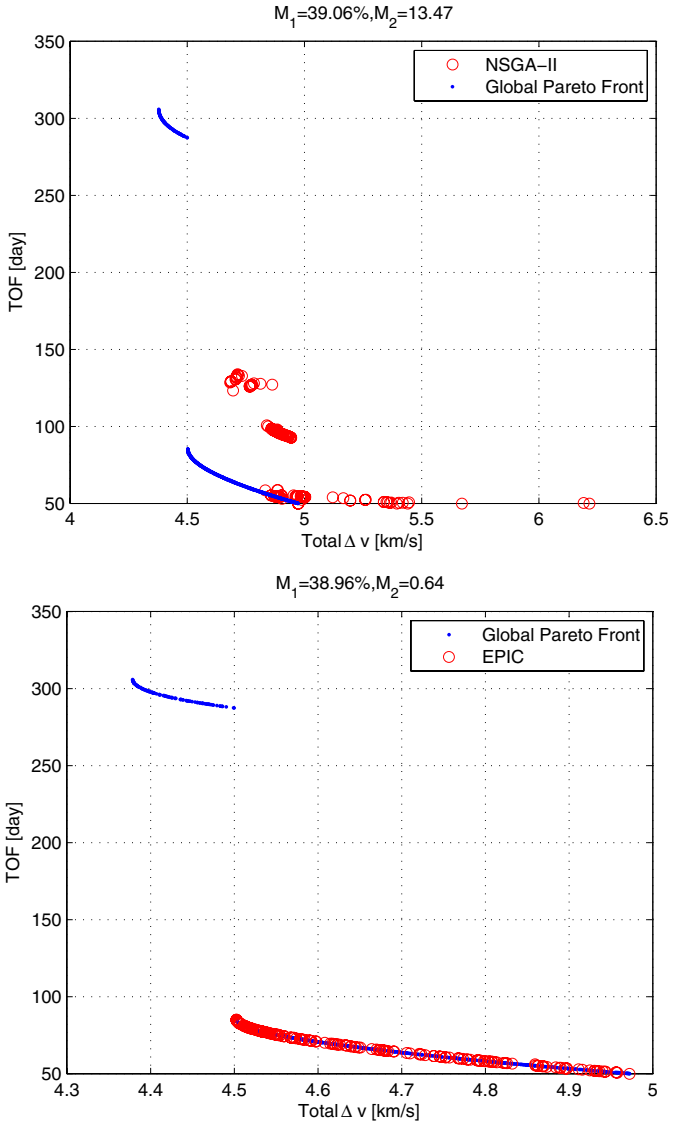


Fig. 6. Earth-Apophis transfer: comparison of two not-converged runs

a low value. If both metrics are high then the Pareto front N_p is partial and poorly accurate. In Table 3 we represented metrics M_1 and metrics M_2, in brackets, for an increasing number of function evaluations and for two different optimizers.

We compared EPIC against the optimization algorithm that displayed the best performances on this case, NSGA-II. NSGA-II was run several times with crossover probability ranging from 0.5 to 0.9 in order to tune the main parameters, in particular we

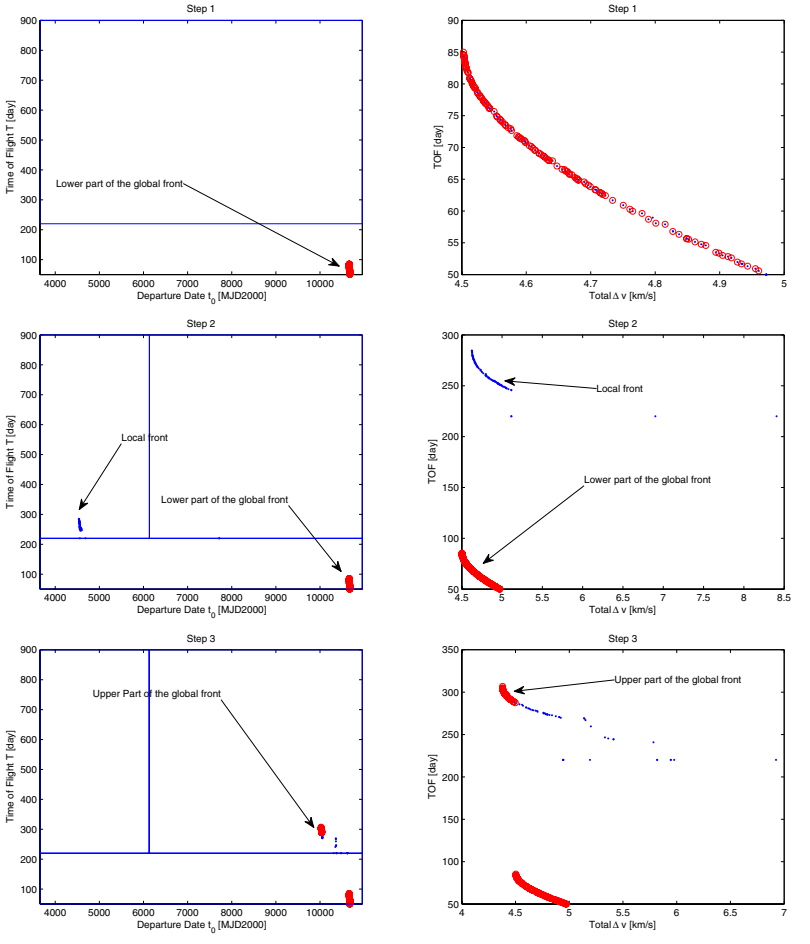


Fig. 7. Earth-Apophis transfer: domain decomposition process. The red circles are the non-dominated solutions at each step while the blues dots are the whole set of solutions3.

performed several tests to find a good population size. The results in Table 3 were the best obtained over all the runs. For 3000 function evaluations we used a population of 200 individuals while for the 6000 evaluations and the 9000 evaluations test we used a population of 300 individuals since it was returning better results.

On the other hand EPIC was run with a very small population of 10 agents, with a filter size of 5 agents. For, 3000 evaluations we did not use the domain decomposition. For 6000 evaluations we used a decomposition in 2 subintervals. For 9000 evaluations, we tested a 3 subdomain decomposition and a 2 subdomain decomposition. In both cases the first cut is always along the TOF coordinate.

For each number of function evaluations we performed 100 independent runs. The table reports the percentage of times the metric M_1 is below 2%, and in brackets the average value of M_1 over the 100 runs. It should be noted that a value of M_1 larger than 2% up to 10% does not necessary correspond to a fully unsuccessful run. The 2% tolerance, on the other hand, guarantees that the algorithm was able to identify both parts of the Pareto front with a good distribution of the points.

In the Table 3 we also reported the average value and the standard deviation (in brackets) of metric M_2. This second metric measures the accuracy of the convergence to even a portion of the whole Pareto front.

As can be seen for a low number of function evaluations, NSGA-II performs better than EPIC, though EPIC achieves a better value of M_2, which means a better local convergence on average. Conversely, when NSGA-II is not converging to the global front, is converging to a local front, while when EPIC is not converging to the whole global front is converging to a portion of it. Figs. 6 are showing two typical cases in which the metric M_1 is over 30% for both the optimization algorithms. In both cases the number of function evaluations is 3000.

Fig. 7 shows an example of the domain decomposition process. At step 1 MACS is run on the entire search space D and in this example identifies only the lower part of the global front. The search space is then partitioned in two subdomains and MACS is run on the unexplored one. The second step leads to the identification of a local front. The third step explores the unexplored subdomain and identifies the upper part of the global front. At each step MACS was run for 3000 function evaluations.

For a higher number of function evaluations, NSGA-II progressively increases the number of successes, though the accuracy remains lower than for EPIC. The large population of NSGA-II, in fact, samples the solution space better than the small population of EPIC. On the other hand the decomposition of the solution space allows EPIC to increase the exploration even with a small number of agents. This is demonstrated by the number of successful runs which is more than double than the one of NSGA-II.

6 Conclusions

In this chapter we presented a hybrid behavioral-based search algorithm for multiobjective optimization problems. We showed its effectiveness on a set of standard problems and in particular on a space trajectory design problem. The latter, though very simple, well illustrates some typical difficulties in the use of global methods for the design of space trajectories.

Though some of them, like NSGA-II, perform statistically well, still on a small number of trials the result could be only a partial reconstruction of the full Pareto front or a full but inaccurate reconstruction of it. The proposed hybridization increases the robustness (i.e. repeatability of the result) and the convergence accuracy at the same time. Even the stochastic part of the algorithm, based on a multiagent system, performs better compared to known optimizer. This is mainly due to good mixture of actions performing both local and global search and to the adaptivity of the search.

References

1. Chipperfield, A.J., Fleming, P.J., Pohlheim, H., Fonseca, C.M.: Genetic Algorithm Toolbox User's Guide, ACSE Research Report No. 512, University of Sheffield (1994)
2. Horst, R., Tuy, H.: Global optimization: deterministic approaches, 3rd edn. Springer, Berlin (1996)
3. Stephens, C.P., Baritompa, W.: Global optimization requires global information. Journal of Optmization Theory and Applications 96, 575–588 (1998)
4. Storn, R., Price, K.: Differential Evolution – a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11(4), 341–359 (1997)
5. Torn, A., Zilinskas, A.: Global Optimization. Springer, Berlin (1987)
6. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, pp. 1942–1948 (1995)
7. Perttunen, C.D., Stuckman, B.E.: Lipschitzian Optimization Without the Lipschitz Constant. JOTA 79(1), 157–181 (1993)
8. Price, K.V., Storn, R.M.: Jouni A. Lampinen. Differential Evolution: A Practical Approach to Global Optimization, 1st edn. Springer, Heidelberg (December 22, 2005)
9. Deb, K., Pratap, A., Meyarivan, T.: Fast elitist multi-objective genetic algorithm: NGA-II. KanGAL Report No. 200001 (2000)
10. Deb, K., Pratap, A., Meyarivan, T.: Constrained test problems for multi-objective evolutionary optimization. KanGAL Report No. 200002 (2002)
11. Sierra, M.R., Coello, C.A.C.: A Study of Techniques to Improve the Efficiency of a Multi-Objective Particle Swarm Optimizer. Evolutionary Computation in Dynamic and Uncertain Environments, 269–296 (2007)
12. Coello, C.A.C., Lamont, G., Van Veldhuizen, D.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007)
13. Gage, P.J., Braun, R.D., Kroo, I.M.: Interplanetary trajectory optimisation using a genetic algorithm. Journal of the Astronautical Sciences 43(1), 59–75 (1995)
14. Gurfil, P., Kasdin, N.J.: Niching genetic algorithms-based characterization of geocentric orbits in the 3D elliptic restricted three-body problem. Computer Methods in Applied Mechanics and Engineering 191(49-50), 5673–5696 (2002)
15. Hartmann, J.W., Coverstone-Carrol, V.L., Williams, S.N.: Optimal Interplanetary Spacecraft Trajectories via Pareto Genetic Algorithm. Journal of the Astronautical Sciences 46(3) (1998)
16. Rauwolf, G., Coverstone-Carroll, V.: Near-optimal low-thrust orbit transfers generated by a genetic algorithm. Journal of Spacecraft and Rockets 33(6), 859–862 (1996)
17. Vasile, M.: Combining Evolution Programs and Gradient Methods for WSB Transfer Optimisation. In: Operational Research in Space & Air, vol. 79, Book Series in Applied Optimization Kluwer Academy Press (2003) ISBN 1-4020-1218-7
18. De Pascale, P., Vasile, M.: Preliminary Design of Low-Thrust Multiple Gravity Assist Trajectories. Journal of Spacecraft and Rockets 43(5), 1065–1076 (2006)
19. Dachwald, B.: Optimization of interplanetary solar sailcraft trajectories using evolutionary neurocontrol. Journal of Guidance, and Dynamics (January/ February 2004)
20. Myatt, D.R., Becerra, V.M., Nasuto, S.J., Bishop, J.M.: Advanced Global Optimization Tools for Mission Analysis and Design. Final Rept. ESA Ariadna ITT AO4532/18138/04/NL/MV, Call03/4101 (2004)
21. Di Lizia, P., Radice, G.: Advanced Global Optimisation Tools for Mission Analysis and Design. European Space Agency, the Advanced Concepts Team, Ariadna Final Report 03-4101b (2004)

22. Dam, R.E., van Husslage, B.G.M., den Hertog, D., Melissen, J.B.M.: Maximin Latin hypercube designs in two dimensions. Operations Research 55, 158–169 (2007)
23. Battin, R.H.: An Introduction to the Mathematics and Methods of Astrodynamics, Revised Edition (Aiaa Education Series). AIAA (American Institute of Aeronautics & Ast; Revised edition (1999) ISBN-13: 978-1563473425
24. Vasile, M.: A Behavioral-based Meta-heuristic for Robust Global Trajectory Optimization. In: IEEE Congress on Evolutionary Computing (CEC 2007) Proceedings, pp. 2056–2063 (2007)
25. Vasile, M., Locatelli, M.: A hybrid multiagent approach for global trajectory optimization. Journal of Glaobal Optimization (accepted) (to appear, 2007)

Nature-Inspired Particle Mechanics Algorithm for Multi-Objective Optimization

Xiang Feng and Francis C.M. Lau

Department of Computer Science, The University of Hong Kong, Hong Kong
{xfeng,fcmlau}@cs.hku.hk

In many real world optimization problems, several optimization goals have to be considered in parallel. For this reason, there has been a growing interest in multi-objective optimization (MOO) in the past many years. Several new approaches have recently been proposed, which produced very good results. However, existing techniques have solved mainly problems of "low dimension", i.e., with less than 10 optimization objectives. This chapter proposes a new computational algorithm whose design is inspired by particle mechanics in physics. The algorithm is capable of solving MOO problems of high dimensions. There is a deep and useful connection between particle mechanics and high dimensional MOO. This connection exposes new information and provides an unfamiliar perspective on traditional optimization problems and approaches. The alternative of *particle mechanics algorithm* (PMA) to traditional approaches can deal with a variety of complicated, large scale, high dimensional MOO problems.

1 Introduction

While single-objective *mimetic algorithms* (MAs) are well established and relatively easy to parallelize, this is not the case for many-objective mimetic algorithms. Until recently, many-objective combinatorial optimization did not receive much attention in spite of its potential application. One of the reasons is due to the difficulty of deriving many-objective combinatorial optimization models that are satisfactory. Nevertheless, the need for parallelizing many-objective mimetic algorithms to solve many-objective combinatorial optimization problems is a real one.

"Distribution problems" is a well-known class of fundamental combinatorial optimization problems, which are much more difficult to solve than assignment problems. Many practical situations can be formulated as a distribution problem. In fact, assignment problems and transportation problems are both sub-problems of the distribution problem.

In [1, 2], some methods were proposed to generate the entire set of exact solutions for simple assignment problems. These methods however seem to be efficient only for small-scale instances. For large-scale instances or more complex problems, their NP-hardness and the multi-objectivity make these problems "intractable" by those methods. For this reason, it makes sense to consider "approximate" methods such as swarm

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 255–277.
springerlink.com

intelligent methods which have been found to be efficient in treating combinatorial optimization problems; these latter methods can function independently of the mathematical structure of the problem, and generate excellent solutions in a very short time [3].

Progress often occurs at the boundaries between disciplines. Throughout history, many artifacts have been built, whose inspiration derives from the natural world. In the field of computer science, especially in artificial intelligence, the need of parallel and distributed intelligent theories and approaches inspired by the principles of nature for difficult problems becomes greater and greater. Bio-inspired approach is probably the most representative one of nature-inspired approaches (NAs) in recent times. On the other hand, physics-inspired approaches did not receive as much attention in spite of their potential advantages. Since simulated annealing algorithm (SAA) was proposed successfully in 1983 [4], no other innovative and significant physics-inspired approach has been suggested. On the contrary, significant bio-inspired approaches such as genetic algorithm (GA) (1975) [5], ant colony optimization (ACO) (1991) [6], particle swarm optimization (PSO) (1995) [7] were emerging one after another. In 1987, the elastic net (EN) approach [8] was presented, which is a physics-inspired approach. Unfortunately, the application and influence of EN were limited by EN's unsubstantial theory foundation. One must not forget although biology is experiencing rapid development, physicists have also made great strides (e.g., particle mechanics) in recent years.

This chapter proposes a new approach—based on what we call the *particle mechanics algorithm* (PMA)—to compute in parallel approximate efficient solutions to the distribution problem with multiple objectives. By proposing this approach, we try to explore a potentially new branch of MA (or NA), which is based on the laws of physical mechanics. Just like other MAs (or NAs) which draw from observations of physical processes that occur in nature, our particle mechanics (PM) based approach is inspired by physical models of particle kinematics and dynamics.

In PMA, we use mathematical formulations to describe or predict the properties and the evolution of the different states of particles. The particles represent distinct parameters in the problem, which follow a path towards a solution. Borrowing from "differential equation theory", we have developed efficient techniques for solving multi-objective optimization problems. The goal is to have all objectives optimized individually and then collectively, and satisfying all the given restrictions.

In the physical world, mutual attraction between particles causes motion. The reaction of a particle to the field of potential would change the particle's coordinates and energies. The change in the state of the particle is a result of the influence of the potential. For PMA, the objectives of individual optimizations are reached by the autonomous self-driving forces of the particles. Global optimization is achieved by the potential of the field, and any restrictions of the problem are satisfied via interaction potential between the particles.

Each particle is described by some differential dynamic equations, and it moves (to a new state in the field) according to the results of these calculations. Specifically, each particle computes the effect of its autonomous self-driving force, the field potential and the interaction potential. If the particles cannot reach an equilibrium, they will proceed to execute a goal-satisfaction process.

Although there are obvious differences between particles in classical mechanics and those in PMA, PMA is by and large inspired by classical mechanics. PMA enables feasible many-objective optimization in very large scales. The approach has a low computational complexity, which is crucial for its functioning in solving large-scale distribution problems.

This chapter is part of the authors' research work on distributed parallel theories and approaches for intelligent processing based on the generalized particle model (GPM). They have proposed the crossbar composite spring net (CCSN) approach [9], from which the GPM approach has evolved. They studied distributed and parallel algorithms for intelligent processing based on GPM, and their application in networks. GPM's application in the bandwidth allocation problem was presented in [10]. Variations of the basic theme then resulted in several extended GPM models, including the "economic generalized particle model" (E-GPM) which draws upon the economics theory of Tatonnement processes; the model has also been applied to the bandwidth allocation problem in communication networks.

All the authors' past methods based on GPM targeted at a specific application to a real-life problem. In this chapter, the PM method is described as a "generic" method meaning potentially it can be applied to a range of different problems.

The structure of this chapter is as follows. In Section 2, we present the multi-objective particle mechanics algorithm (PMA). Section 3 introduces the parallel mimetic PM algorithm for solving multi-objective problems. In Section 4, we discuss the physical meanings of PMA. In Section 5, we give some experimental results. We conclude the chapter in Section 6.

2 The PM Approach for the Multi-Objective Distribution Problem

Definition 1. In a multi-objective framework, the distribution problem can be formulated as

$$\begin{cases} \min: z^q(X) = (C^q)^T X = \sum_{i=1}^{I}\sum_{j=1}^{J} c_{ij}^q x_{ij} \quad q = 1,2,\cdots,Q \\ s.t. \quad \sum_{i=1}^{I} x_{ij} = 1 \quad j = 1,2,\cdots,J \\ \sum_{j=1}^{J} x_{ij} = 1 \quad i = 1,2,\cdots,I \end{cases} \tag{1}$$

where X is a two-dimensional distribution vector, C^q a two-dimensional weight vector ($q = 1,2,\cdots,Q$), and Q is the number of multiple objectives.

With this problem model, we can now examine the evolutionary multi-objective model which can mathematically describe PMA for the multi-objective distribution problem. The theory of evolution is a dynamical theory. The evolutionary dynamics will drive PMA to the equilibrium state.

Definition 2. The distribution and weight dynamic equations of PMA are defined, respectively, by

$$x(t+1) = x(t) + \Delta x(t) \tag{2}$$

$$c(t+1) = c(t) + \Delta c(t) \tag{3}$$

The two dynamic equations are seen as the "PMA evolution" by fictitious agents which manipulate the distribution and weight vectors until an equilibrium is reached. In PMA, the rows and columns of distribution vector X are treated as two kinds of fictitious agents (service particles and task particles). In fact, the weight vector is invariable; the evolution of the weight vector only occurs in the computing process in order for us to obtain efficient solutions of the distribution vector.

For fictitious agents—service particles (O) and task particles (S), there are three factors related to the distribution vector (X) and the weight vector (C):

- personal utility (u) (to realize the multiple objectives);
- minimal personal utility (to realize max-min fair distribution and to increase the whole utility) (F);
- interaction among particles (to satisfy the restrictions) (I).

According to "differential equation theory", a variable's increment to make it minimum is equal to the sum of negative items from related factors differentiating the variable. So we have the following definitions.

Definition 3. The increments of distribution and weight are defined, respectively, by

$$\Delta x \approx \frac{dx}{dt} = -\sum_{q=1}^{Q} (\lambda_1^q \frac{\partial u_O^q}{\partial x} + \lambda_2^q \frac{\partial F_O^q}{\partial x}) - \lambda_3 \frac{\partial I_O}{\partial x} \tag{4}$$

$$\Delta c^q \approx \frac{dp}{dt} = -(\gamma_1^q \frac{\partial u_S^q}{\partial c} + \gamma_2^q \frac{\partial F_S^q}{\partial c}) - \gamma_3 \frac{\partial I_S}{\partial c} \quad q = 1,2,\cdots,Q \tag{5}$$

where $\lambda_1^q, \lambda_2^q, \lambda_3, \gamma_1^q, \gamma_2^q, \gamma_3$ are coefficients ($q = 1,2,\cdots,Q$).

Definition 4. Three kinds of factor functions for service particles and task particles are defined, respectively, by

$$u_{Oi}^q = 1 - \exp\left(-\sum_{j=1}^{J} c_{ij}^q \cdot x_{ij}\right) \quad q = 1,2,\cdots,Q \tag{6}$$

$$F_O^q = (k_O^q)^2 \mathrm{In} \sum_{i=1}^{I} \exp[(u_{Oi}^q)^2 \Big/ 2(k_O^q)^2] \quad q = 1,2,\cdots,Q \tag{7}$$

$$I_O = a_1 \sum_{i=1}^{I} \left(\sum_{j=1}^{J} x_{ij} - 1\right)^2 + a_2 \sum_{j=1}^{J} \left(\sum_{i=1}^{I} x_{ij} - 1\right)^2 \tag{8}$$

$$u_{Sj}^q = 1 - \exp\left(-\sum_{i=1}^{I} c_{ij}^q \cdot x_{ij}\right) \quad q = 1,2,\cdots,Q \tag{9}$$

$$F_S^q = (k_S^q)^2 \mathrm{In} \sum_{i=1}^{J} \exp[(u_{Sj}^q)^2 \Big/ 2(k_S^q)^2] \quad q = 1,2,\cdots,Q \tag{10}$$

$$I_S = I_O \tag{11}$$

where k_O^q, a_1, a_2, k_S^q are coefficients.

Now, we explain why the three kinds of functions are chosen.

1. The smaller the value of the summation in Eq. 6, the more profit the i-th service particle would obtain. The optimization problem here is posed as a minimization problem. And we use the exponential function in order that $u^q_{Oi}(t)$ would be between 0 and 1. $u^q_{Oi}(t)$ can be regarded as the q-th dimensional utility of service particle. The smaller $u^q_{Oi}(t)$ is, the more profit service particle O_i would get. Schematically, the q-th dimensional utility function u^q_{Oi} of a particle corresponds to the q-th dimensional coordinate of the q-th dimensional force-field. We define the distance from the bottom boundary to the upper boundary of all q dimensional force-fields to be 1. The physical meaning of PMA is discussed in Section IV. A graphical presentation of $u^q_{Oi}(t)$ is shown in Fig. 1. Obviously, the smaller $u^q_{Oi}(t)$ is the better. The presentation of $u^q_{Sj}(t)$ in Eq. 9 is similar.
2. For Eq. 7, $0 < k^q_O < 1$ is a parameter to be tuned in the implementation. The smaller F^q_O is, the better. With Eq. 7, we attempt to construct a potential energy function, F^q_O, such that the decrease of its value would imply the decrease of the maximal utility of all the service particles. We prove that in Theorem 1. This way we can optimize the distribution problem in the sense that we consider not only the individual personal utility, but also the aggregate utilities, by decreasing the maximum utility of all the service particles again and again. In fact, k^q_O represents the strength of the downward gravitational force in the q-th dimensional force-field. The bigger k^q_O is, the faster the particles would move down; hence, k^q_O influences the convergence speed of the distribution problem. k^q_O needs to be carefully adjusted in order to minimize the q-th objective. The explanation of F^q_S in Eq. 10 is likewise.
3. For Eq. 8, $0 < a_1, a_2 < 1$. The smaller I_O is, the better. a_1, a_2 are weights applied to the distribution availability of service and the satisfactory ratio of the demands, respectively. Eq. 8 describes the effect of interactions among service particles during the distribution process. The first term and the second term of $I_O(t)$ perform penalty functions with respect to the constraints on the utilization of service (or resources) (i.e., service particles) and the degree of satisfaction of the demands (i.e., task particles) respectively. Therefore, distribution utilization and demands' satisfaction can be explicitly included as optimization objectives through some appropriate choices

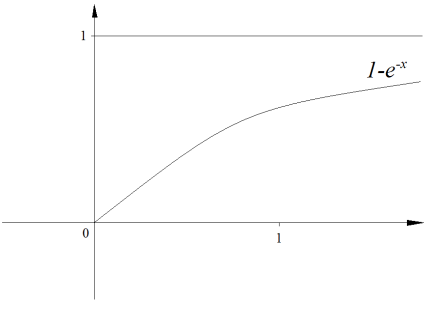


Fig. 1. Graphical presentation of $u^q_{Oi}(t)$

of the coefficients a_1 and a_2 respectively. We presume that there are specific interactive forces among particles, and these forces may cause the potential energy components represented by the first and second term of $I_O(t)$ to decrease. In Eq. 11, I_S is defined as the same as I_O.

We can therefore obtain the iteration velocity of service particles and task particles by the following equations, respectively.

$$v_{Oi}^q = du_{Oi}^q/dt = \frac{\partial u_{Oi}^q}{\partial x_{ij}} \frac{dx_{ij}}{dt} \tag{12}$$

$$v_{Sj}^q = du_{Sj}^q \Big/ dt = \frac{\partial u_{Sj}^q}{\partial c_{ij}^q} \frac{dc_{ij}^q}{dt} \tag{13}$$

v_{Oi}^q represents the iteration velocity of the q-th objective by service particle O_i. Meanwhile, v_{Oi}^q represents the velocity of the downward movement of service particle O_i in the q-th dimensional force-field. The meaning of v_{Sj}^q is similar; it represents the iteration velocity of the q-th objective by task particle S_j and the velocity of the downward movement of task particle S_j in the q-th dimensional force-field.

Proving the PM model

In order to justify our choice of Eqs. 1-13 for the PM mathematical model, we give the following theorems.

Theorem 1. If k_O^q is very small, the decrease of F_O^q will cause a decrease of the service particles' maximal utility. (Likewise, if k_S^q is very small, a decrease of F_S^q will cause a decrease of the task particles' maximal utility.)

Proof. Supposing that
$M(t) = \max_i[(u_{Oi}^q)^2(t)]$. Because

$$M(t) = \max_i (u_{Oi}^q)^2(t) \le \sum_{i=1}^{I} (u_{Oi}^q)^2(t) \le I \cdot \max_i (u_{Oi}^q)^2(t) = I \cdot M(t),$$

we have

$$\left[e^{\frac{M(t)}{2(k_O^q)^2}} \right]^{2(k_O^q)^2} \le \left[\sum_{i=1}^{I} e^{\frac{(u_{Oi}^q)^2(t)}{2(k_O^q)^2}} \right]^{2(k_O^q)^2} \le \left[I \cdot e^{\frac{M(t)}{2(k_O^q)^2}} \right]^{2(k_O^q)^2}.$$

Simultaneously taking the logarithm of each side of the equation above leads to

$$M(t) \ge 2(k_O^q)^2 \mathrm{In} \sum_{i=1}^{I} e^{\frac{(u_{Oi}^q)^2(t)}{2(k_O^q)^2}} \ge M(t) + 2(k_O^q)^2 \mathrm{In} I,$$

$$2(k_O^q)^2 \mathrm{In} \sum_{i=1}^{I} e^{\frac{(u_{Oi}^q)^2(t)}{2(k_O^q)^2}} \le M(t) \le 2(k_O^q)^2 \mathrm{In} \sum_{i=1}^{I} e^{\frac{(u_{Oi}^q)^2(t)}{2(k_O^q)^2}} - 2(k_O^q)^2 \mathrm{In} I,$$

$$2F_O(t) \le \max_i u_{Oi}^q(t) \le 2F_O(t) - 2(k_O^q)^2 \mathrm{In} I.$$

Since I is the number of service particles (the number of rows of the distribution vector X), $2(k_O^q)^2 \mathrm{In} I$ is constant.

It turns out that $F_O^q(t)$ at time t represents the maximum among $u_{Oi}^q(t)$ obtained by the service particle O_i, namely, the minimum of the personal profit obtained by a service particle at time t. Hence decreasing $F_O^q(t)$ implies the decrease of the maximal utility of the service particles. □

Based on Theorem 1, we can construct the potential energy functions, F_O^q and F_S^q (see Eqs. 7 and 10), such that the decrease of their values would imply the decrease of the maximal utilities of the two kinds of particles. By decreasing the maximum utilities of the two kinds of particles again and again, the aggregate utilities of whole problem will decrease, which is the basis that PM algorithm can obtain the Max-min fair solution.

Firstly, we give the definition of Max-min Fairness. Then, Theorem 2-5 will explain why the Max-min fair solution can be obtained by the PM mathematical model as defined in Eqs. 1-13.

Definition 5. (Max-min Fairness) [11] A feasible distribution X is max-min fair if and only if a decrease of any distribution x within the domain of feasible distributions must be at the cost of an increase of some already larger distribution x. Formally, for any other feasible distribution Y, if $y_{ij} < x_{ij}$ then there must exist some i' such that $x_{i'j} \geq x_{ij}$ and $y_{i'j} > x_{i'j}$.

Theorem 2. The behavior of the service particle O_i that is related to the term of Eq. 4, $-\lambda_2^q \frac{\partial F_O^q}{\partial x}$, will always bring about the decrease of the maximal utility of all service particles, and the decrement of the maximal utility is directly proportional to the coefficient vector λ_2^q. (Likewise, The behavior of the task particle S_j that is related to the term of the Eq. 5, $-\gamma_2^q \frac{\partial F_S^q}{\partial c}$, will always bring about the decrease of the maximal utility of all task particles, and the decrement of the maximal utility is directly proportional to the coefficient vector γ_2^q.)

Theorem 3. The behavior of the service particle O_i that is related to the term of the Eq. 4, $-\lambda_1^q \frac{\partial u_O^q}{\partial x}$, will always result in the decrease of the personal utility of service particle O_i, and the decrement of its personal utility is related to coefficient vectors λ_1^q. (Likewise, The behavior of the task particle S_j that is related to the term of the Eq. 5, $-\gamma_1^q \frac{\partial u_S^q}{\partial c}$, will always result in the decrease of the personal utility of task particle S_j, and the decrement of its personal utility is related to coefficient vectors γ_1^q.

Theorem 4. The behavior of the service particle O_i that is related to the term of the Eq. 4, $-\lambda_3 \frac{\partial I_O}{\partial x}$, will decrease the potential interaction energy function I_O, with the intensity of the decrease being proportional to coefficient vector λ_3. (Likewise, The behavior of the task particle S_j that is related to the term of the Eq. 5, $-\gamma_3 \frac{\partial I_S}{\partial c}$, will decrease the potential interaction energy function I_S, with the intensity of the decrease being proportional to coefficient vector γ_3.

Theorem 5. (Max-min fair allocation) Max-min fair allocation can be obtained by the mathematical model for the distribution problem with multi-objectives as defined in Eqs. 1-13.

The proofs of Theorems 2-5 are omitted.

3 The Parallel PM Algorithm

The results given in the previous sections suggest that we may use a parallel implementation of the evolutionary particle mechanics approach to solve the multi-objective distribution problem. We consider such an algorithm in Table 1.

The algorithm PMA has in general a complexity of $O(I+J)$, where $I+J$ is the number of particles (the sum of the number of rows and columns of X). The time complexity of the algorithm is $O(I_1)$, where I_1 is the number of iterations for Costep 2 (the while loop).

4 Physical Meaning of PMA

PMA puts emphasis on

- providing a view of individual and global optimization (with one to two objectives);
- parallelization with reasonably low time complexity;
- all objectives being optimized individually as well as collectively;
- the ability to deal with social interactions;
- the physical meaning of the model.

The mathematical model of PMA has its physical meaning.

In PMA, the rows and columns of the distribution vector X are treated as two kinds of generalized particles (service particles and task particles) that are located in two groups of force-fields, respectively, hence transforming the distribution problem into the kinematics and dynamics of the particles in the two groups of force-fields.

The two groups of force-fields are a group of service (or resource) force-fields and a group of task force-fields. Every force-field in a group of service force-fields or in a group of task force-fields is a Q-dimensional space where coordinates in the space are in $[0,1]$.

Table 1. The PM algorithm

1. Initialization:	
$t \leftarrow 0$	
$x_{ij}(t)$, $c_{ij}^q(t)$	——Initialize in parallel
2. **While** ($v_{Oi}^q \neq 0$ or $v_{Sj}^q \neq 0$) **do**	
$t \leftarrow t+1$	
$u_{Qi}^q(t)$	——Compute in parallel according to Eq. 6
v_{Qi}^q	——Compute in parallel according to Eq. 12
$u_{Sj}^q(t)$	——Compute in parallel according to Eq. 9
v_{Sj}^q	——Compute in parallel according to Eq. 13
$dx_{ij}(t)/dt$	——Compute in parallel according to Eq. 4
$x_{ij}(t) \leftarrow x_{ij}(t-1)+dx_{ij}(t)/dt$	——Compute in parallel according to Eq. 2
$dc_{ij}^q(t)/dt$	——Compute in parallel according to Eq. 5
$c_{ij}^q(t) \leftarrow c_{ij}^q(t-1)+dc_{ij}^q(t)/dt$	——Compute in parallel according to Eq. 3

If the number of minimum objectives is 1, the particles will move downwards on a 1-dimensional space (a line) ($x_1 \in [0,1]$) during the optimization process. If the number of minimum objectives is 2, the particles will move towards the origin in a 2-dimensional space (a plane) ($x_1 \in [0,1], x_2 \in [0,1]$) during the optimization process. Analogously, if the number of minimum objectives is Q, the particles will move towards the origin on a Q-dimensional space ($x_1 \in [0,1], \cdots, x_q \in [0,1], \cdots, x_Q \in [0,1]$) during the optimization process, where x_q is a coordinate of the q-dimensional space.

Particles in PMA move not only under outside forces, but also under their internal forces; hence they are different from particles in classical physics. The kinds of force-fields (resource force-field F_R and demands force-field F_D) are geometrically independent, without any forces directly exerted from each other; they are mutually influenced and conditioned by each other through a reciprocal procedure whereby the distribution policy of the distributions x and the weight policy of the weights c change alternatively. In this way, the two groups of force-fields form a pair of reciprocal dual force-field groups.

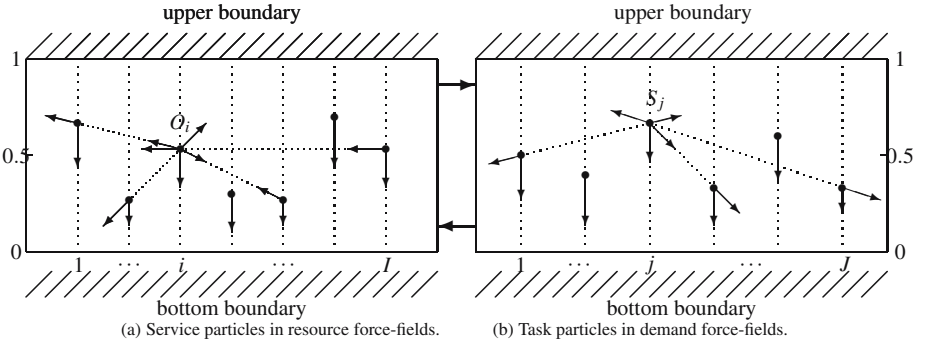


Fig. 2. The physical model of PMA for the distribution problem with one objective

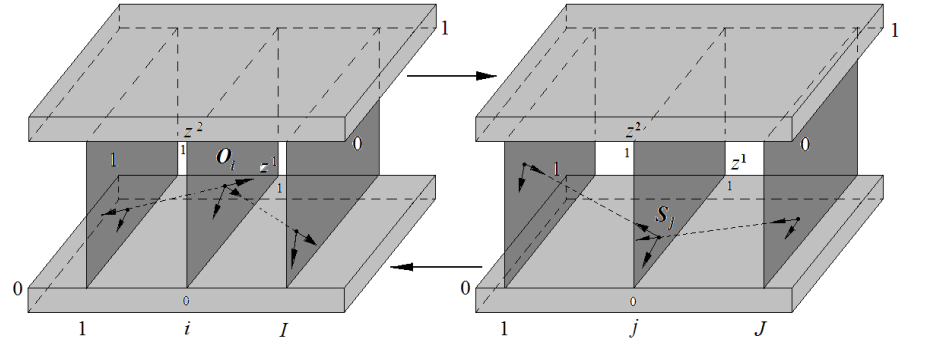


Fig. 3. The physical model of PMA for the distribution problem with two objectives

In a resource force-field F_R, the coordinates of the Q-dimensional space of service particles represent the utilities of the rows of the distribution vector X that are described by the service particles. A particle will be influenced simultaneously by several kinds of forces in the Q-dimensional space, which include the gravitational force of the Q-dimensional space force-field where the particle is located, the pulling or pushing forces stemming from the interactions with other particles in the same force-field, and the particle's own autonomous driving force.

When the number of minimum objectives is 1, all the above-mentioned forces that are exerted on a particle are dealt with as forces along a vertical direction (along a line). Thus a particle will be driven by the resultant force of all the forces that act on it upwards or downwards, and moves along a vertical direction. The larger the downward resultant force on a particle, the faster the downward movement of the particle. When the downward resultant force on a particle is equal to zero, the particle will stop moving, being at an equilibrium status. As shown in Fig. 2, the service particles that have service or resource move in the resource force-fields F_R, and the task particles that require distribution move in the demand force-fields F_D.

The downward gravitational force of a force-field on a particle causes a downward component of the motion of the particle, which represents the tendency that the particle pursues the common benefit of the whole group. The downward or upward component of the motion of a particle, which is related to the interactions with other particles, depends upon the strengths and categories of the interactions. The particle's own autonomous driving force is proportional to the degree the particle tries to move downwards in the force-field where it is located, i.e., the particle (service particle or task particle) tries to acquire its own minimum utility.

When the number of minimum objectives is two, each service particle and task particle move towards the origin in a unit plane, as shown in Fig. 3.

When the number of minimum objectives is Q, each service particle and task particle move towards the origin in a Q-dimensional space.

One major difference between the particle of the proposed generalized particle model and the particle of a classical physical model is that the generalized particle has its own driving force which depends upon the autonomy of the particle. All the generalized particles, both in different Q-dimensional spaces of the same force-field and in different force-fields simultaneously, evolve under their exerted forces; as long as they gradually reach their equilibrium positions from their initial positions which are set at random, we can obtain a feasible solution to the multi-objective distribution problem.

5 Simulations

Here, we give the experimental results which serve six purposes. First, we use a simple example (a small-scale problem) to explain how our PM algorithm is used. Secondly, the effectiveness of PM algorithm is tested on many-objective large-scale problems. Thirdly, we show the actual times and iterations used to solve multi-objective optimization problems on a cluster, which can verify the efficiency and parallelism of our PM algorithm. Fourthly, we test the effectiveness of our PM algorithm to solve optimization problems with different number of objectives (from single objective to many-objective

problems). Fifthly, the performance of the PM algorithm is compared against NSGA-II [12] for two-objective distribution problems. Finally, we make a general comparison between the PM algorithm and other benchmark MAs.

All the experiments presented in this section are completed on a cluster. Each of the machines of the cluster has a Pentium 4 2.0 GHz CPU with 512 Kbytes of L2 cache and 512 Mbytes of DDR SDRAM, and they are interconnected via Fast Ethernet.

5.1 How to Use the PM Algorithm

Here we give a simple distribution problem, and then use our method to find the solution.

$$C^1 = \begin{pmatrix} 8 & 7 & 10 & 1 & 4 \\ 7 & 11 & 16 & 0 & 5 \\ 2 & 7 & 6 & 19 & 15 \\ 3 & 6 & 4 & 7 & 11 \\ 14 & 5 & 7 & 3 & 2 \end{pmatrix} \quad C^2 = \begin{pmatrix} 10 & 7 & 10 & 5 & 11 \\ 3 & 19 & 5 & 3 & 13 \\ 2 & 18 & 9 & 0 & 1 \\ 13 & 3 & 7 & 5 & 12 \\ 7 & 4 & 6 & 15 & 3 \end{pmatrix} \qquad q = 1,2.$$

Find an X satisfying

$$\begin{cases} \min : z^q(X) = (C^q)^T X = \sum_{i=1}^{5} \sum_{j=1}^{5} c_{ij}^q x_{ij} \qquad q = 1,2 \\ s.t. \qquad \sum_{i=1}^{5} x_{ij} = 1 \qquad j = 1,2,\cdots,5 \\ \qquad\quad \sum_{j=1}^{5} x_{ij} = 1 \qquad i = 1,2,\cdots,5 \end{cases}$$

We use the PM algorithm to solve this distribution problem.

Step 1. Initialization: $(t = 0)$

$$X = \begin{pmatrix} 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.2 & 0.2 \end{pmatrix}$$

x_{ij} is also initialized as a random number between 0 and 1. Based on some experiments we have done, we found that the results are not affected by the initialization of X.

Standardization of C^1 and C^2:

$$C^1 = \begin{pmatrix} 0.80 & 0.70 & 1.00 & 0.10 & 0.40 \\ 0.44 & 0.69 & 1.00 & 0 & 0.31 \\ 0.11 & 0.37 & 0.32 & 1.00 & 0.79 \\ 0.27 & 0.55 & 0.36 & 0.64 & 1.00 \\ 1.00 & 0.36 & 0.50 & 0.21 & 0.14 \end{pmatrix} \quad C^2 = \begin{pmatrix} 0.91 & 0.64 & 0.91 & 0.45 & 1.00 \\ 0.16 & 1.00 & 0.26 & 0.16 & 0.68 \\ 0.11 & 1.00 & 0.50 & 0 & 0.06 \\ 1.00 & 0.23 & 0.54 & 0.38 & 0.92 \\ 0.47 & 0.27 & 0.40 & 1.00 & 0.20 \end{pmatrix}$$

1. According to Eq. 1, $z^q(X) = (C^q)^T X = \sum_{i=1}^{5} \sum_{j=1}^{5} c_{ij}^q x_{ij}$,

we get

$$z^1 = 2.6120, \quad z^2 = 2.6500$$

2. According to Eq. 6, $u_{Oi}^q = 1 - \exp\left(-\sum_{j=1}^{5} c_{ij}^q \cdot x_{ij}\right)$, we compute in parallel and get

$$u_{O1}^1 = 0.4512 \quad u_{O2}^1 = 0.3861 \quad u_{O3}^1 = 0.4043 \quad u_{O4}^1 = 0.4311 \quad u_{O5}^1 = 0.3573$$
$$u_{O1}^2 = 0.5425 \quad u_{O2}^2 = 0.3636 \quad u_{O3}^2 = 0.2839 \quad u_{O4}^2 = 0.4588 \quad u_{O5}^2 = 0.3737$$

3. According to Eq. 9, $u_{Sj}^q = 1 - \exp\left(-\sum_{i=1}^{5} c_{ij}^q \cdot x_{ij}\right)$, we compute in parallel and get

$$u_{S1}^1 = 0.4079 \quad u_{S2}^1 = 0.4137 \quad u_{S3}^1 = 0.4706 \quad u_{S4}^1 = 0.3229 \quad u_{S5}^1 = 0.4102$$
$$u_{S1}^2 = 0.4114 \quad u_{S2}^2 = 0.4663 \quad u_{S3}^2 = 0.4067 \quad u_{S4}^2 = 0.3283 \quad u_{S5}^2 = 0.4356$$

Step 2. Compute in parallel:

The first evolutionary iteration ($t = 1$):

1. According to Eq. (4), we have

$$\Delta x_{ij} \approx \frac{dx_{ij}}{dt} = -\sum_{q=1}^{2}\left(\lambda_1^q \frac{\partial u_{Oi}^q}{\partial x_{ij}} + \lambda_2^q \frac{\partial F_O^q}{\partial x_{ij}}\right) - \lambda_3 \frac{\partial I_O}{\partial x_{ij}}$$
$$= -\lambda_1^1 \frac{\partial u_{Oi}^1}{\partial x_{ij}} - \lambda_2^1 \frac{\partial F_O^1}{\partial x_{ij}} - \lambda_1^2 \frac{\partial u_{Oi}^2}{\partial x_{ij}} - \lambda_2^2 \frac{\partial F_O^2}{\partial x_{ij}} - \lambda_3 \frac{\partial I_O}{\partial x_{ij}}$$

where $\frac{\partial u_{Oi}^q}{\partial x_{ij}} = c_{ij}^q \cdot \exp\left(-\sum_{j=1}^{J} c_{ij}^q \cdot x_{ij}\right)$

$$\frac{\partial F_O^q}{\partial x_{ij}} = \frac{\partial F_O^q}{\partial u_{Oi}^q} \cdot \frac{\partial u_{Oi}^q}{\partial x_{ij}} = (k_O^q)^2 \cdot \frac{\exp\left\{(u_{Oi}^q)^2/[2(k_O^q)^2]\right\} \cdot [(u_{Oi}^q)/(k_O^q)^2]}{\sum_{i=1}^{5} \exp\left\{(u_{Oi}^q)^2/[2(k_O^q)^2]\right\}} \cdot \frac{\partial u_{Oi}^q}{\partial x_{ij}}$$

$$\frac{\partial I_O}{\partial x_{ij}} = 2Ja_1 \sum_{i=1}^{I}\left(\sum_{j=1}^{J} x_{ij} - 1\right) + 2Ia_2 \sum_{j=1}^{J}\left(\sum_{i=1}^{I} x_{ij} - 1\right)$$

2. According to Eq. (5), we have

$$\Delta c_{ij}^q \approx \frac{dc_{ij}^q}{dt} = -\left(\gamma_1^q \frac{\partial u_{Sj}^q}{\partial c_{ij}^q} + \gamma_2^q \frac{\partial F_S^q}{\partial c_{ij}^q}\right) - \gamma_3 \frac{\partial I_S}{\partial c_{ij}^q}$$

where $\frac{\partial u_{Sj}^q}{\partial c_{ij}^q} = x_{ij} \cdot \exp\left(-\sum_{i=1}^{I} c_{ij}^q \cdot x_{ij}\right)$

$$\frac{\partial F_S^q}{\partial c_{ij}^q} = \frac{\partial F_S^q}{\partial u_{Sj}^q} \cdot \frac{\partial u_{Sj}^q}{\partial c_{ij}^q} = (k_S^q)^2 \cdot \frac{\exp\left\{(u_{Sj}^q)^2/[2(k_S^q)^2]\right\} \cdot [(u_{Sj}^q)^2/(k_S^q)^2]}{\sum_{j=1}^{5} \exp\left\{(u_{Sj}^q)^2/[2(k_S^q)^2]\right\}} \cdot \frac{\partial u_{Sj}^q}{\partial c_{ij}^q}$$

$$\frac{\partial I_S}{\partial c_{ij}^q} = 0$$

3. In addition,

$$x_{ij}(t=1) = x_{ij}(t=0) + \Delta x_{ij}(t=1)$$
$$c_{ij}^1(t=1) = c_{ij}^1(t=0) + \Delta c_{ij}^1(t=1)$$
$$c_{ij}^2(t=1) = c_{ij}^2(t=0) + \Delta c_{ij}^2(t=1)$$
$$\lambda_1^1 = 0.05 \quad \lambda_2^1 = 0.05 \quad \lambda_1^2 = 0.05 \quad \lambda_2^2 = 0.05 \quad \lambda_3 = 0.01$$
$$\gamma_1^1 = 0.05 \quad \gamma_2^1 = 0.05 \quad \gamma_1^2 = 0.05 \quad \gamma_2^2 = 0.05 \quad \gamma_3 = 0.01$$
$$a_1 = 0.5 \quad a_2 = 0.5 \quad k_O^1 = k_O^2 = k_S^1 = k_S^2 = 0.8$$

As for these coefficients, we can draw the following conclusions from the experiments we have done.

- When k_O^q (or k_S^q) is larger, the corresponding convergence speed is faster.
- If the values of λ and γ change in direct proportion, the experimental results will hardly be influenced.
- If we increase λ_1^q, λ_2^q and do not touch the other coefficients, the q-th objective will take precedence over all the other objectives.

We compute in parallel and get

$$X(t=1) = \begin{pmatrix} 0.1890 & 0.2007 & 0.1819 & 0.2277 & 0.2006 \\ 0.2133 & 0.1708 & 0.1881 & 0.2301 & 0.1978 \\ 0.2246 & 0.1766 & 0.2001 & 0.1968 & 0.2019 \\ 0.1973 & 0.2138 & 0.2100 & 0.2054 & 0.1736 \\ 0.1778 & 0.2109 & 0.2003 & 0.1886 & 0.2223 \end{pmatrix}$$

$$C^1(t=1) = \begin{pmatrix} 0.7751 & 0.6784 & 0.9719 & 0.0965 & 0.3876 \\ 0.4263 & 0.6687 & 0.9719 & 0 & 0.3004 \\ 0.1066 & 0.3586 & 0.3110 & 0.9648 & 0.7655 \\ 0.2616 & 0.5330 & 0.3499 & 0.6175 & 0.9690 \\ 0.9688 & 0.3489 & 0.4860 & 0.2026 & 0.1357 \end{pmatrix}$$

$$C^2(t=1) = \begin{pmatrix} 0.8818 & 0.6219 & 0.8816 & 0.4343 & 0.9702 \\ 0.1550 & 0.9717 & 0.2519 & 0.1544 & 0.6597 \\ 0.1066 & 0.9717 & 0.4844 & 0 & 0.0582 \\ 0.9690 & 0.2235 & 0.5231 & 0.3667 & 0.8926 \\ 0.4554 & 0.2624 & 0.3875 & 0.9650 & 0.1940 \end{pmatrix}$$

$$z^1(t=1) = 2.5243, \qquad z^2(t=1) = 2.5590$$

Obviously, $z^1(t=1) < z^1(t=0)$ and $z^2(t=1) < z^2(t=0)$, and the distribution problem is optimized.

The evolutionary experimental results and the optimization trend from $t=0$ to $t=18$ are shown in Fig. 4 and Fig. 5.

As shown in Fig. 4 and Fig. 5, the two-objective distribution problem is optimized by the evolution of the PM algorithm. The convergence speed is faster at the beginning of evolution. The optimization trend of z^1 and z^2 reflects exactly the optimization of the problem, that is, the distribution problem is optimized step by step.

5.2 Effectiveness of the PM Algorithm

We pick 320×200-scale combinatorial optimization problems with five objectives to test our PM algorithm. Of course, for larger scales and problems of more objectives, our PM algorithm can work out the optimum solutions quickly too.

The problem-related matrixes C^1, C^2, C^3, C^4, C^5 are randomly generated. We let
(1) $\lambda_1^q, \lambda_2^q (q = \overline{1,5})$ be 0.05;
(2) $\lambda_3 = 0.01$;
(3) $\gamma_1^q, \gamma_2^q (q = \overline{1,5})$ be 0.05;
(4) $\gamma_3 = 0.01$;

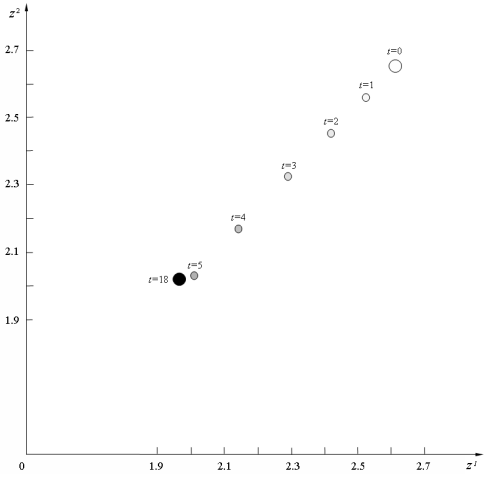


Fig. 4. Optimization from $t = 0$ to $t = 18$

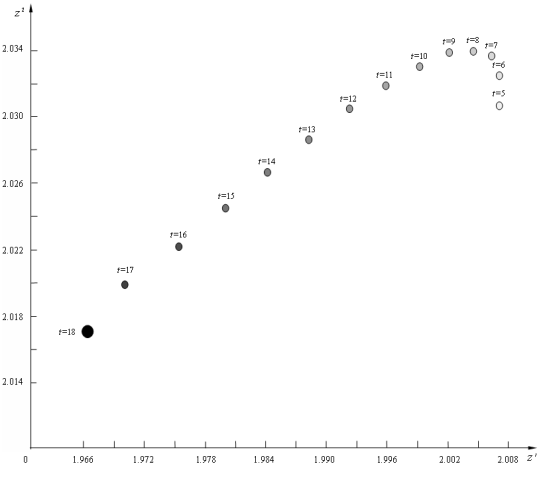


Fig. 5. Optimization from $t = 5$ to $t = 18$

(5) a_1=0.5 and a_1=0.5;
(6) $k_O^q, k_S^q (q = \overline{1,5})$ be 0.8.

Because the problem-related matrices are too large to list here, we give the results of the problem directly. The evolutionary experimental results for z^1, z^2, z^3, z^4, z^5 are depicted in Fig. 6.

We use 16 computing modes of the cluster (mentioned at the beginning of this section) to compute the solution. More data about the solution are shown in Table 2.

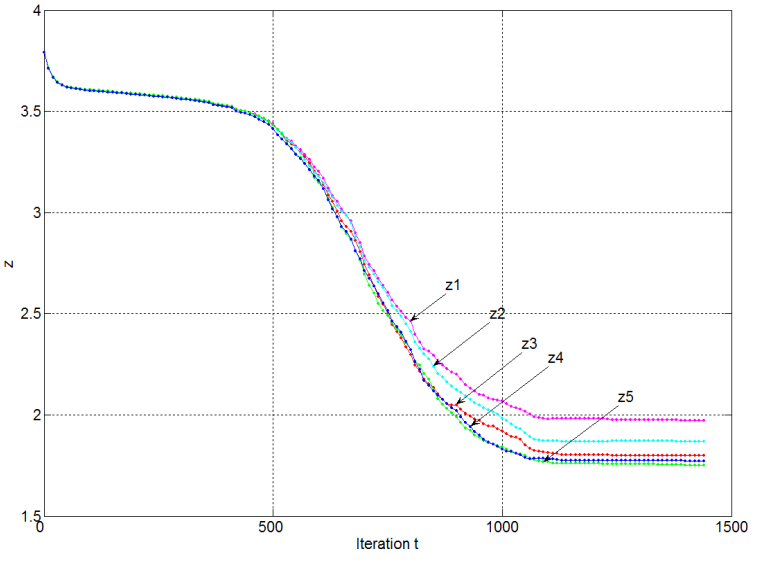


Fig. 6. The optimization trend of five objectives in a large-scale problem

Table 2. Other solution-related data

Objectives	5
Processors	16
Scale	320×200
Max iterations	50000
Convergence iterations	1447
Convergence time (second)	49.934
$z(t = 1447)$	9.16125010
$z(t = 1448)$	9.16125010

As shown in Fig. 6, the five optimization curves of z^1, z^2, z^3, z^4, z^5 reflect exactly the optimization of the problem and five objectives. The convergence speeds of five objectives are fastest at the beginning (about from $t = 0$ to $t = 20$) and faster within the medium range. At $t = 1447$, $z(= z^1 + z^2 + z^3 + z^4 + z^5)$ reaches its minimum (9.16125010), and stays unchanged in the remainder of the iterations. For this 320×200-scale problem, PM algorithm converges to a stable equilibrium state when $t = 1447$. Undoubtedly this solution is the optimum solution as it minimizes $z(z^1, z^2, z^3, z^4, z^5)$, which verifies the approach's convergence and its ability to arrive at the optimum for large-scale many-objective problems.

In Table 2, "Processors" represents the number of computing nodes in the cluster we used to compute the solution. We use the number of rows and columns of distribution matrix (X) to represent the "scale" of the problem. "Convergence time and

iterations" represent the actual time and iterations used to compute the solution (a stable equilibrium) on the cluster. Considering the situation that, after many iterations, stable equilibrium is still not reached by the PM algorithm, we should end the computation at this time. When $t =$"Max iterations", the PM algorithm ends. z is aggregated by z^1, z^2, z^3, z^4, z^5, representing the optimization of the whole problem.

5.3 Efficiency and Parallelism of the PM Algorithm

Mimetic algorithms (MAs) provide a valuable alternative to traditional methods because of their inherent parallelism and their ability to deal with difficult problems. Our PM approach as a new branch of MA, has these two main advantages too. The distribution and weight variables, x_{ij} and c_{ij}^q, can be computed and updated in parallel without any information exchange, which is the foundation of PMA's parallelism.

In order to test the high parallelism and good scalability of the PM algorithm, we compute multi-objective problems of different scales both in the parallelized way and the non-parallelized way. The hardware and software environment in which these simulations are run have been mentioned at the beginning of this section.

Convergence time and iterations for different multi-objective problems using PMA are shown in Table 3 and Table 4. In the two Tables, the data can be categorized into two

Table 3. Convergence time and iterations of PMA for multi-objective medium-scale problems

Objective	Scale	1 parallel node		4 parallel nodes		8 parallel nodes		16 parallel nodes	
		time(s)	iterations	time(s)	iterations	time(s)	iterations	time(s)	iterations
5	32×4	0.085	101	0.296	101	0.516	101	0.705	99
5	32×8	0.167	102	0.295	100	0.509	95	0.747	105
5	32×12	0.272	101	0.289	99	0.505	101	0.729	100
5	32×16	0.365	102	0.337	98	0.516	102	0.738	100
5	32×20	0.436	105	0.394	108	0.63	104	0.77	105
5	32×24	0.611	113	0.453	110	0.607	111	0.786	108
5	32×28	0.672	115	0.473	116	0.645	114	0.864	123
5	32×32	0.842	121	0.51	124	0.744	120	0.881	125
5	32×36	1.059	140	0.568	132	0.737	139	0.878	126
5	32×40	1.087	133	0.664	139	0.819	145	1.16	153
5	32×44	1.333	147	0.807	159	0.853	154	1.01	144
5	32×48	1.866	183	0.805	161	0.916	161	1.047	149
5	32×52	1.74	161	0.87	171	0.962	174	1.268	189
5	32×56	2.367	201	0.995	189	1.135	195	1.285	190
5	32×60	2.753	217	1.111	189	1.292	222	1.242	183
5	32×64	2.851	222	1.516	256	1.239	213	1.401	218
5	32×68	2.915	206	1.386	229	1.325	226	1.601	239
5	32×72	3.329	220	1.342	217	1.465	250	2.085	267
5	32×76	3.737	238	1.701	268	1.667	283	1.865	286
5	32×80	5.304	318	2.005	302	1.613	263	1.897	287

Table 4. Convergence time and iterations of PMA for multi-objective large-scale problems

Objective	Scale	1 parallel node		4 parallel nodes		8 parallel nodes		16 parallel nodes	
		time(s)	iterations	time(s)	iterations	time(s)	iterations	time(s)	iterations
5	320×40	12.81	153	3.702	159	2.39	160	1.90	166
5	320×80	55.26	324	14.161	316	7.92	316	5.74	348
5	320×120	136.11	537	39.516	580	19.65	554	12.62	602
5	320×160	289.79	851	79.355	907	43.05	933	25.87	973

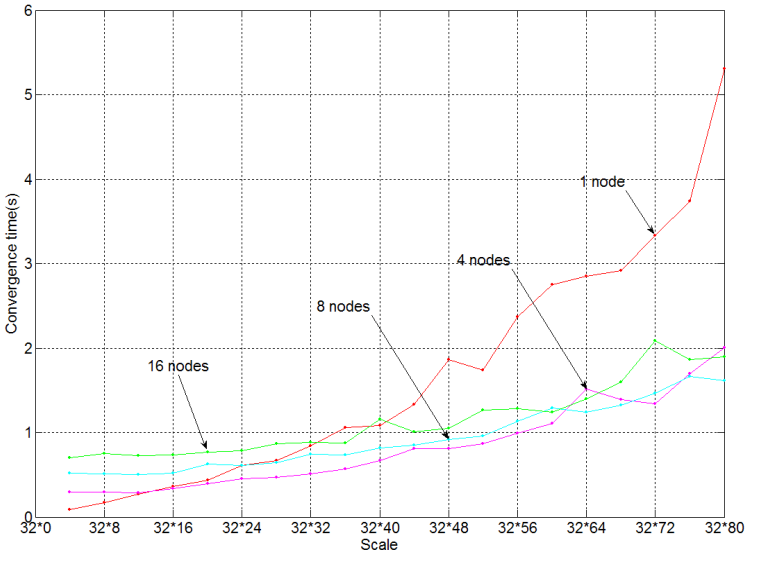


Fig. 7. Convergence time of PMA for multi-objective medium-scale problems

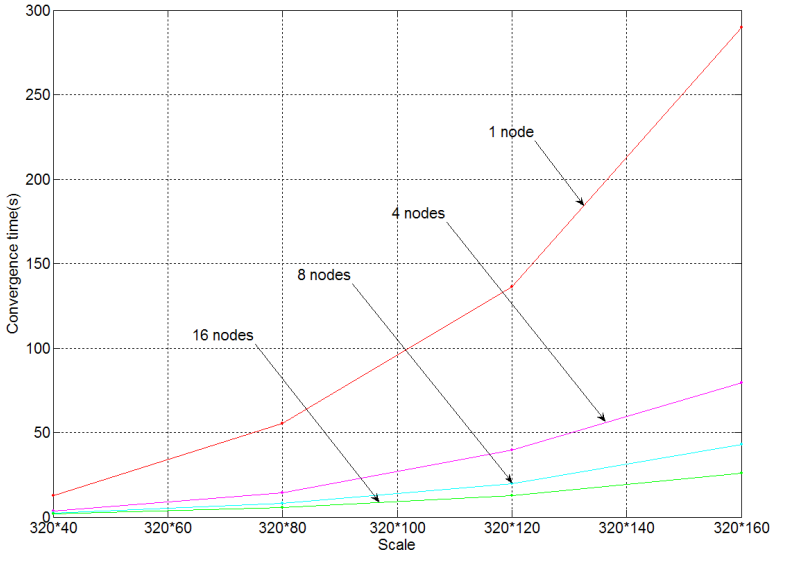


Fig. 8. Convergence time of PMA for multi-objective large-scale problems

parts. One part is related to the sequential version, which comes from our experimental results using one computing node of the cluster. The other part is related to the parallel version, which comes from the results using 4, 8 and 16 computing codes of the cluster. “Iterations” and “time” are the iterations and time PMA takes to converge.

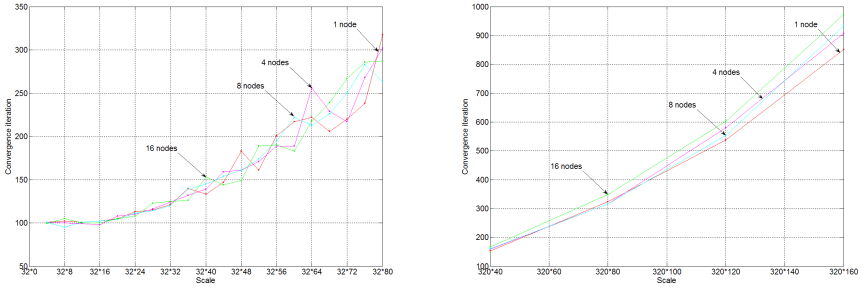


Fig. 9. Convergence speeds of PMA for multi-objective problems

As shown in Fig. 7 and Fig. 8, the convergence time of the sequential version increases exponentially with the scale, which is similar to all the exact methods. When parallelized, the convergence time drops significantly across the larger scale problems. The times for the smaller scale problems are dominated by the message exchange time, and the sequential version appears to be more efficient in that range. For the large-scale problems (see Fig. 8), the more computing nodes of the cluster we used, the less actual times used to compute the solutions.

As shown in Fig. 9, the convergence speeds (in terms of number of iterations to convergence) of the sequential version and parallel version increase steadily with the scale. The convergence speed of PMA is almost not related to the computing version (sequential version or parallel version).

5.4 Problems with Different Numbers of Objectives by PMA

Here we test the effectiveness of our PM algorithm to solve problems with different numbers of objectives (from single objective to many-objective problems). Convergence time and speeds using PMA for different numbers of objectives are shown in Table 5 and Table 6. Table 5 and Fig. 10 show the relation between the convergence time, the number of objectives and computing version (sequential or parallel) when the scale of the problems is fixed (at 80×80). Table 6 and Fig. 11 show the relation between the convergence time, the number of objectives and the scale when the computing version is fixed (parallel version using 16 computing nodes of the cluster).

As shown in Fig. 10, the convergence time increases steadily when the number of objectives of the problem increases. The more computing nodes of the cluster are used to compute the solutions, the slower the convergence time's increase with the number of objectives. The experimental results verify the good parallelism of our PM approach for multi-objective optimization problems.

As shown in Fig. 11, the more the objectives, the faster the convergence time would increase with the scale. The experimental results verify that our PM approach can deal with many-objective, large-scale optimization problems.

Table 5. Convergence time and speeds of PMA for different numbers of objectives, medium-scale problems

Objective	Scale	1 parallel node		4 parallel nodes		8 parallel nodes		16 parallel nodes	
		time(s)	iterations	time(s)	iterations	time(s)	iterations	time(s)	iterations
1	80×80	4.69	539	1.57	526	1.35	536	1.47	531
5	80×80	13.00	308	3.92	305	2.77	297	2.40	280
10	80×80	25.25	298	7.80	310	4.80	283	4.49	288
15	80×80	36.69	292	11.28	297	7.53	296	6.92	313
20	80×80	51.90	308	16.02	319	11.22	331	10.37	335
25	80×80	70.70	336	21.07	326	14.10	324	12.47	344
30	80×80	81.58	323	24.19	319	17.42	342	12.79	308
35	80×80	101.45	338	31.46	355	19.85	338	17.39	361
40	80×80	123.20	367	36.58	362	24.09	360	19.55	355
45	80×80	141.15	373	40.62	354	26.76	347	21.91	343
50	80×80	157.11	373	49.10	382	29.26	342	25.41	353
55	80×80	181.71	386	52.29	372	34.65	376	28.76	379
60	80×80	186.66	369	58.20	383	37.07	364	31.30	380
65	80×80	221.14	399	67.81	381	41.55	376	36.08	397
70	80×80	237.41	404	69.53	416	46.24	391	39.32	402
75	80×80	251.73	396	75.90	402	50.80	380	41.70	409
80	80×80	278.09	409	83.24	405	52.46	416	43.63	389
85	80×80	313.88	440	85.04	397	56.10	387	47.35	406
90	80×80	345.14	456	86.90	375	63.68	422	52.20	426

Table 6. Convergence time and speeds of PMA for different numbers of objectives, large-scale problems using 16 computing nodes of the cluster

Scale	1 objective		5 objectives		50 objectives	
	time(s)	iterations	time(s)	iterations	time(s)	iterations
320×40	0.983	242	1.90	166	16.767	176
320×80	2.583	600	5.74	348	58.827	393
320×120	5.453	1046	12.62	602	169.793	816
320×160	10.8	1759	25.87	973	343.651	1312
320×200	15.868	2227	42.034	1293	569.017	1832
320×240	29.419	3327	63.617	1722	1044.953	2842
320×280	38.901	4159	99.039	2276	1571.968	3711
320×320	63.718	5711	130.258	2679	2113.836	4351
320×360	84.762	6806	183.987	3410	3358.083	5839
320×400	118.984	8404	247.028	4122	4276.371	6758
320×440	155.671	9816	336.842	4988	5839.789	8245
320×480	262.693	12544	446.898	5834	8130.714	9963
320×520	307.017	13909	578.865	6845	9729.199	11014
320×560	374.298	15629	699.978	7718	13528.261	13064
320×600	425.01	17180	861.57	8609	15294.553	14319
320×640	594.781	19854	997.091	9528	18108.043	15921
320×680	698.079	21922	1641.718	11881	25035.577	18459
320×720	774.948	23927	1418.417	11523	27951.384	20024
320×760	987.354	26780	2122.651	13738	37178.713	22961
320×800	1332.217	30418	2196.459	14275	38786.457	24314

5.5 Performance Comparison between PMA and NSGA-II

In this subsection, we compare the proposed algorithm, PMA, with NSGA-II [12] for a number of two-objective distribution problems which are similar to the example problem mentioned in subsection 5.1. For the NSGA-II, we use a standard real-parameter SBX and polynomial mutation operator with $\eta_c = 10$ and $\eta_m = 10$, respectively. We run

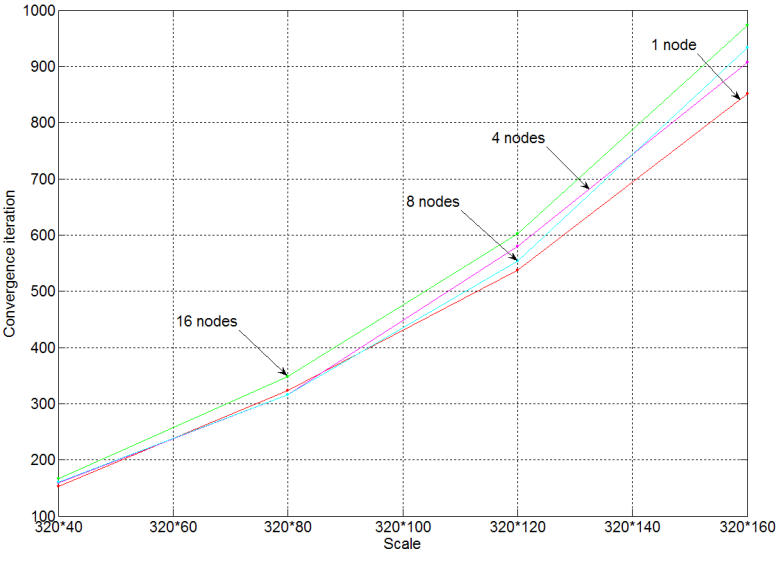


Fig. 10. Convergence time of PMA for different numbers of objectives, medium-scale problems

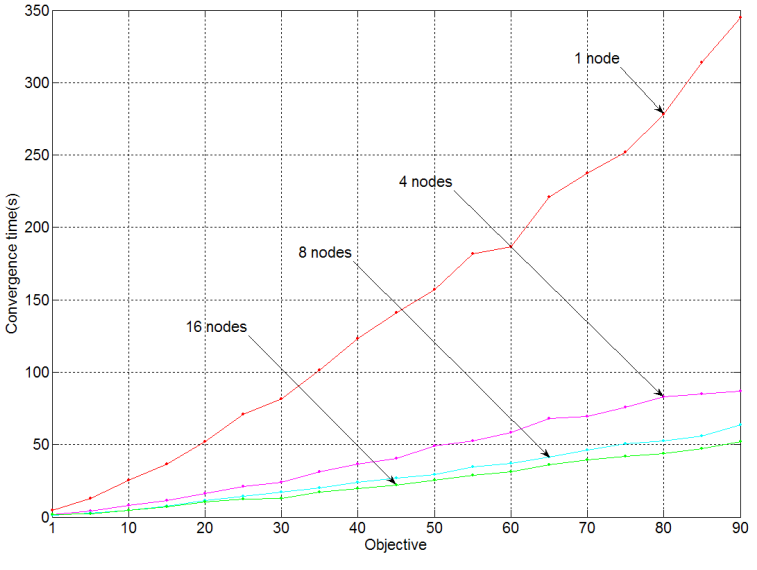


Fig. 11. Convergence time of PMA for different numbers of objectives, large-scale problems using 16 computing nodes of the cluster

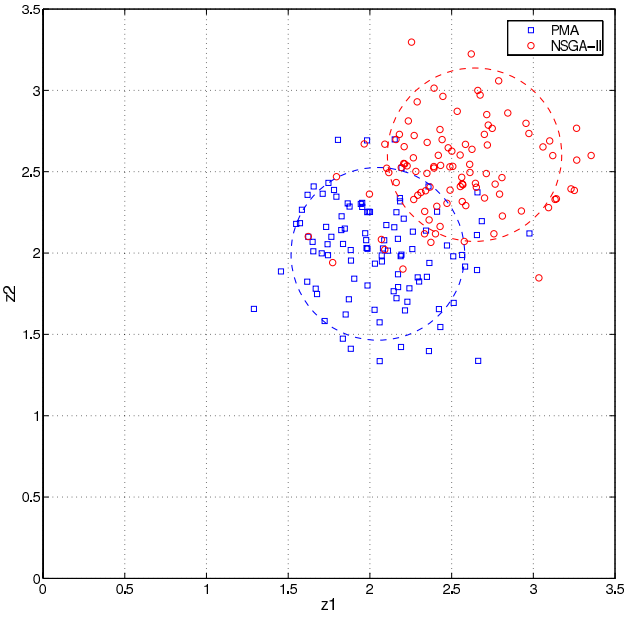


Fig. 12. The two solution spaces of PMA and NSGA-II respectively for 100 random two-objective problems

each algorithm for 100 random two-objective problems (using the same 100 problems for both); the results are presented in Fig. 12.

As marked by the two rings in Fig. 12, the solution space of PMA is closer to the origin than that of NSGA-II. From the experimental results, it can be seen that PMA has better performance than the normal NSGA-II for two-objective distribution problems.

5.6 Comparison between PMA and Other Benchmark MAs (NAs)

As mentioned in Section 1, popular mimetic algorithms (MAs) (or nature-inspired approaches (NAs)) include genetic algorithm (GA), simulated annealing algorithm (SA), ant colony optimization (ACO), particle swarm optimization (PSO), etc. The proposed

Table 7. Common features between PMA and other benchmark MAs

Aspects	Common features
Drawn from	Observations of physical processes that occur in nature
Belong to	The class of meta-heuristics, approximate algorithms
Parallelism	Have inherent parallelism
Performance	Consistently perform well
Fields of application	Artificial intelligence including multi-objective optimization
Solvable problems	All kinds of difficult problems

Table 8. Relative differences between PMA and other benchmark MAs

	PMA	GA	SA	ACO	PSO
Inspired by	Physical models of particle dynamics	Natural evolution	Thermodynamics	Behaviors of real ants	Biological swarm (e.g., swarm of bees)
Key components	Energy function; differential dynamic equations	Chromosomes	Energy function	Pheromone laid	Velocity-coordinate model
Exploration	Both Macro-evolutionary and Micro-evolutionary processes	Macro-evolutionary processes	Micro-evolutionary processes	Macro-evolutionary processes	Macro-evolutionary processes
Dynamics	Can capture the entire dynamics inherent in the problem	Cannot capture	Can capture partly	Cannot capture	Cannot capture
High-dimensional, highly nonlinear, random behaviors and dynamics	Can describe	Cannot describe	Can describe partly	Cannot describe	Cannot describe
Adaptive to problem changes	Fast	Middle	Fast	Low	Middle
Exchange overhead	Low	Middle	Low	Low	Middle

PMA and these benchmark MAs share some common features, which are listed in Table 7.

We also summarize the relative differences between our PMA and the benchmark MAs in Table 8.

6 Conclusion

In this chapter, we propose a novel mimetic approach—particle mechanics algorithm (PMA) for solving multi-objective distribution problems, which is based on the theory of particle mechanics. The approach maps a given distribution problem to the movement of particles in a multi-dimensional space in a pair of (dual) groups of force-fields. The particles move according to certain rules defined by a mathematical model until arriving at a stable state; subsequently, the solution of the multi-objective distribution problem is obtained by anti-mapping the stable state.

Although there are many differences between particles in classical mechanics and those in PMA, we have shown that being inspired by classical mechanics, PMA enables feasible many-objective optimization of problems of very large scale. The PM approach has a low computational complexity, which is crucial for the functioning of large-scale distribution problems.

References

1. Ulungu, E.L., Teghem, J.: The two phases method: an efficient procedure to solve bi-objective combinatorial optimization problems. Journal Foundations of Computing & Decision Sciences 20(2), 149–165 (1995)
2. Tuyttens, D., Teghem, J., Fortemps, P., Van Nieuwenhuyse, K.: Performance of the MOSA method for the bicriteria assignment problem. Journal of Heuristics 6, 295–310 (2000)
3. Pirlot, M.: General local search methods. Europe Journal of Operational Research 92, 493–511 (1996)
4. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science 220(4598), 671–680 (1983)
5. Forrest, S.: Genetic algorithms—principles of natural-selection applied to computation. Science 261(5123), 872–878 (1993)
6. Bonabeau, E., Dorigo, M., Theraulaz, G.: Inspiration for optimization from social insect behaviour. Nature 406(6791), 39–42 (2000)
7. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proc. IEEE Conf. Neural Networks, Piscataway, NJ, vol. IV, 1942–1948 (1995)
8. Durbin, R., Willshaw, D.: An analogue approach to the travelling salesman problem using an elastic net method. Nature 326(6114), 689–691 (1987)
9. Shuai, D., Feng, X.: Distributed Problem Solving in Multi-Agent Systems: A Spring Net Approach. IEEE Intelligent Systems 20(4), 66–74 (2005)
10. Shuai, D., Feng, X.: The Parallel Optimization of Network Bandwidth Allocation Based on Generalized Particle Model. Computer Networks 50(9), 1219–1246 (2006)
11. Le Boudec, J.-Y.: Rate adaptation, Congestion Control and Fairness: A Tutorial (November 22, 2005)
12. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: Afast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)

Part IV

Information Exploited for Local Improvement

Combination of Genetic Algorithms and Evolution Strategies with Self-adaptive Switching

Tatsuya Okabe[1], Yaochu Jin[2], and Bernhard Sendhoff[2]

[1] Honda Research Institute Japan Co., Ltd. (HRI-JP),
8-1 Honcho, Wako-Shi, Saitama, 351-0188, Japan
okabe@jp.honda-ri.com
[2] Honda Research Institute Europe GmbH (HRI-Eu),
Carl-Legien-Strasse 30, 63073 Offenbach/Main, Germany

For tackling an multi-objective optimization problem (MOP), evolutionary computation (EC) gathers much attention due to its population-based approach where several solutions can be obtained simultaneously. Since genetic algorithm (GA) and evolution strategy (ES) are often used in EC, we discuss only GA and ES in this chapter. Although both of them have global and local search capability, theoretical/empirical analysis reveals that GA is rather global search and ES is rather local search on MOP. These facts are related to how to generate offspring, i.e. crossover in GA and mutation in ES. On MOP, the crossover in GA and the mutation in ES generate differently distributed offspring. If mating in the crossover is not restricted, the crossover in GA can generate new offspring globally due to combination of parents which converge different points. Oppositely, the mutation in ES can generate the similar offspring with parent, i.e. locally distributed new offspring, because the offspring is generated by adding normally distributed random values to the parent. Recently, memetic algorithm, which combines GA with local search algorithm, is popular due to its performance. Since ES on MOP works as local search, we combine GA with ES as one of memetic algorithms in this chapter. This algorithm is called as *hybrid representation*. Several issues caused by the combination of GA and ES are discussed, e.g. the discretization error, self-adaptation and adaptive switching. Experiments are conducted on five well-known test functions using six different performance indices. The results show that the hybrid representation exhibits better and more stable performance than the original GA/ES.

1 Introduction

Evolutionary computation (EC), which mimics nature evolution, gathers much attention, in particular, in optimization research field [10, 18]. The main reasons are followings: no gradient information of objective function is necessary; no explicit equation of objective function is requested; it is easy to escape a local optimum; its performance is robust; its usage is easy and so on. Many literature, where EC solves optimization problems with one objective, have been already reported (see [2]).

The EC has three main streams, i.e. genetic algorithm (GA) [10], evolution strategy (ES) [18], evolutionary programming (EP) [9]. Later, genetic programming (GP) was

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 281–307.
springerlink.com

added as a forth direction [13]. Since many books are available for the details of GA, ES, EP and GP, we will not explain the details here[1].

Usually all of these algorithms have a specific representation which is used to encode the design parameters of a particular problem in chromosome, although some can be used in conjunction with a variety of different representations. In EC two types of representations are widely used: the binary representation, which is motivated by the building block hypothesis in GAs [10] and the real-valued representation which is motivated by the idea to choose the natural problem representation especially in the context of ESs [18]. In particular the real-valued representation has frequently been used also with GAs and the separation between GA (binary encoding) and ES (real-valued encoding) is certainly not valid any longer [3, 7, 8, 17]. This research field is called Real-Coded Genetic Algorithms (RCGAs). Note that we use the terms representation and encoding synonymously.

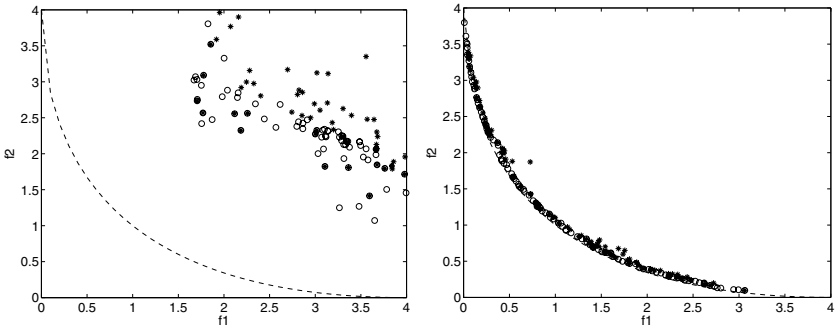
Recently, multi-objective optimization (MOO) is becoming more popular than single objective optimization because most of real-world applications have several objectives which generally conflict with each other. In order to solve multi-objective optimization problems (MOPs), EC are also often used because of its population-based approach. Since EC has several individuals, the set of solutions, which is a target of MOPs, can be obtained in one run. In the literature, a wide range of methods has been proposed; examples are multi-objective genetic algorithm (MOGA), non-dominated sorting genetic algorithm (NSGA), fast and elitist NSGA (NSGA-II), Pareto archived evolution strategy (PAES), see [1, 5] for a comprehensive list.

As one of algorithms for MOPs, we proposed an algorithm called *hybrid representation (HR)* by combining ES and GA [15]. Our original motivation for using hybrid representation was to exploit the different dynamics of populations based on different representations during the search process. These dynamics must be analyzed in both the parameter as well as in the fitness space [14]; they are strongly influenced by the choice of the representation and the variation operators. In [14], the dynamics of ESs based on the real-valued representation were observed by projecting the normal distribution onto the fitness space. Since the ES-mutation is carried out by adding a normally distributed random value to the current parent, a more local search can be realized. In GAs with binary encoding, the main search operator is crossover. By exchanging parts of the chromosomes between several parents, new offspring are generated. It is intuitive that the resulting offspring distribution and therefore, the dynamics will be very different from the one generated by the ES-mutation.

We show two typical snapshots of the different dynamics on test function SCH1 (20 dimensions) in Figure 1 and 2. Here, SCH1 is a test function which can be found in Section 3. In Table 1, the parameters used here are shown. Besides the representations and operators all other parameters of the search process are identical.

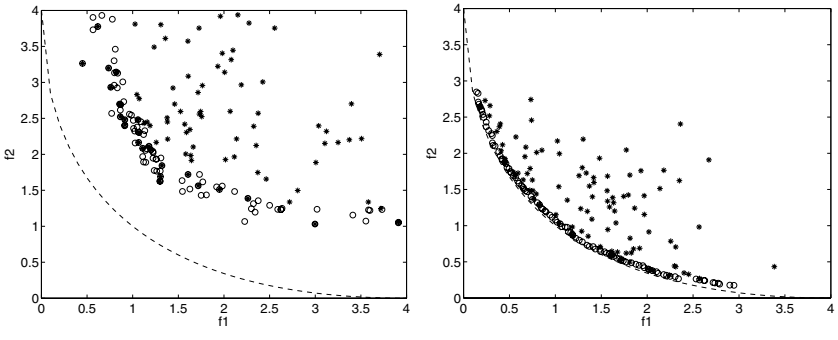
In general in the early generations (exploration phase), the distribution of population should be wider. Following this argument, the distribution of offspring in Figure 2 is better. However, in later generations (exploitation phase), the distribution should be concentrated near the Pareto front. In this case, the distribution of offspring in Figure 1 seems to be better.

[1] In this paper, we will discuss only GA and ES because they are often used in the literature.



(a) Generation 25 (b) Generation 200

Fig. 1. Snapshot of the distribution of parents (circles) and offspring (asterisks) at generation 25 and 200 using real-valued representation and ES-mutation



(a) Generation 25 (b) Generation 200

Fig. 2. Snapshot of the distribution of parents (circles) and offspring (asterisks) at generation 25 and 200 using binary encoding and crossover

Our basic idea is to exploit both dynamics. Using the binary representation we want to realize a wider distribution of offspring in the early stage and using the real-valued representation with the ES-mutation we favor a concentrated distribution of offspring in the later stage to facilitate efficient local search. Since ES works rather as local search, HR can be also classified in memetic algorithms which combine GA with local search.

We will discuss the framework of Hybrid Representation (HR) in Section 2. In Section 3, we will introduce performance indices and test functions to be used for evaluation and compare the HR with two standard methods. As an extension of *HR*, we combine several representations, i.e., binary coding, Gray coding and real-valued representation, in Section 4. In Section 5, we discuss our results and we conclude this chapter in Section 6.

Table 1. Parameters for investigating different dynamics

Figure 1	Representation	Real-valued representation
	Population size	100
	Mutation	ES-mutation. See Section 2.1
	Self-adaptation	Also see Section 2.1
	Recombination	Not used
	Initial step size	$\sigma_i \in [0, 1]$
	Lower threshold	$\sigma_i \geq 0.004 \times \lvert x_i \rvert$
	Selection	Crowded tournament selection by Deb [6]
Figure 2	Representation	Binary encoding
	Population size	100
	Coding	Gray coding
	Crossover	One-point crossover
	Crossover rate	0.9
	Mutation (GA)	Bit flip
	Mutation rate (GA)	0.01
	Number of bits per one design parameter	20
	Selection	Crowded tournament selection by Deb [6]

2 Hybrid Representation

2.1 Representation and Genetic Operators

In the *Hybrid Representation (HR)*, each individual has two chromosomes. One is the binary encoding and the other is the real-valued representation of the design parameters. Additionally, each individual has one special chromosome, called switching chromosome, with two alleles that indicate which representation is to be used, i.e. allele *B* specifies the binary encoding (and respective operators) and *R* the real-valued representation (and ES-mutation).

An example of chromosomes for the hybrid representation is shown in Figure 3. Since both chromosomes have the same value, the encoding is redundant and synchronization is necessary. Note that for each individual either the binary or the real-valued representation is used while the other representation is only updated. Although the analogy with dominant and recessive alleles in biological systems can be seen, it is not entirely correct. In our system the value of the design parameter is always identical, it is the representation of the parameters that changes, which would be like changing the genetic composition of the allele without changing its actual value.

To make sure that the binary and the real-valued chromosomes always encode the same parameter value they have to be synchronized:

1. If the switching chromosome reads *B*, the data in the binary representation are copied to the real-valued representation.
2. If the switching chromosome reads *R*, the data in the real-valued representation are copied to the binary representation. If the value in the real-valued representation is out of the encoding range, the value is set to the nearest boundary.

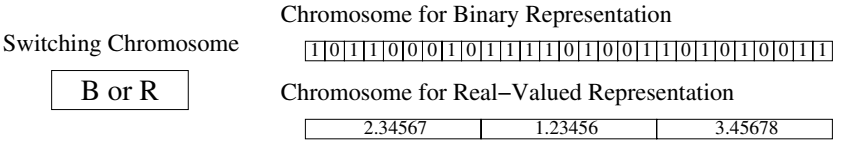


Fig. 3. The hybrid representation using three chromosomes. The alleles of the switching chromosome are *B* or *R*. If the switching chromosome has the allele of *B*, the binary encoding is active and the real-valued representation is inactive, vise versa.

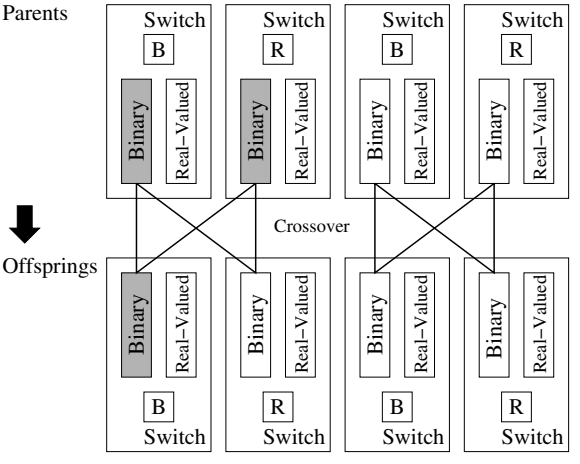


Fig. 4. Crossover in the hybrid representation. Even parents with allele *R* take part in the crossover operation.

In step 2, a discretization error will occur. Furthermore, the values which are out of the encoding range are shifted. On some test functions, this discretization error and the shifting shows significant influence. Both are very important when we compare the performance of the real-valued representation and the binary representation. We will discuss this topic in more detail in Section 5.2.

Associated with the binary representation are the one-point crossover and GA-type mutation, i.e., flipping bits with probability P_m. The crossover operator relies on a sufficient number of parents to act upon, therefore we apply the crossover operator to the whole population irrespective of whether the first chromosome indicates *B* or *R*. Therefore, parents with the value *R* in their switching chromosome also join the crossover operation, see Figure 4. GA-mutation is only applied to individuals with active binary encoding, i.e., with the allele *B* in their switching chromosome.

ES-mutation is applied to individuals with active real-valued representation, thus with the allele *R* in the switching chromosome. Furthermore, following the concept of mutative self-adaptation, the standard deviations of the normal distribution, σ_i, which are called step-sizes or strategy parameters, are also mutated:

$$\sigma_i(t) = \sigma_i(t-1)\exp\left(\tau' z\right)\exp\left(\tau z_i\right);\quad i = 1,\ldots,n, \tag{1}$$

where t and n are the generation number and the dimension of a design parameter, respectively. The variables, z and z_i, are $\mathcal{N}(0,1)$ distributed random values. The parameters, τ and τ' have standard values:

$$\tau = \frac{1}{\sqrt{2n}}, \quad \tau' = \frac{1}{\sqrt{2\sqrt{n}}}. \tag{2}$$

However, the problem with the self-adaptation is that in the case when the active encoding is the binary one, the parameter value is changed without any changes in the strategy parameter. If at a later stage the representation is switched (back) to the real-valued one, the setting of strategy parameters is likely to be inappropriate. In Section 2.3, we will consider this case in more detail.

2.2 Switching Representations

We initialize the switching chromosome of the population with equal probabilities for the alleles B and R, i.e., on average half of the individuals will start with an active binary representation and half with an active real-valued representation.

During the optimization the offspring inherits the switching chromosome with equal probability from either of its parents. In addition, the chromosome is mutated or flipped (from B to R and R to B) with a probability ($P_s = 0.01$). In order to avoid extinction of any of the alleles, at least 5% of the whole population is forced into having an active binary or real-valued representation.

2.3 Self-adaptation in GA-Based Method

In the first trial, the following approximate method is used to adapt the strategy parameters even during binary representation based search. If an offspring is generated by the

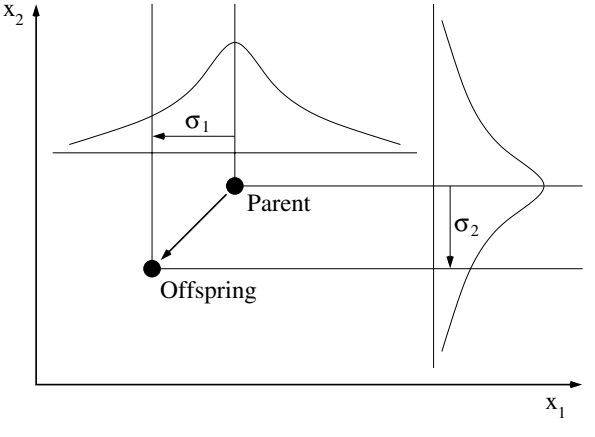


Fig. 5. Self-adaptation of the strategy parameters of ES-mutation during the search process with active binary representation

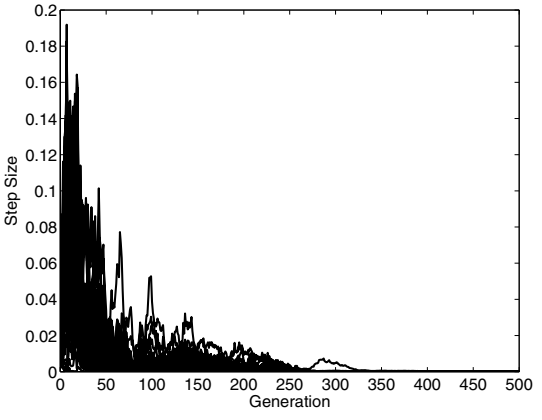


Fig. 6. The history of step sizes on ZDT1 ($n = 50$). The switching chromosome is fixed to B. The average values of 30 runs are shown.

crossover operator, its distance to the parent is measured and directly is used to adjust the strategy parameters σ_i along each coordinate axis, see Figure 5. Since the strategy parameters tend to converge to small values rather quickly the overestimation of the size that might occur using the above outlined method is not harmful to the overall search process.

To see whether this self-adaptation works or not, we fix the switching chromosome to B for all generations and record the history of step sizes in the chromosomes, which is shown in Figure 6 for the ZDT1 ($n = 50$) test function, see Section 3. The plot shows all 50 strategy parameters averaged over 30 runs. It is evident that a "typical" convergence behavior of the step sizes is realized.

2.4 Algorithm

The rough algorithm of HR is shown as below.

1. Initialize parental population.
2. Initialize switching chromosome.
3. Evaluate parental population.
4. Set the generational index $t = 1$.
5. if $(t \leq t_{max})$
   a) Reproduce offspring population.
   b) Mutate switching chromosome.
   c) Compensate the minimum Number of offspring with "B" and "R".
   d) Crossover for all offspring.
   e) Mutation in offspring with "B".
   f) ES-type self-adaptation in the offspring with "R".
   g) ES-type mutation in the offspring with "R".
   h) Synchronize the design parameters according to the switching chromosome.
   i) Self-Adaptation in GA for the offspring with "B".

j) Evaluate offspring population.
k) Crowded tournament selection.
l) Set $t = t + 1$.

6. Output parental population.

In the step (1), parental population is randomly initialized. First, the binary chromosome is initialized. Secondly, the initialized binary chromosome is decoded and the value is stored in the real-value chromosome. By this flow, the chromosomes store the identical design parameters. The strategy parameters are also initialized. The step (2) initializes the switching chromosome, i.e. whether "B" or "R". Then, the parental population is evaluated in the step (3). The generational index t is set as 1 in the step (4). Until t becomes the maximum generation t_{max}, the steps (a)-(l) are repeated in the step (5). The step (a) reproduces offspring population from the parental population. With the mutation rate p_s, the switching chromosome is mutated in the step (b). To keep the minimum number of offspring with "B" and "R", the switching chromosome is checked. If the number of offspring with "B" or "R" is less than the minimum number, the switching chromosome is mutated until the minimum number is kept in the step (c). With all offspring, the crossover is carried out in the step (d) and the mutation is carried out in the offspring with "B" in the step (e). For the offspring with "R", the ES-type self-adaptation and the ES-type mutation are carried out in the steps (f) and (g). According to the switching chromosome, design parameters in the binary chromosome and the real-valued chromosome are synchronized in the step (h). For the offspring with "B", the self-adaptation is carried out in the step (i). The step (j) evaluates the offspring. With the crowded tournament selection, the next parent is selected in the step (k). Then, the generational index is increased by "1". After t becomes t_{max}, the parental population is output as a final solution set in the step (6).

3 Simulation Results

Judging the performance of multi-objective optimization algorithms is not an easy task; this is reflected by the large number of performance indices (PIs) that can be found in the literature [1, 5, 12, 15, 20, 21]. Since no single performance index is sufficient for comparing two algorithms, (See [15] for a criticism of performance indices), it seems most appropriate to use a portfolio of different PIs.

In [15], the performance indices were grouped into several different classes. In this section, six different PIs are used:

- **Deb's γ index [5, 6] :**
  This PI is based on the distance from the solution set, S, to the Pareto optimal solution set, P, to measure the accuracy. The equation of the γ index is given by:

$$\gamma(S,P) = \frac{\sum_{i=1}^{|S|} \left\{ \min_{p \in P} \sqrt{\sum_{k=1}^{M} \left(f_k(s_i) - f_k(p)\right)^2} \right\}}{|S|} \tag{3}$$

  Here, $|S|$ and M are the number of solutions in S and the number of objective functions. The variable s_i shows the components of S.

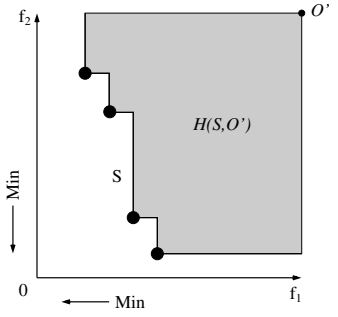


Fig. 7. The definition of the H index in the minimization problem. H is the area generated by the solution set S and the defined origin O', which needs to be specified.

- **Zitzler's H index** [19] **:**
  This PI is based on the size of the area that is dominated by S to measure the accuracy. Figure 7 describes the H index.
- **Deb's Δ' index** [4, 5] **:**
  This PI is for measuring the distribution of S. The Euclidean distances between consecutive solutions, d_i, in S are used. The equation of the Δ' index is given by:

$$\Delta(S)' = \sum_{i=1}^{|S|-1} \frac{|d_i - \bar{d}|}{|S|-1}. \tag{4}$$

  Here, $\bar{d}$ is the average distance of d_i.
- **Zitzler's $\mathscr{M}_2$ index** [19] **:**
  This PI is based on the concept of niching for measuring the distribution. The equation of the $\mathscr{M}_2$ index is given by:

$$\mathscr{M}_2(S) = \frac{1}{|S|-1} \sum_{s_1 \in S} |\{s_2 \in S : \| s_1 - s_2 \| > \sigma\}|. \tag{5}$$

  Here, σ is a niche radius.
- **Zitzler's $\mathscr{M}_3$ index** [19] **:**
  This PI is for measuring the spread of S. The distances between the boundary solutions are calculated. The equation of the $\mathscr{M}_3$ index is given by:

$$\mathscr{M}_3 = \sqrt{\sum_{k=1}^{M} \left\{ \max_{s \in S} f_k(s) - \min_{s \in S} f_k(s) \right\}^2}. \tag{6}$$

- **Hansen's $R1_R$ index** [11] **:**
  This PI is based on a decision maker's (DM) preference. The solution set which is often selected by the DM is the better one. The equation of the $R1_R$ index is given by:

$$R1_R(S,R,U,p) = \int_{u \in U} C(S,R,u) p(u) du. \tag{7}$$

Table 2. Test functions

SCH1	$f_1 = \frac{1}{n}\sum_{i=1}^{n} x_i^2, \quad f_2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - 2.0)^2, \quad -4 \le x_i \le 4$
ZDT1	$f_1 = x_1, \quad g(x_2,...,x_n) = 1.0 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$ $f_2 = g \times (1.0 - \sqrt{\frac{f_1}{g}}), \quad 0 \le x_i \le 1$
ZDT2	$f_1 = x_1, \quad g(x_2,...,x_n) = 1.0 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$ $f_2 = g \times (1.0 - (\frac{f_1}{g})^2), \quad 0 \le x_i \le 1$
ZDT3	$f_1 = x_1, \quad g(x_2,...,x_n) = 1.0 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$ $f_2 = g \times (1.0 - \sqrt{\frac{f_1}{g}} - (\frac{f_1}{g})sin(10\pi f_1)), \quad 0 \le x_i \le 1$
FON2	$f_1 = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right)$ $f_2 = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right)$ $-2 \le x_i \le 2$

Here, R, U and p are a reference solution set, a set of utility functions and an intensity function, respectively. The function of $C(S,R,u)$ is an outcome function of the comparison between S and R which is given by:

$$C(S,R,u) = \begin{cases} 1.0 \text{ if } u^*(S) > u^*(R) \\ 0.5 \text{ if } u^*(S) = u^*(R) \\ 0.0 \text{ if } u^*(S) < u^*(R) \end{cases}, \tag{8}$$

here, $u^*(S) = \max_{s \in S} u(s)$ and $u^*(R) = \max_{r \in R} u(r)$.

For each of the listed indices, the rank of the algorithms is calculated, i.e., *Rank 1* refers to the best optimizer, *Rank 2* to the second best and so on. Thereafter, all six ranks of the six performance indices are averaged ending up with one scalar performance measure for each algorithm.

Five different test functions, i.e., SCH1, ZDT1, ZDT2, ZDT3 and FON2 from [5], are used. Table 2 shows these test functions. For all of these functions, the true Pareto front, PF_{true}, can be determined analytically [14]. We use the test functions with the following dimensions: $n = 2$, $n = 20$ and $n = 50$.

Three optimizers with the binary representation, with the real-valued representation, and with the hybrid representation are executed on the five test functions. Table 3 shows the parameter settings.

Table 3. Parameters in HR

Parameter	Value
Population size	100
Maximum iterations	500
Coding	Gray coding
Crossover	One-point crossover
Crossover rate	0.9
Mutation (GA)	Bit flip
Mutation rate (GA)	0.025
Initial step size (ES)	$\sigma_i \in [0.00, 0.10]$ for SCH1, FON2 $\sigma_i \in [0.00, 0.01]$ for ZDT1, 2, 3
Lower threshold	$\sigma_i \geq 0.004 \times \lvert x_i \rvert$
Minimum number of individuals	5%
Mutation rate (Switching)	0.01
Number of bits per one design parameter	20

Table 4. Parameter settings for PIs

Pareto Solution Set for γ	500 solutions
Reference Solution Set for $R1_R$	100 solutions
Origin for H	(5.0, 5.0) for SCH1
	(1.1, 6.0) for ZDT1, 2, 3
	(1.1, 1.1) for FON2
Niche Radius for $\mathscr{M}_2$	0.03250 for SCH1
	0.00741 for ZDT1
	0.00746 for ZDT2
	0.00920 for ZDT3
	0.00712 for FON2

As the selection operator, we use the crowded tournament selection proposed by Deb [6]. The following steps are carried out: (1) sort offspring by their rank, (2) sort offspring by the crowded distance within the same rank, and (3) select the best offspring deterministically.

The obtained solution sets by the HR are shown in Figure 8. Using the six performance indices (PIs) outlined in [15], we determine the rank of each algorithm. The parameters of all PIs are shown in Table 4. The overall rank, i.e., the average over the six performance indices (which are the median over 30 runs), is shown in Table 5 for each algorithm, for each test function and for each of the three different dimensions ($n = 2, 20, 50$). Since the tendency of superiority and inferiority even with 25% and 75% percentile are similar to the median, only the median will be compared.

First, we compare the binary representation and the real-valued representation. The real-valued representation shows good performance on SCH1, FON2 (except $n = 50$) test functions and for all low dimensional cases. On the other hand, the binary representation shows good performance on the high dimensional cases of ZDT1, ZDT2 and

Table 5. Averaged ranks for each test function

Function	Binary	Real-Valued	Hybrid
SCH1 $n = 2$	2.67	1.67	1.67
SCH1 $n = 20$	2.33	1.50	1.83
SCH1 $n = 50$	2.67	1.83	1.00
ZDT1 $n = 2$	2.33	2.00	1.50
ZDT1 $n = 20$	2.00	2.67	1.33
ZDT1 $n = 50$	2.00	2.50	1.33
ZDT2 $n = 2$	2.33	2.00	1.50
ZDT2 $n = 20$	2.17	2.50	1.17
ZDT2 $n = 50$	2.00	2.67	1.17
ZDT3 $n = 2$	2.33	2.00	1.50
ZDT3 $n = 20$	2.00	2.67	1.33
ZDT3 $n = 50$	1.83	2.67	1.33
FON2 $n = 2$	2.83	1.50	1.67
FON2 $n = 20$	2.50	1.33	2.00
FON2 $n = 50$	1.33	2.17	1.83

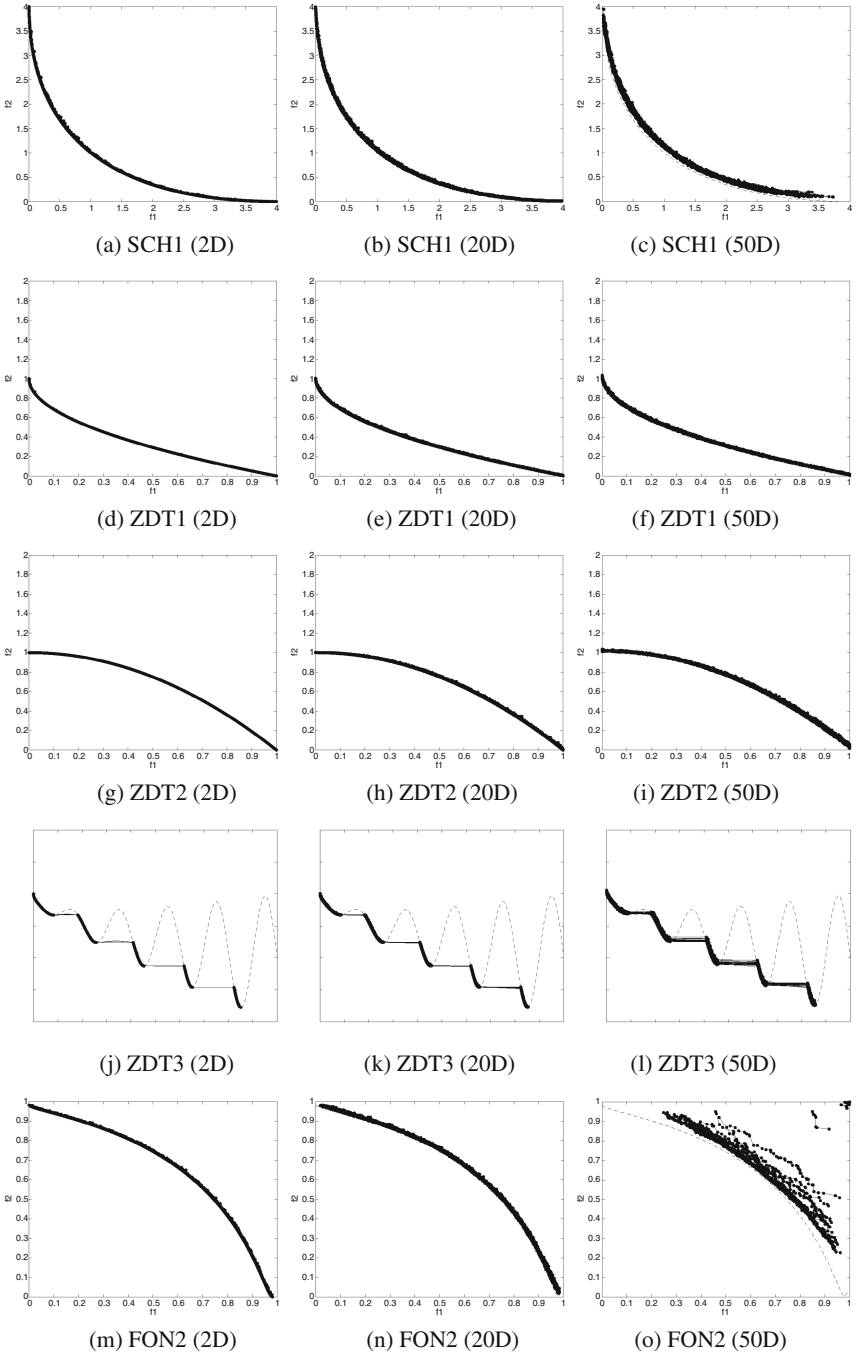
ZDT3 ($n = 20, 50$) and on FON2 ($n = 50$). Apparently, one can classify the five test functions into two classes. The first class is the group of SCH1 and FON2, and the second one is the group of ZDT1, ZDT2 and ZDT3. We will discuss the characteristics of the two classes in Section 5.1.

Second, we compare the *Hybrid Representation (HR)* with both other representations. The averaged ranks are less than or equal to 2. Compared to the binary and the real-valued representations, the high performance of the HR is very stable, i.e., it is the best or close to the best algorithm for all test cases. To analyze the HR, the history of the active representation during the optimization is shown in Figure 9. In order to minimize the stochastic influence as well as the genetic drift on finite populations when there is little selective pressure, the average of 30 identical runs is shown.

The results for 2 dimensions show the clear tendency that the real-valued representation is better than the binary representation. This tendency is in good agreement with the better performance of the real-valued representation in these cases.

For the 20 dimensional cases, the binary representation is selected in the early generations but the real-valued representation is selected in the later generations. Although we did not encourage this behavior, it corresponds to our initial motivation outlined in Section 1.

In the 50 dimensional cases, the results are very similar to the 20 dimensional cases, i.e., the binary representation is selected in the early generation and the real-valued representation in the late generation. However, the tendency for FON2 ($n = 50$) is not so clear. The insufficient result in Figure 8 (o) might be caused by this unclear tendency. If tendency of selected representation is unclear, performance of the HR seems to be not good, and vice versa. We will discuss it again in Section 4.2 and 4.3.



(a) SCH1 (2D) (b) SCH1 (20D) (c) SCH1 (50D)

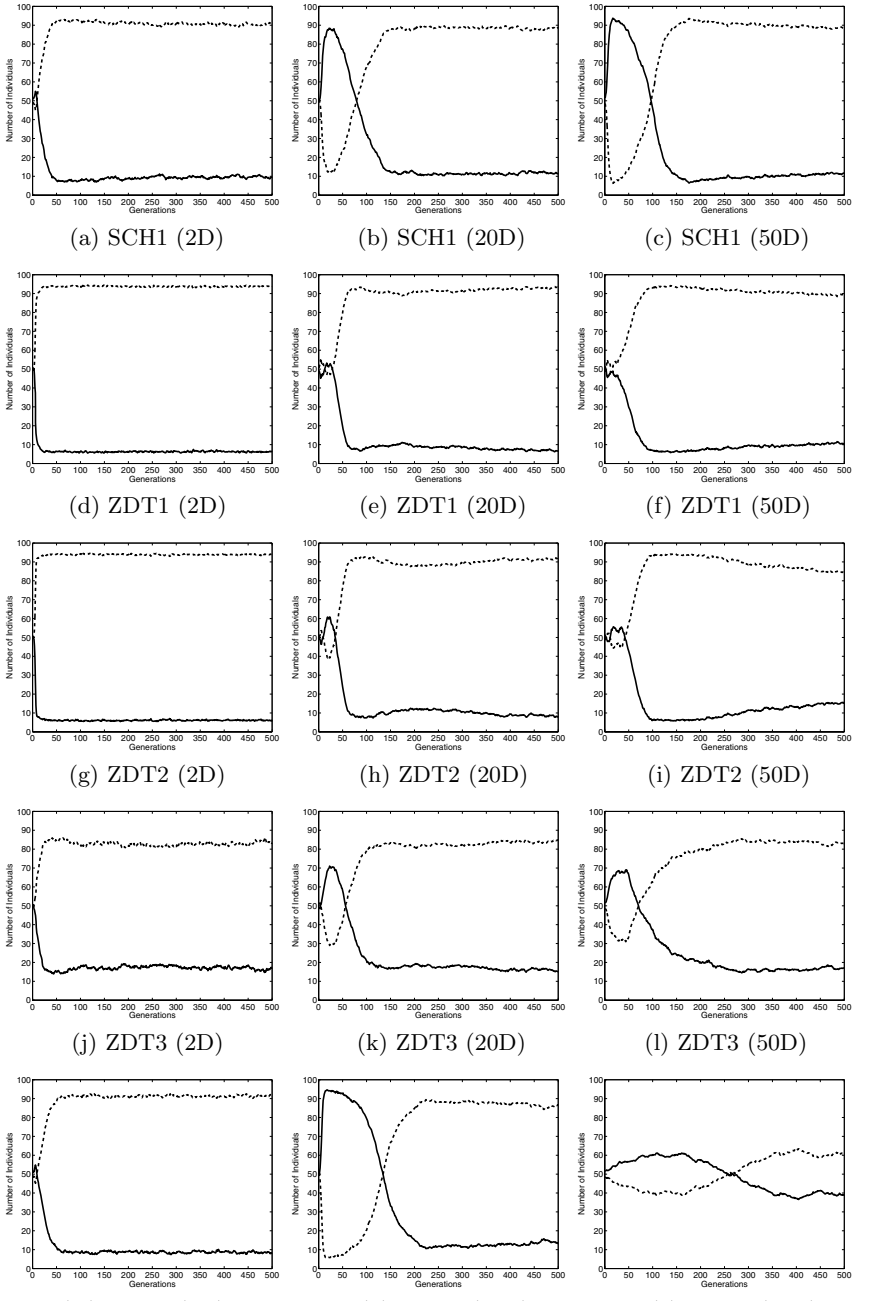
(d) ZDT1 (2D) (e) ZDT1 (20D) (f) ZDT1 (50D)

(g) ZDT2 (2D) (h) ZDT2 (20D) (i) ZDT2 (50D)

(j) ZDT3 (2D) (k) ZDT3 (20D) (l) ZDT3 (50D)

(m) FON2 (2D) (n) FON2 (20D) (o) FON2 (50D)

Fig. 8. Thirty obtained solution sets by the *Hybrid Representation (HR)* (connected by a line). Dotted curves show the boundary of a feasible region.



(a) SCH1 (2D) (b) SCH1 (20D) (c) SCH1 (50D)

(d) ZDT1 (2D) (e) ZDT1 (20D) (f) ZDT1 (50D)

(g) ZDT2 (2D) (h) ZDT2 (20D) (i) ZDT2 (50D)

(j) ZDT3 (2D) (k) ZDT3 (20D) (l) ZDT3 (50D)

(m) FON2 (2D) (n) FON2 (20D) (o) FON2 (50D)

Fig. 9. The history of the allele of the switching chromosome. The average of 30 runs are shown. The x-axis labels the generations and the y-axis the number of individuals in the population with binary representation (solid line) and with real-valued representation (dotted line).

Our proposed HR requests two additional parameters, i.e. mutation rate for switching chromosome P_s, and minimum individuals in a population. In the followings, we discuss the influences of them to the performance.

3.1 Influence of the Switching Mutation Rate P_s

To show the influence of the switching mutation rate P_s on the performance of the HR, only the mutation rate is changed. The other conditions are identical. We use $P_s = 0.000, 0.001, 0.005, 0.010, 0.050, 0.100$. Figure 10 summarizes results for different values P_s. The figure shows the dependency of the three performance indices, i.e., γ, Δ' and $\mathscr{M}_3$, on the mutation rate. Each result is the average of 5 test functions with $n = 2, 20, 50$ and 30 runs. All results are normalized by the value of $P_s = 0.000$ to avoid the influence of differences of absolute values. Additionally, to show all results in one figure, the value of $\mathscr{M}_3$ is shifted, i.e. $-\mathscr{M}_3 + 2$. The performance of the HR is stable in a range of $P_s \in [0.002, 0.010]$.

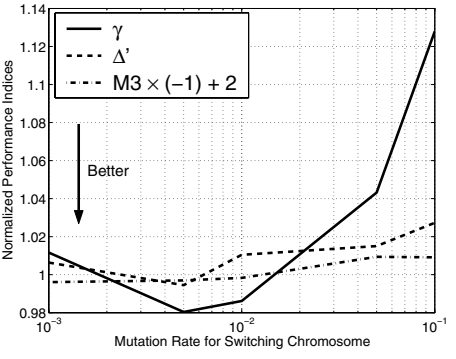


Fig. 10. Dependency of performance indices on the mutation rate. Each line is the average of different test functions, different dimensions and 30 runs. Smaller value is better.

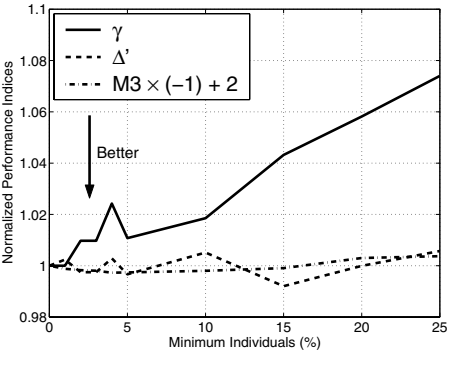


Fig. 11. Dependency of performance indices on the minimum number of individuals. Each line is the average of different test functions, different dimensions and 30 runs. Smaller value is better.

3.2 Influence of the Minimum Number of Individuals

To observe the influence of the minimum number of individuals, we only change the minimum number of individuals, i.e. 0%, 1%, 2%, 3%, 4%, 5%, 10%, 15%, 20% and 25%. The other conditions are identical. The results are shown in Figure 11. Each result is the average of 5 test functions with $n = 2, 20, 50$ and 30 runs. To avoid the influence of absolute values, the normalized values by the result with 0% are used. Additionally, to show all results in one figure, the performance of $\mathscr{M}3$ is modified. Figure 11 tells us that the suitable minimum number of individuals depend on the performance indices. However, less than 10% seems to show the stable performance.

4 Extension of Hybrid Representation

In this section, some possible extensions of the *Hybrid Representation (HR)* will be discussed. Note that the HR in the previous section used the conjunction of the Gray coding and the real-valued representation, we will call this HR the *Original HR* to avoid confusion.

4.1 Binary Coding and Real-Valued Representation

In Section 3, we used the Gray coding in GA. Since the binary coding has the drawback of *Hamming Cliff* [10], the Gray coding is often used for single objective optimization (SOO). In the Gray coding, the similar phenotype has the similar genotype. Although the opposite does not hold, it is believed that the Gray coding is more suitable for optimization.

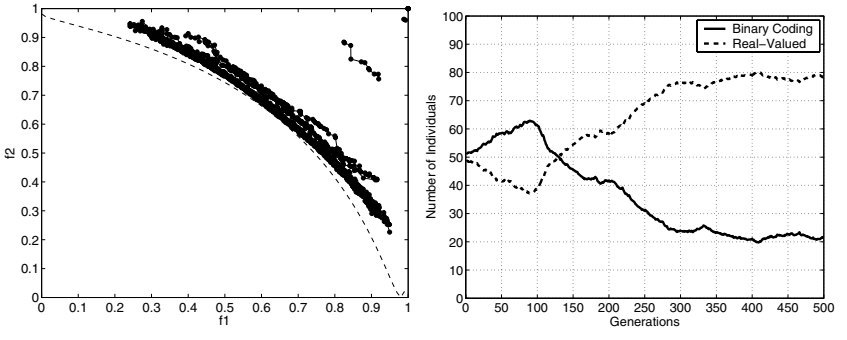
In this section, we apply the binary coding to the proposed HR to investigate the difference between the Gray coding and the binary coding in the HR. We call the HR with the binary coding *HR-BR*.

Except for the coding scheme, we use the same parameter setting in Table 3. The results are shown in Table 6. From Table 6, we cannot observe any big difference between the Gray coding and the binary coding except for the FON2 test function. Although the Gray coding is often used in SOO, the difference seems to be small in MOO, at least on these test functions. Some theoretical work for this issue can be found in [16]. The binary coding seems to be suitable for FON2 test function. The property of FON2 is that the whole feasible region has very low value of probability density function (PDF) except for the $(f_1, f_2) \approx (1.0, 1.0)$ where the value of PDF is nearly infinity [16]. Even if two individuals have different design parameters in the parameter space, both are nearly the same in the fitness space, i.e., $(f_1, f_2) \approx (1.0, 1.0)$. In particular, if two individuals are neighbors in the parameter space, i.e., their design parameters are nearly the same, difference of them in the fitness space cannot be observed at all. In this case, the concept of neighbors in the Gray coding is not necessary any longer. Since the binary coding has higher probability to generate non-neighbor than the Gray coding do, the binary coding seems to be preferable on FON2.

Figure 12 (a) shows the obtained solution set on FON2 ($n = 50$) by HR-BR. The obtained solution set by HR-BR seems to be better than Figure 8 (o) by the original

Table 6. Comparison of the original HR and the HR-BR. Averaged ranks for each test function are shown.

Function	Hybrid (Gray) (Original HR)	Hybrid (Binary) (HR-BR)
SCH1 $n = 2$	1.33	1.67
SCH1 $n = 20$	1.67	1.33
SCH1 $n = 50$	1.17	1.67
ZDT1 $n = 2$	1.33	1.50
ZDT1 $n = 20$	1.33	1.50
ZDT1 $n = 50$	1.50	1.33
ZDT2 $n = 2$	1.33	1.33
ZDT2 $n = 20$	1.83	1.00
ZDT2 $n = 50$	1.33	1.50
ZDT3 $n = 2$	1.33	1.17
ZDT3 $n = 20$	1.67	1.17
ZDT3 $n = 50$	1.17	1.67
FON2 $n = 2$	1.67	1.33
FON2 $n = 20$	1.67	1.17
FON2 $n = 50$	1.67	1.17



(a) Obtained solution set (b) Selected representation

Fig. 12. The result of HR-BR on FON2 ($n = 50$). (a) is the obtained solution set by 30 runs. (b) is the averaged history of the allele of the switching chromosome over 30 runs.

HR. By taking closer look, the number of runs which are trapped near $(1.0, 1.0)$ is less than the one of the original HR. HR-BR seems to easily escape the high PDF area, i.e. $(1.0, 1.0)$, due to the easy generation of non-neighbors. Figure 12 (b) indicates the number of individuals with the switching chromosome "B" and "R". Comparing with Figure 9 (o), the tendency of selected representation is clearer.

4.2 Binary Coding and Gray Coding

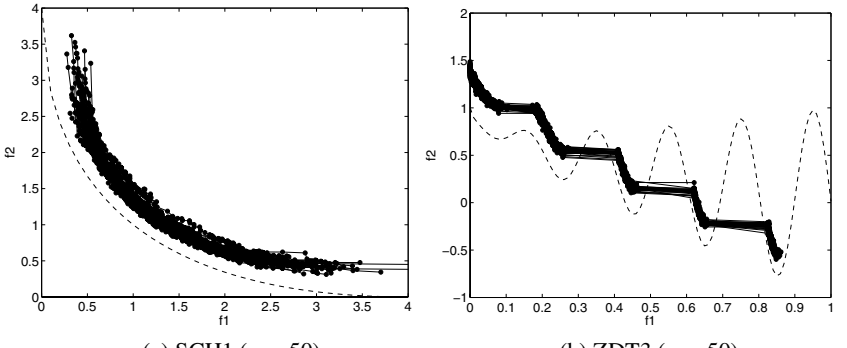
As the second type of the HR, we use the binary coding and the Gray coding in this section, namely *HR-BG*. We modify the original HR as follows:

1. The possible components of switching chromosome are B and G. The component B indicates that the binary coding is active. Oppositely, the component G shows that the Gray coding is active.
2. The real-valued chromosome is replaced by the binary chromosome.

We execute this HR-BG on the same test functions we used before. For each test function and each dimension, we average 30 runs. The set of parameters used is shown in Table 7. Figure 13 shows the sample solution sets on SCH1 ($n = 50$) and ZDT3 ($n = 50$). The results are not better than the results of the original HR. To find out the reasons for this, the number of selected representation, i.e., binary or Gray coding, are plotted in Figure 14.

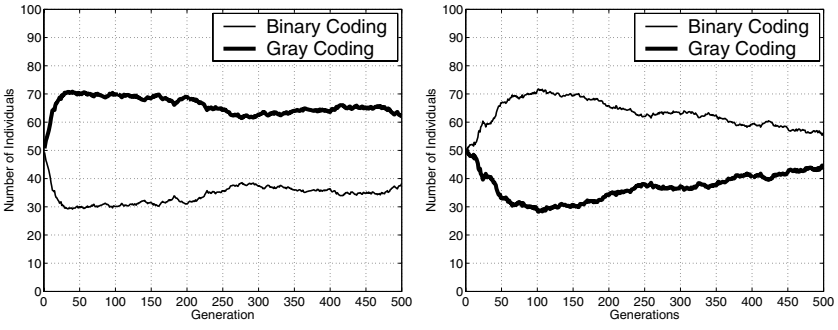
Table 7. Parameters in *HR-BG*

Parameter	Value
Population size	100
Maximum iterations	500
Crossover	One-point crossover
Crossover rate	0.9
Mutation (GA)	Bit flip
Mutation rate (GA)	0.025
Minimum number of individuals	5%
Mutation rate (Switching)	0.01
Number of bits per one design parameter	20



(a) SCH1 ($n = 50$) (b) ZDT3 ($n = 50$)

Fig. 13. Thirty obtained solution sets by *HR-BG* on SCH1 ($n = 50$) and ZDT3 ($n = 50$)



(a) SCH1 ($n = 50$) (b) ZDT3 ($n = 50$)

Fig. 14. History of the number of individuals with the active binary coding and Gray coding. The thinner line shows the number of individuals with the active binary coding and the thicker line shows the one of individuals with the active Gray coding.

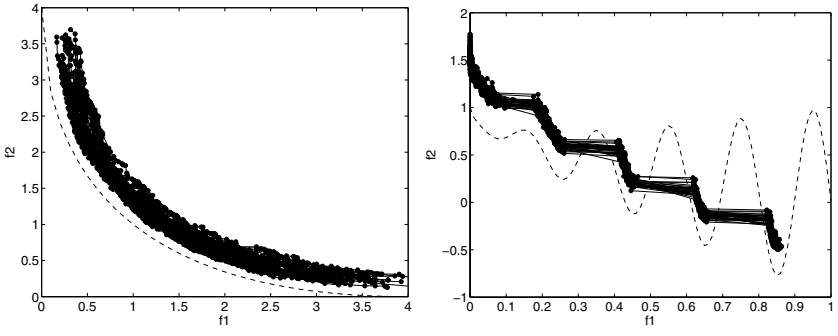
From Figure 14, we cannot observe any clear trend. By comparison of the success case of the original HR in Figure 9, one may say that the proposed HR will show a good performance if the trend of selected representation is clear, vice versa. The unclear trend may be caused by the similar distribution of offspring. If both of the corresponding operators generate the similar offspring, the more suitable operator is not determined. In this situation, only randomly selected representation (or operator) will survive. Thus, we cannot exploit the suitable offspring distribution correctly. This results in poor performance. We will consider again the result of the original HR on FON2 ($n = 50$) in Figure 9 (o). Although our originally proposed method is better than the state-of-the-art MOO algorithms on FON2 ($n = 50$), its performance is not sufficient. Apparently, the tendency of the selected representation on FON2 ($n = 50$) with the original HR, see Figure 9 (o), is also unclear.

4.3 Binary Coding, Gray Coding and Real-Valued Representation

As the third extension, we couple three representations of the binary representation with binary code and with Gray code and the real-valued representation, namely *HR-BGR*.

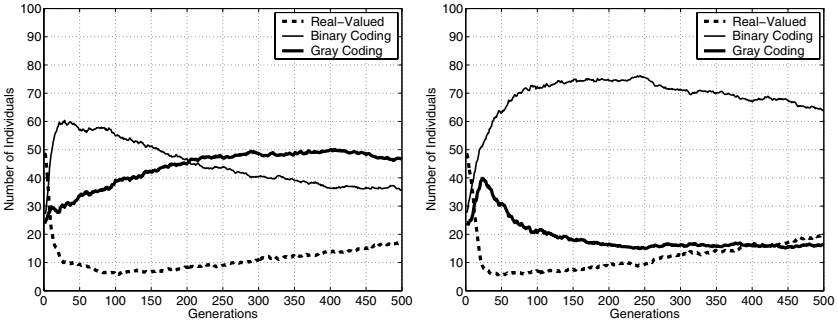
The original HR shows a better performance than the single representation. The HR-BR also shows a better performance. Only the HR-BG algorithm shows a bad performance according to the defined measures. In this section, we combine all of them to observe phenomena. To conduct this coupling, the following minor changes are carried out.

1. The components of the switching chromosome are B, G and R. The components, B, G and R mean the binary coding, the Gray coding and the real-valued representation, respectively.
2. The probabilities for the initial population are 0.25 for B, 0.25 for G and 0.50 for R.



(a) SCH1 ($n = 50$) (b) ZDT3 ($n = 50$)

Fig. 15. Thirty obtained solution sets by *HR-BGR* on SCH1 ($n = 50$) and ZDT3 ($n = 50$).



(a) SCH1 ($n = 50$) (b) ZDT3 ($n = 50$)

Fig. 16. History of selected representation by *HR-BGR* on SCH1 ($n = 50$) and ZDT3 ($n = 50$). The dotted line, the thinner line and the thicker line correspond to the number of individuals with the active real-valued representation, the one with the binary coding and the one with the Gray coding, respectively.

With the same parameter setting as in Table 3, we execute *HR-BGR* 30 times on each test function and each different dimension. The sample solution sets are shown in Figure 15 and the history of the selected methods is shown in Figure 16. Although this method includes the successful combination (B,R) and (G,R), it cannot show the same good performance. Maybe, the bad influence of (B,G), e.g randomly selected method, results in this bad performance.

5 Discussions

5.1 Characteristics of Test Functions

As explained before, the binary representation seems to be particularly suitable for the higher dimensional cases of ZDT1, ZDT2, ZDT3 and FON2. On the other hand, the

real-valued representation is suitable for the low dimensional cases and for the test functions SCH1 and FON2 (except 50 dimensions). In Section 3, we proposed to group the test functions into two classes and to observe their characteristics. If we analyze the Pareto fronts in parameter space (PS), we obtain the following equations:

SCH1

$$\begin{aligned} &x_1 = x_2 = \ldots = x_n, \\ &0.0 \leq x_i \leq 2.0 \ \text{ for } i = 1,2,\ldots,n \end{aligned} \tag{9}$$

ZDT1, ZDT2

$$\begin{aligned} &0 \leq x_1 \leq 1, \\ &x_i = 0 \ \text{ for } i = 2,3,\ldots,n \end{aligned} \tag{10}$$

ZDT3

$$\begin{aligned} &x_1 \in [0.0000, 0.0830], (0.1823, 0.2579], \\ &(0.4095, 0.4541], (0.6187, 0.6528], \\ &(0.8237, 0.8523] \\ &x_i = 0, \ \text{ for } i = 2,3,\ldots,n \end{aligned} \tag{11}$$

FON2

$$\begin{aligned} &x_1 = x_2 = \ldots = x_n, \\ &-1/\sqrt{n} \leq x_i \leq 1/\sqrt{n} \ \text{ for } i = 1,2,\ldots,n. \end{aligned} \tag{12}$$

Compared with each search space, the Pareto front of SCH1 and FON2 in the parameter space is not on the boundary. Whereas the Pareto front of ZDT1, ZDT2 and ZDT3 is on the boundary of the search space.

To observe the difference of the dynamics of binary and real-valued representation, the offspring distribution is calculated theoretically. For uniformly distributed parents in $[0,1]$, the offspring distributions are shown in Figure 17 and Figure 18. Note that the distribution of the binary representation is shown by the respective probabilities whereas for the real-valued representation the probability density function (PDF) $\phi(x)$ is shown.

The distribution of the offspring is very different. The distribution of the binary representation is still uniform in a discrete sense but the distribution of the real-valued representation is not uniform. The probability at the center is higher than at the boundaries. One may expect that the relation between the shape of the offspring distribution and the nature of the Pareto front in the PS explains why some representations are more suitable for a particular problem class.

5.2 Comparison of Binary and Real-Valued Representation

In Section 3, we compared binary, real-valued and hybrid representations. However, the real-valued representation apparently has disadvantages caused by the portion of

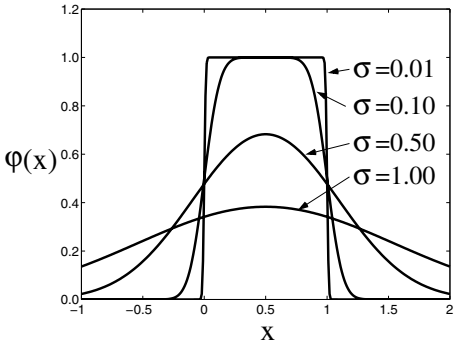


Fig. 17. The offspring distribution by evolution strategies under the assumption of uniformly distributed parents. Here, the probability density function (PDF) is shown.

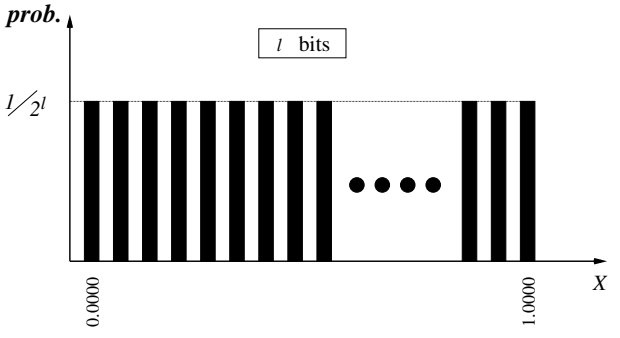


Fig. 18. The offspring distribution by genetic algorithms under the assumption of uniformly distributed parents. Here, the probability is shown.

the Pareto front in the search space (the parameter space). The binary representation can search only discrete points within in the coding range. By setting the number of bits and the coding range, we restrict the search space. If the restricted search space includes the optimal solutions, the binary representation has a strong priori advantage. On the other hand, the real-valued representation is able to search a continuous space without any limitations. If the optimal solutions are not in the coding range for the binary representation, they cannot be identified.

Furthermore, for a fair comparison we have to take the previously mentioned discretization error into account. In order to do this, we add the discretization error to the real-valued representation. After ES-mutation, we convert the real-valued representation to the binary representation and back. This way, we artificially discretize and restrict the search space also for the real-valued representation. The results are shown in Table 8. The results with conversion are denoted as *real-valued (E)*.

The performance of the binary representation is the worst except for FON2 ($n = 50$). In all other cases the real-valued representation with conversion is better. The hybrid representation still exhibits stable superior performance.

Table 8. Averaged ranks on each test function

Function	Binary	Real-Valued (E)	Hybrid
SCH1 $n = 2$	2.67	1.33	2.00
SCH1 $n = 20$	2.33	1.83	1.67
SCH1 $n = 50$	2.67	1.50	1.17
ZDT1 $n = 2$	2.50	1.67	1.33
ZDT1 $n = 20$	2.67	1.67	1.67
ZDT1 $n = 50$	2.33	1.50	1.50
ZDT2 $n = 2$	2.50	1.67	1.33
ZDT2 $n = 20$	3.00	1.67	1.33
ZDT2 $n = 50$	2.67	1.50	1.33
ZDT3 $n = 2$	2.50	1.67	1.33
ZDT3 $n = 20$	2.67	2.00	1.33
ZDT3 $n = 50$	2.33	1.83	1.33
FON2 $n = 2$	2.83	1.33	1.67
FON2 $n = 20$	2.67	1.50	1.83
FON2 $n = 50$	1.33	2.17	1.83

5.3 Another Type of Self-adaptation

In Section 2.3, the self-adaptation in GA was introduced. In this section, a different type of self-adaptation in GA will be used to investigate the importance of self-adaptation.

At each generation, the distribution of offspring is checked. The deviation of it will be used as a step size. Unlike the former self-adaptation based on an individual, this self-adaptation is based on a population. The illustration is drawn in Figure 19.

This HR with the self-adaptation in Figure 19 is called *HR-SA2*. With the same calculation condition in Table 3 except for the self-adaptation, HR-SA2 was executed on the same test functions 30 times. The results of the averaged ranks are shown in Table 9.

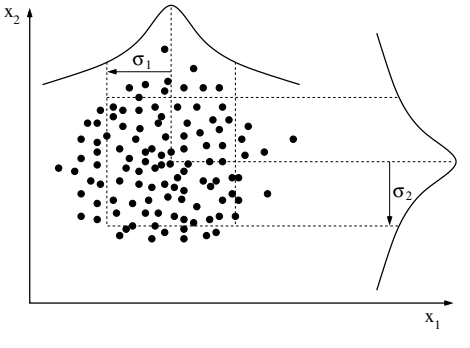


Fig. 19. Another type of self-adaptation in GA. The step size is determined by the deviation of offspring distribution.

Table 9. Comparison of the original HR and the HR with another self-adaptation (HR-SA2). Averaged ranks for each test function are shown.

Function	Hybrid (Original HR)	Hybrid (SA2) (HR-SA2)
SCH1 $n = 2$	1.00	2.00
SCH1 $n = 20$	1.33	1.67
SCH1 $n = 50$	1.00	1.83
ZDT1 $n = 2$	1.33	1.50
ZDT1 $n = 20$	1.17	1.83
ZDT1 $n = 50$	1.17	1.67
ZDT2 $n = 2$	1.33	1.50
ZDT2 $n = 20$	1.00	2.00
ZDT2 $n = 50$	1.00	1.83
ZDT3 $n = 2$	1.00	1.83
ZDT3 $n = 20$	1.17	1.83
ZDT3 $n = 50$	1.17	1.67
FON2 $n = 2$	1.00	2.00
FON2 $n = 20$	1.17	1.83
FON2 $n = 50$	1.17	1.67

Table 10. Comparison of the original HR and the HR with deterministic switching (HR-D). Averaged ranks for each test function are shown.

Function	Hybrid (Original HR)	Hybrid (Deterministic) (HR-D)
SCH1 $n = 2$	1.83	1.17
SCH1 $n = 20$	1.50	1.33
SCH1 $n = 50$	1.83	1.00
ZDT1 $n = 2$	1.50	1.33
ZDT1 $n = 20$	1.50	1.33
ZDT1 $n = 50$	1.33	1.33
ZDT2 $n = 2$	1.83	1.00
ZDT2 $n = 20$	1.33	1.67
ZDT2 $n = 50$	1.33	1.50
ZDT3 $n = 2$	1.50	1.33
ZDT3 $n = 20$	1.50	1.50
ZDT3 $n = 50$	1.83	1.00
FON2 $n = 2$	1.50	1.33
FON2 $n = 20$	1.67	1.33
FON2 $n = 50$	1.67	1.17

Clearly, the original HR is better than HR-SA2 on all test functions. With the comparison of single representation, HR-SA2 seems to be worse than the single representation. This indicates that the self-adaptation in GA is one of the key issues.

5.4 Deterministic Switching

Up to now, we assume a certain priori knowledge about the better representation to be unavailable. In this section, a certain priori knowledge is assumed, i.e. the first half of generations uses the Gray coding and the second half of generations uses the real-valued representation. We call this hybrid representation with deterministic switching as *HR-D*.

We execute HR-D on the same test functions. Except switching method, all calculation conditions are identical. The results are shown in Table 10. Except for ZDT1 ($n = 50$), ZDT2 ($n = 20$ and $n = 50$) and ZDT3 ($n = 20$), the performance of HR-D is better than the original HR. This indicates that an available knowledge of the better representation should be used. However, the results of the above four cases show that a certain priori knowledge sometimes decreases the performance or sometimes has no meaning.

6 Conclusion

In this chapter, we suggested and analyzed a *Hybrid Representation (HR)*, which is one of memetic algorithms, consisting of a binary representation and a real-valued representation of which only one is active in each individual in each generation. We tested the HR on five different functions for different problem space dimensions and measured the performance averaged over six different performance indices for multi-objective optimization. First of all the pragmatic observation can be made that the HR exhibits stable superior performance compared to the other two encodings where only one representation is used. There are different reasons. Our initial motivation for the HR was the observation of different dynamics both theoretically as well as empirically for different encodings. Our intuitive idea to combine a wider search at early generations (binary representation) with a more local search at later generations (real-valued representation) was confirmed by the empirical results of the adaptation of the representation shown in Figure 9. At the same time, the discussion in Section 5 shows that the comparison has to be made with great care because the discretization error combined with specific properties of test functions can produce unexpected effects.

In the HR, two additional parameters were introduced, i.e., the *mutation rate for the switching chromosome (P_s)* and the *minimum number of individuals*. The experiments showed that the recommendable value of the P_s is $0.002 \sim 0.01$ and the one of the *minimum number of individuals* is less than 10% of the population.

We also extended the original HR to different HRs combining the binary coding and the real-valued representation (HR-BR), the binary coding and the Gray coding (HR-BG) and the binary coding, the Gray coding and the real-valued representation (HR-BGR). The HR-BR shows a similar performance as the original HR. Whereas, the HR-BG and the HR-BGR show worse performance than the single representation. The additional investigation showed that the selection mechanism of a suitable representation does not work well due to the similar dynamics.

Acknowledgment

The authors would like to thank Prof. Dr. Edgar Körner (HRI-Eu), Mr. Tomohiko Kawanabe (HRI-JP), Mr. Naoya Watanabe (HRI-JP) and Dr. Tatsuhiko Sekiguchi for their supports and Prof. Dr. Helge Ritter (Bielefeld Univ.) for his comments on this work.

References

1. Coello, C.A.C., Van Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, Dordrecht (2001)
2. http://www.lania.mx/~ccoello/EMOO/
3. Deb, K., Agrawal, R.B.: Simulated Binary Crossover for Continuous Search Space. Complex Systems 9, 115–148 (1995)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II. Technical Report 200001, Indian Institute of Technology, Kanpur Genetic Algorithms Laboratory (KanGAL), Kanpur (2000)
5. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons LTD, Chichester (2001)
6. 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)
7. Deb, K., Anand, A., Joshi, D.: A Computationally Efficient Evolutionary Algorithms for Real-parameter Optimisation. Technical Report 2002003, Indian Institute of Technology, Kanpur Genetic Algorithms Laboratory (KanGAL), Kanpur (2002)
8. Eshelman, L.J., Schaffer, J.D.: Real-coded Genetic Algorithms and Interval-schemata. In: Proceedings of Foundations of Genetic Algorithms, vol. 2, pp. 187–202 (1993)
9. Fogel, L.J.: Autonomous Automata. Industrial Research 4, 14–19 (1962)
10. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and machine Learning. Addison-Wesley, Reading (1989)
11. Hansen, M.P., Jaszkiewicz, A.: Evaluating the quality of approximations to the nondominated set. Technical Report IMM-REP-1998-7, Institute of Mathematical Modeling, Technical University of Denmark (1998)
12. Knowles, J., Corne, D.: On Metrics for Comparing Nondominated Sets. In: Proceedings of Congress on Evolutionary Computation (CEC 2002), pp. 711–716 (2002)
13. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
14. Okabe, T., Jin, Y., Sendhoff, B.: On the Dynamics of Evolutionary Multi-Objective Optimisation. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2002), pp. 247–255 (2002)
15. Okabe, T., Jin, Y., Sendhoff, B.: A Critical Survey of Performance Indices for Multi-Objective Optimisation. In: Proceedings of Congress on Evolutionary Computation (CEC 2003), pp. 878–885 (2003)
16. Okabe, T.: Evolutionary Multi-Objective Optimization - On the Distribution of Offspring in Parameter and Fitness Space. Sharker Verlag (2004)
17. Ono, I., Kobayashi, S.: A Real-coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover. In: Proceedings of the Seventh International Conference on Genetic Algorithms, pp. 246–253 (1997)
18. Rechenberg, I.: Evolutionsstrategie 1994, Frommann-Holzboog (1994)

19. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation 8(2), 173–195 (2000)
20. Zitzler, E., Laumanns, M., Thiele, L., Fonseca, C.M., Grunert da Fonseca, V.: Why Quality Assessment of Multiobjective Optimizers Is Difficult. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2002), pp. 666–673 (2002)
21. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Transactions on Evolutionary Computation 7(2), 117–132 (2003)

Comparison between MOEA/D and NSGA-II on the Multi-Objective Travelling Salesman Problem

Wei Peng[1,⋆], Qingfu Zhang[2], and Hui Li[2]

[1] School of Computer Science and Technology
National University of Defense Technology
Changsha, Hunan, 410073, China
wpeng@nudt.edu.cn
[2] Department of Computing and Electronic Systems
University of Essex
Colchester, Essex, CO4 3SQ, UK
qzhang, hlil@essex.ac.uk

Most multiobjective evolutionary algorithms are based on Pareto dominance for measuring the quality of solutions during their search, among them NSGA-II is well-known. A very few algorithms are based on decomposition and implicitly or explicitly try to optimize aggregations of the objectives. MOEA/D is a very recent such an algorithm. One of the major advantages of MOEA/D is that it is very easy to design local search operator within it using well-developed single-objective optimization algorithms. This chapter compares the performance of MOEA/D and NSGA-II on the multiobjective travelling salesman problem and studies the effect of local search on the performance of MOEA/D.

1 Introduction

A multiobjective optimization problem (MOP) can be stated as follows:

$$\begin{aligned} &\text{minimize } F(x) = (f_1(x), \ldots, f_m(x)) \\ &\text{subject to} \qquad\quad x \in \Omega \end{aligned} \tag{1}$$

where Ω is the decision (variable) space, R^m is the objective space, and $F : \Omega \rightarrow R^m$ consists of m real-valued objective functions.

Very often, no single solution can optimize all the objectives in a MOP since these objectives conflict each other. Pareto optimal solutions, which characterize optimal trade-offs among these objectives, are of practical interest in many real-life applications. A solution is called Pareto optimal if any improvement in one single objective must lead to deterioration in at least one other objective. The set of all the Pareto optimal solutions in the objective space is called the Pareto front (PF). Many MOPs may have a huge (or even infinite) number of Pareto optimal solutions. It is very time-consuming, if

⋆ This work was carried out during the first author's sabbatical leave at University of Essex as an academic visitor.

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 309–324.
springerlink.com

possible, to obtain the complete PF. Multiobjective evolutionary algorithms (MOEAs) are designed to find a set of representative Pareto solutions.

The majority of existing MOEAs are based on Pareto dominance [1]-[3]. In these algorithms, the utility of each individual solution is mainly determined by its Pareto dominance relations with other solutions visited in the previous search. Since using Pareto dominance alone could discourage the diversity, some techniques such as fitness sharing and crowding have often been used as compensation in these MOEAs. Arguably, NSGA-II [4] is the most popular Pareto dominance based MOEAs. The characteristic feature of NSGA-II is its fast non-dominated sorting procedure for ranking solutions in its selection.

A Pareto optimal solution to a MOP could be an optimal solution of a scalar optimization problem in which the objective is an aggregation function of all the individual objectives. Therefore, approximation of the Pareto front can be decomposed into a number of scalar objective optimization subproblems. This is a basic idea behind many traditional mathematical programming methods for approximating the PF. A very small number of MOEAs adopt this idea to some extent, among them the Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D) is a very recent one [5]. MOEA/D attempts to optimize these subproblems simultaneously. The neighborhood relations among these subproblems are defined based on the distances between their aggregation coefficient vectors. Each subproblem (i.e., scalar aggregation function) is optimized in MOEA/D by using information only from its neighboring subproblems. One of the major advantages of MOEA/D over Pareto dominance based MOEAs is that single objective local search techniques can be readily used in MOEA/D.

We believe that comparison studies between MOEAs based on Pareto dominance and those using decomposition on different multiobjective optimization problems could be very useful for understanding the strengths and weaknesses of these different methodologies and thus identifying important issues which should be addressed in MOEAs.

This chapter proposes an implementation of MOEA/D for the multiobjective travelling salesman problem and compares it with NSGA-II. The effect of local search on the performance of MOEA/D has also been experimentally studied. The chapter is organized as follows. Section 2 presents the multiobjective travelling salesman problem and the local search method used in our experiments. Section 3 presents MOEA/D and NSGA-II. Section 4 gives the experimental setting and performance metrics. Section 5 presents the experimental results. Finally, section 6 concludes the chapter.

2 Multiobjective Travelling Salesman Problem

2.1 Problem

Given a number of cities and the cost of travel between each pair of them, the travelling salesman problem is to find the cheapest tour of visiting each city exactly once and returning to the starting point. The single objective TSP is NP-hard [6].

Mathematically, in the multiobjective TSP (mo-TSP), the decision space Ω is the set of all the permutations of $1,2,\ldots,n$, and the m objectives to minimize are:

$$\begin{aligned} f_1(x_1,\ldots,x_n) &= \sum_{i=1}^{n-1} d^1_{x_i,x_{i+1}} + d^1_{n,1} \\ &\vdots = \vdots \\ f_m(x_1,\ldots,x_n) &= \sum_{i=1}^{n-1} d^m_{x_i,x_{i+1}} + d^m_{n,1} \end{aligned} \tag{2}$$

where $x = (x_1,\ldots,x_n)$ is a permutation vector, and $d^k_{i,j}$ can be regarded as the travel cost from city i to j in the k-th objective.

The mo-TSP has been used as a benchmark problem in studying the performance of MOEAs in recent years [7]-[9].

2.2 2-opt Local Search for the TSP

The 2-opt local search is a very popular and efficient local search method for improving a solution in the single objective TSP problem. A 2-interchange move on a solution (tour) to the TSP is to break it into two paths by deleting two edges and reconnect them in the other possible way. The neighborhood of a solution x consists of all the solutions which can be obtained by applying a 2-interchange move on it. In order to improve a solution x, 2-opt local search searches the neighborhood of x to find a solution y which is better than x and then replace x by y. This process is repeated until no solution in the neighborhood of x is better than y or a predefined stopping condition is met.

Comparisons of the costs of two neighboring solutions are major computational overhead in 2-opt local search. Since 2 neighboring solutions are different only in 2 edges, the cost difference of these two solutions can be computed with several basic operations. Therefore, the computational overhead of 2-opt local search is often not high.

3 MOEA/D

3.1 Weighted Sum Approach

MOEA/D requires a decomposition approach for converting approximation of the PF of (1) into a number of single objective optimization problems. In principle, any decomposition approach can serve for this purpose. In the chapter, we use the weighted sum approach [10]. This approach considers a convex combination of the different objectives. Let $\lambda = (\lambda_1,\ldots,\lambda_m)^T$ be a weight vector, i.e. $\lambda_i \geq 0$ for all $i = 1,\ldots,m$ and $\sum_{i=1}^m \lambda_i = 1$. Then the optimal solution to the following scalar optimization problem

$$\begin{aligned} &\text{minimize } g(x|\lambda) = \textstyle\sum_{i=1}^m \lambda_i f_i(x) \\ &\text{subject to} \qquad x \in \Omega \end{aligned} \tag{3}$$

is a Pareto optimal solution to (1), where we use $g(x|\lambda)$ to emphasize that λ is a coefficient vector in this objective function while x is the variables to be optimized. To generate a set of different Pareto optimal vectors, one can use different weight vectors λ in the above scalar optimization problem. If the PF is convex (concave in the case

of maximization), this approach could work well. However, not every Pareto optimal vector can be obtained by this approach in the case of nonconvex PFs. To overcome these shortcomings, some effort has been made to incorporate other techniques such as ε-constraint into this approach, more details can be found in [10].

3.2 General Framework

MOEA/D needs to decompose the MOP under consideration. In the following, we employ the weighted sum approach. It is very trivial to modify the following MOEA/D when other decomposition methods are used.

Let $\lambda^1,\ldots,\lambda^N$ be a set of evenly spread weight vectors. The problem of approximation of the PF of (1) can be decomposed into N scalar optimization subproblems by using the weighted sum approach and the objective function of the j-th subproblem is:

$$g(x|\lambda^j) = \sum_{i=1}^{m} \lambda_i^j f_i(x) \tag{4}$$

where $\lambda^j = (\lambda_1^j,\ldots,\lambda_m^j)^T$. MOEA/D minimizes all these N objective functions simultaneously in a single run.

Note that g is continuous of λ, the optimal solution of $g(x|\lambda^i)$ should be close to that of $g(x|\lambda^j)$ if λ^i and λ^j are close to each other. Therefore, any information about these g's with weight vectors close to λ^i should be helpful for optimizing $g(x|\lambda^i)$. This is a major motivation behind MOEA/D.

In MOEA/D, a neighborhood of weight vector λ^i is defined as a set of its several closest weight vectors in $\{\lambda^1,\ldots,\lambda^N\}$. The neighborhood of the i-th subproblem consists of all the subproblems with the weight vectors from the neighborhood of λ^i. Each subproblem has its best solution found so far in the population. Only the current solutions to its neighboring subproblems are exploited for optimizing a subproblem in MOEA/D.

At each generation t, MOEA/D with the weighted sum approach maintains:

- a population of N points $x^1,\ldots,x^N \in \Omega$, where x^i is the current solution to the i-th subproblem;
- $FV^1,\ldots,FV^N$, where FV^i is the F-value of x^i, i.e., $FV^i = F(x^i)$ for each $i = 1,\ldots,N$;
- an external population EP, which is used to store nondominated solutions found during the search.

The algorithm works as follows:

Input:
- MOP (1);
- a stopping criterion;
- N : the number of the subproblems considered in MOEA/D;
- a uniform spread of N weight vectors: $\lambda^1,\ldots,\lambda^N$;
- T : the number of the weight vectors in the neighborhood of each weight vector.

Output: EP.

Step 1: Initialization

Step 1.1: Set $EP = \emptyset$.

Step 1.2: Compute the Euclidean distances between any two weight vectors and then work out the T closest weight vectors to each weight vector. For each $i = 1,\ldots,N$, set $B(i) = \{i_1,\ldots,i_T\}$ where $\lambda^{i_1},\ldots,\lambda^{i_T}$ are the T closest weight vectors to λ^i.

Step 1.3: Generate an initial population $x^1,\ldots,x^N$ randomly or by a problem-specific method. Set $FV^i = F(x^i)$.

Step 2. Update

For $i = 1,\ldots,N$, do

Step 2.1 Reproduction: Randomly select two indexes k,l from $B(i)$, and then generate a new solution y from x^k and x^l by using genetic operators.

Step 2.2 Improvement: Apply a problem-specific repair/improvement heuristic on y to produce y'.

Step 2.3 Update of Neighboring Solutions: For each index $j \in B(i)$, if $g(y'|\lambda^j) \leq g(x^j|\lambda^j)$, then set $x^j = y'$ and $FV^j = F(y')$.

Step 2.4 Update of EP:

Remove from EP all the vectors dominated by $F(y')$.

Add $F(y')$ to EP if no vectors in EP dominate $F(y')$.

Step 3. Stopping Criteria If stopping criteria is satisfied, then stop and output EP. Otherwise go to **Step 2**.

In initialization, $B(i)$ contains the indices of the T closest vectors of λ^i. We use the Euclidean distance to measure the closeness between any two weight vectors. Therefore, λ^i's closest vector is itself and then $i \in B(i)$. If $j \in B(i)$, the j-th subproblem can be regarded as a neighbor of the i-th subproblem.

In the i-th pass of the loop in Step 2, the T neighboring subproblems of the i-th subproblem are considered. Since x^k and x^l in Step 2.1 are the current best solutions to neighbors of the i-th subproblem, their offspring y should hopefully be a good solution to the i-th subproblem. In Step 2.2, a problem-specific heuristic is used to repair y in the case when y invalidates any constraints, and/or optimize the i-th g. Therefore, the resultant solution y' is feasible and very likely to have a lower function value for the neighbors of i-th subproblem. Step 2.3 considers all the neighbors of the i-th subproblem, it replaces x^j with y' if y' performs better than x^j with regard to the j-th subproblem. FV^j is needed in computing the value of $g(x^j|\lambda^j)$ in Step 2.3. Step 2.4 updates the external population.

Step 2.2 allows MOEA/D to be able to make use of a scalar optimization method very naturally. One can take the $g(x|\lambda^i)$ as the objective function in the heuristic in Step 2.2. Although it is one of the major features of MOEA/D, Step 2.2 is not a must in MOEA/D, particularly if Step 2.1 can produce a feasible solution.

3.3 Improvement Heuristic

As mentioned in previous section, we use the 2-opt local search heuristic process to improve the solutions obtained by genetic operators. One of the key issues in 2-opt local search is how to search the neighborhood. In our implementation, we adopt the first improvement strategy, i.e. search the neighboring solutions in a predetermined order

and y is the first solution found which is better than x. To limit the computational cost, the number of neighborhoods the 2-opt local search searches is not allowed to exceed a predefined limit, L_s. The local search process is operated on one solution s_0 and is illustrated as follows.

Step 1. Randomly start from one edge in the tour of s_0. Traverse the tour, find another edge with which a better tour can be obtained by exchanging it with the starting edge. Check the next edge after exchanging.
Step 2. If no edge can be found for the starting edge, then let the next edge be the starting edge and repeat Step 1).
Step 3. Stopping Criteria: if a predefined limit on the number of exchanging is reached or all edges are checked, then stop.

3.4 NSGA-II

NSGA-II [4] is a very popular MOEA based on Pareto domination. NSGA-II maintains a population P_t of size N at generation t and generate P_{t+1} from P_t in the following way.

Step 1. Use selection, crossover and mutation to create an offspring population Q_t from P_t.
Step 2. Choose N best solutions from $P_t \cup Q_t$ to form P_{t+1}.

The characteristic of NSGA-II is embodied in a fast nondominated sorting and crowding-distance estimation procedure for comparing different qualities of different solutions in **Step 2** and selection in **Step 1**. The computational complexity of each generation in NSGA-II is $O(mN^2)$, where m is the number of the objectives and N is its population size.

For a fair comparison, an external population is used in our implementation of NSGA-II to record all non-dominated solutions found in the search process.

4 Experiment Settings

4.1 Multiobjective TSP Instances

In this chapter, we only consider two objectives. To generate a bi-objective test instance with n cities, a random distribution of cities in a $[0,n] \times [0,n]$ area is generated for each objective. The travel cost between any two cities is set to be the distance between them. We have tested three different instances in which the numbers of cities are 500, 1000 and 1500.

4.2 Parameter Settings

In our experiments, we tested three algorithms, namely, MOEA/D without local search, MOEA/D with local search, and NSGA-II. The experiments were carried out in a desktop PC with Intel Pentium 4 CPU 3.20GHz and 1.50GB RAM. Table 1 gives the parameter settings used in the experiments.

Table 1. Experimental parameter Settings

Genetic operator in three algorithms	inver-over crossover [8],
δ(Random inverse rate in inver-over crossover)	0.02,
Number of runs for each instances	10,
n (The number of cities)	500,1000,1500,
N (The population size in all the algorithms)	100,
Weight vectors in MOEA/D	100 uniformly distributed vectors
T in MOEA/D	25,
L_s in local Search	100,
CPU time used in each run (in second)	600 for the instance with 500 cities, 1600 for 1000 cites, and 4000 seconds for 1,500 cities.

4.3 Performance Metrics

Due to the nature of multiobjective optimization, no single performance metric is always able to compare the performances of different algorithms properly. In our experiments, we use the following two metrics:

- **Set Coverage (C-metric):** Let A and B be two approximations to the PF of a MOP, $C(A,B)$ is defined as the percentage of the solutions in B that are dominated by at least one solution in A, i.e.

$$C(A,B) = \frac{|\{u \in B | \exists v \in A : v \, dominates \, u\}||}{|B|} \tag{5}$$

  $C(A,B)$ is not necessarily equal to $1-C(B,A)$. $C(A,B) = 1$ means that all solutions in B are dominated by some solutions in A, and $C(A,B) = 0$ means that no solution in B is dominated by a solution in A.

- **Distance from Representatives in the PF(D-metric):** Let P^* be a set of uniformly distributed points along the PF. Let A be an approximation to the PF, the average distance from A to P^* is defined as: [5]

$$D(A,P^*) = \frac{\sum_{v \in P^*} d(v,A)}{|P^*|} \tag{6}$$

  where $d(v,A)$ is the minimum Euclidean distance between v and the points in A. If $|P^*|$ is large enough to represent the PF very well, $D(A,P^*)$ could measure both the diversity and convergence of A in a sense. To have a low value of $D(A,P^*)$, set A must be very close to the PF and cannot miss any part of the whole PF.

  As we do not know the actual PFs of the test instances, we use an approximation of the PF as P^*. The approximation of PF is obtained from all non-dominated solutions found in all the runs of the three algorithms.

Table 2. C-metric vs execution time

The instances	exec time (seconds)	NSGA-II vs MOEA/D	MOEA/D vs NSGA-II
500 cities	100	0.0	0.954062
	200	0.0	0.942587
	300	0.0	0.881377
	400	0.0	0.852883
	500	0.0	0.844775
	600	0.0	0.829717
1000 cities	200	0.0	0.997921
	400,600,...,1600	0.0	1.0
1500 cities	500,1000,...,4000	0.0	1.0

Table 3. C-metric vs. the number of function evaluations

The instances	number of function evaluations	NSGA-II vs MOEA/D	MOEA/D vs NSGA-II
500 cities	0.5×10^6	0.968367	0.011017
	1.0×10^6	0.898114	0.031785
	1.5×10^6	0.761665	0.117353
	2.0×10^6	0.220869	0.402011
	2.5×10^6	0.009831	0.569514
	3.0×10^6	0.0	0.624579
1000 cities	0.5×10^6	0.511688	0.252936
	1.0×10^6	0.009333	0.614016
	1.5×10^6	0.0	0.745755
	2.0×10^6	0.0	0.810331
	2.5×10^6	0.0	0.857586
	3.0×10^6	0.0	0.903073
	3.5×10^6	0.0	0.927789
	4.0×10^6	0.0	0.947508
	4.5×10^6	0.0	0.961898
	5.0×10^6	0.0	0.979815
1500 cities	1.0×10^6	0.0	0.826044
	2.0×10^6	0.0	0.945171
	3.0×10^6	0.0	0.973439
	4.0×10^6	0.0	0.988983
	5.0×10^6	0.0	0.997826
	6.0×10^6	0.0	1.0
	7.0×10^6	0.0	1.0
	8.0×10^6	0.0	1.0

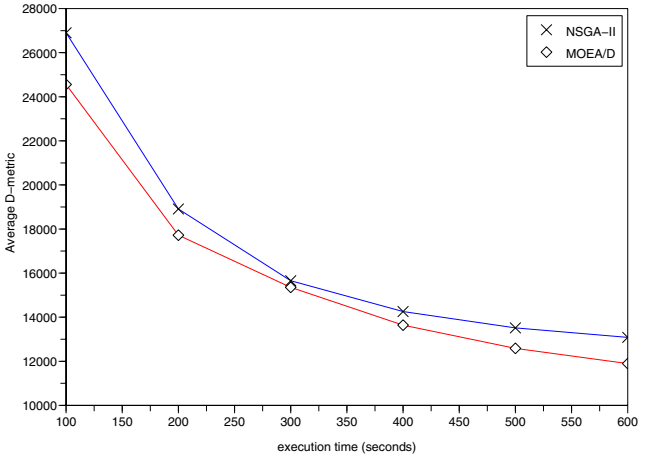


Fig. 1. D-metric vs. execution time in the instance with 500 cities

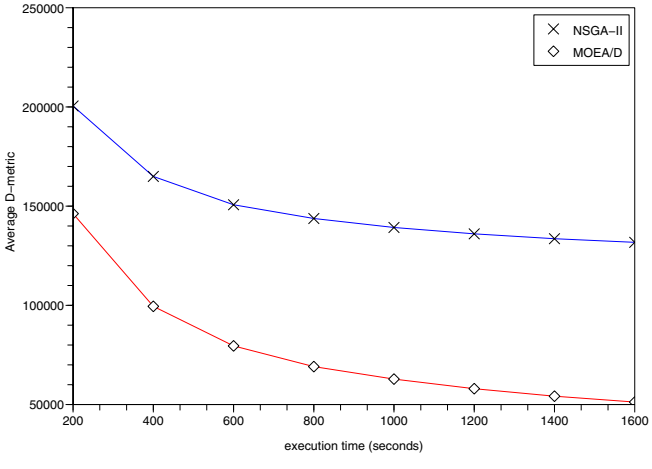


Fig. 2. D-metric vs. execution time in the instance with 1000 cities

5 Experiment Results

5.1 Comparison of MOEA/D and NSGA-II

We firstly compare the performance of MOEA/D algorithm without local search and the NSGA-II algorithm. Note that both algorithms use the same genetic operator.

5.1.1 C-Metric

Table 2 presents the average C-metrics of MOEA/D and NSGA-II on different test instances. It is evident from this table that no solution obtained in NSGA-II dominates

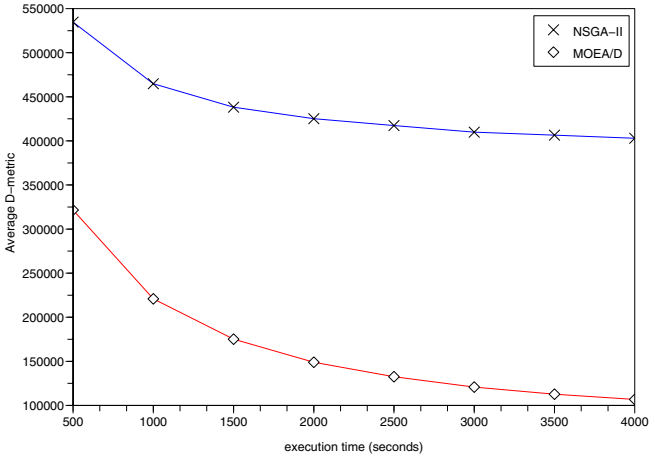


Fig. 3. D-metric vs. execution time in the instance with 1500 cities

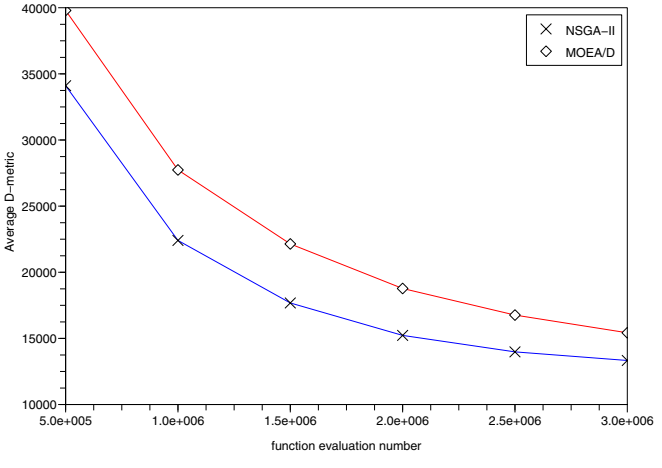


Fig. 4. D-metric vs. the number of function evaluations in the instance with 500 cities

any solutions in MOEA/D at these selected observation times. Most solutions in NSGA-II are dominated by the solutions in MOEA/D in the test instance with 500 cities and all the solutions in NSGA-II are dominated by those in MOEA/D in the other two larger test instances while $n = 1000$ and 1500. A very interesting observation is that the C-metric of MOEA/D versus NSGA-II decreases as the execution time increases in the instance with 500 cities. But it does not happen in the other two larger instances. This implies that MOEA/D, compared with NSGA-II, is very promising in dealing with large scale problems.

In some real-world applications, the number of function evaluations matters. Table 3 gives the C-metrics of these two algorithms versus the number of function evaluations. Clearly, MOEA/D outperforms NSGA-II after a number of function evaluations in all the three instances.

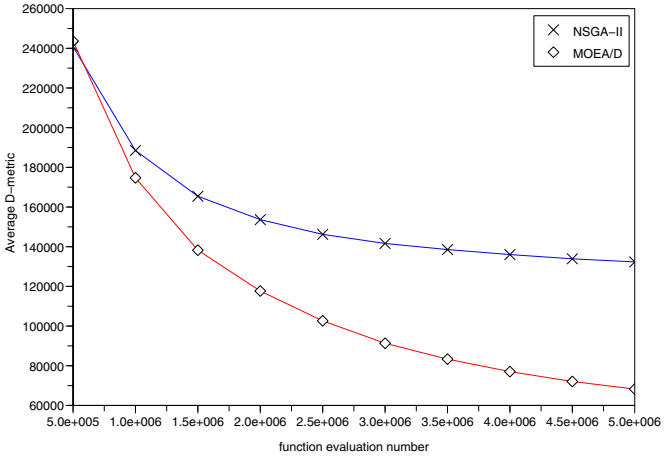


Fig. 5. D-metric vs. the number of function evaluations in the instance with 1000 cities

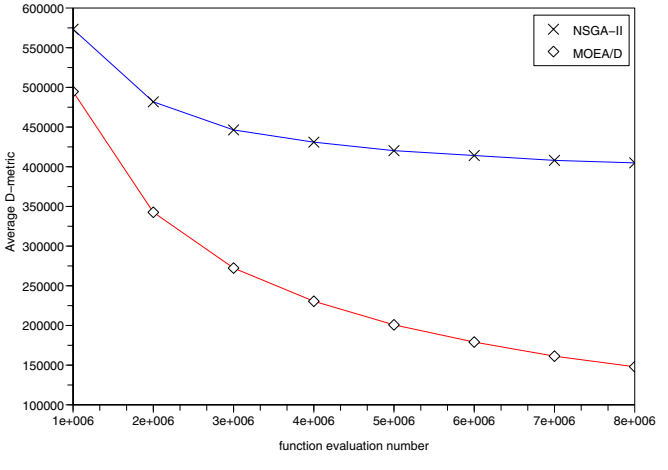


Fig. 6. D-metric vs. the number of function evaluations in the instance with 1500 cities

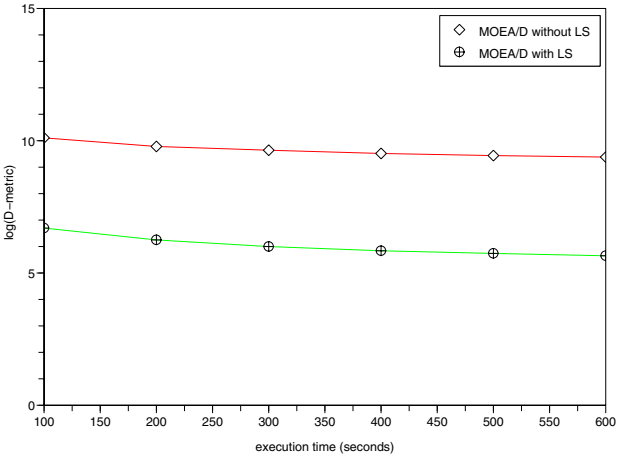


Fig. 7. D-metric vs. execution time in MOEA/D with and without local search for the instance with 500 cities. The scale of the D-metric is log.

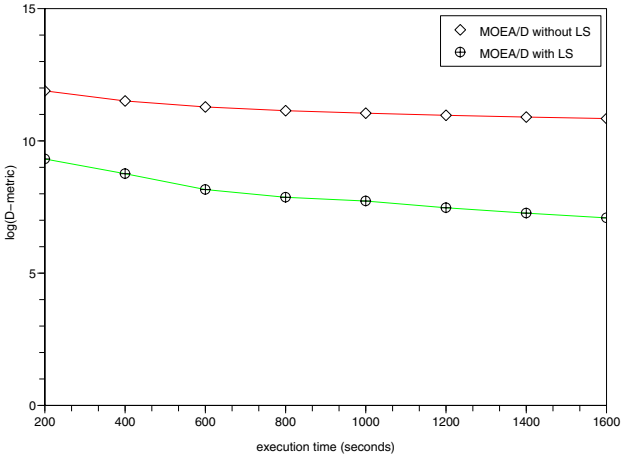


Fig. 8. D-metric vs. execution time in MOEA/D with and without local search for the instance with 1000 cities. The scale of the D-metric is log.

5.1.2 D-Metric

Figures 1-3 present the evolution of the D-metrics of two algorithms with the execution time. It is evident from these figures that MOEA/D always outperforms NSGA-II in terms of D-metric with the same execution time.

Figures 4-6 show how the D-metrics of two algorithms evolve with the number of function evaluations. In the instances with 1000 cities and 1500 cities, it is obvious

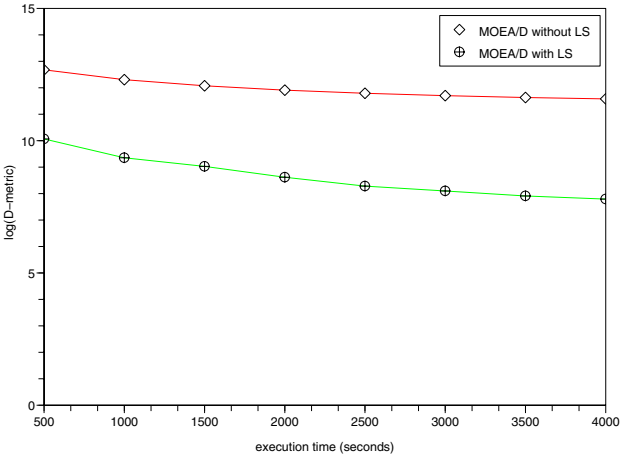


Fig. 9. D-metric vs. execution time in MOEA/D with and without local search for the instance with 1500 cities. The scale of the D-metric is log.

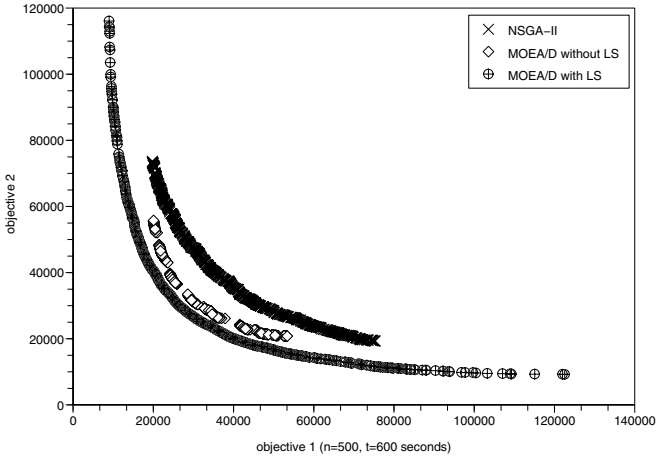


Fig. 10. Distribution of the final solutions generated by the three algorithm generated for the instance with 500 cities

that MOEA/D outperforms NSGA-II after a number of function evaluations in terms of D-metric. In the instance with 500 cities, D-metric in NSGA-II is slightly lower than that in MOEA/D. The reason is that the solutions generated in NSGA-II have better spread than those in MOEA/D in this instance. However, the final solutions found by MOEA/D are much closer to the real Pareto front than those in NSGA-II as shown in terms of C-metric.

These figures also indicate that MOEA/D can have more function evaluations than NSGA-II with the same execution time. This observation confirms the analysis on the computational complexity of these two algorithms in [5].

5.2 The Role of Local Search in MOEA/D

It is well known that hybridizing local search in evolutionary algorithms could improve the algorithmic performance very effectively. However, it is not an easy task to combine

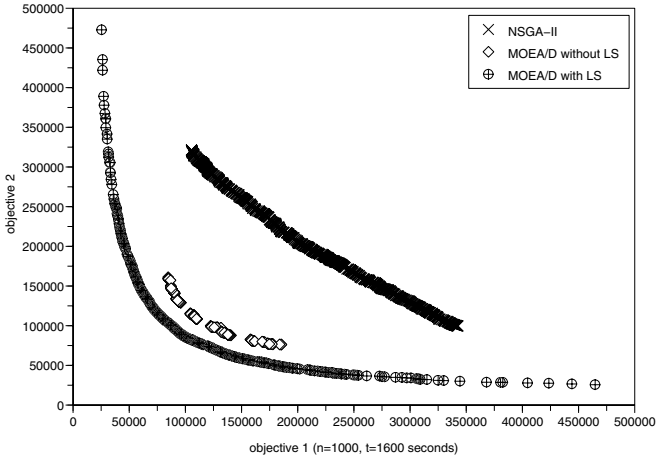


Fig. 11. Distribution of the final solutions generated by the three algorithm generated for the instance with 1000 cities

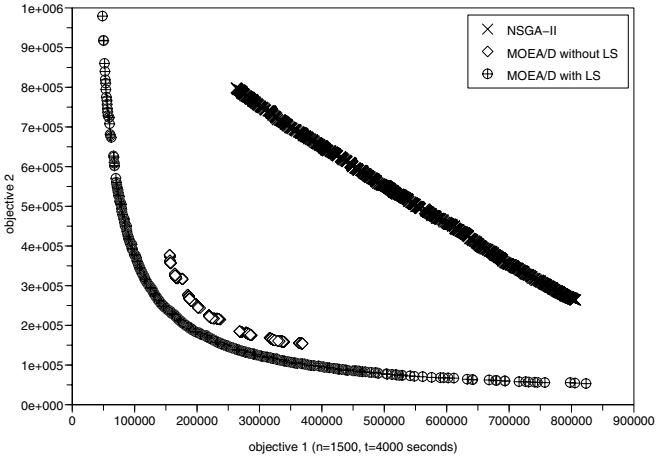


Fig. 12. Distribution of the final solutions generated by the three algorithm generated for the instance with 1500 cities

single-objective optimization local search with Pareto domination based MOEAs such as NSGA-II. In contrast, MOEA/D can readily take the advantage of well-developed single-objective optimization methods. To study the effect of local search (Step 2.2 in MOEA/D) on the performance of algorithm, we have implemented MOEA/D with local search and compared it with MOEA/D without local search. Figures 7-9 compare the evolution of the D-metrics of MOEA/D with and without local search. It is clear that MOEA/D with local search significantly outperforms MOEA/D without local search, which implies that local search can greatly enhance the performance of algorithm.

5.3 The Pareto Fronts Found by Three Algorithms

To visualize the performance of the three algorithms, we plot the distribution of the final solutions in the objective space found by a random run in each algorithm on each test instance in figures 10-12. The results shown in these figures are consistent with observations on the C-metric and D-metric performance. The differences among these three algorithms can be easily noticed from these figures.

6 Summary

Most multiobjective evolutionary algorithms such as NSGA-II are based on Pareto dominance. Single optimization local search is not easy to be combined with these algorithms. A very small number of multiobjective population algorithm are based on decomposition. We believe that the comparison between these two different strategies could be very helpful for understanding their strengths and weaknesses and developing effective and efficient MOEAs. In this chapter, we compared two MOEAs, where one is NSGA-II, a very popular MOEA based on Pareto domination, and the other is MOEA/D, a very recent MOEA based on decomposition. We have taken the multiobjective TSP as a test problem. Our experimental results have shown that MOEA/D without local search outperforms NSGA-II on the three test instances with the same execution time. We have also demonstrated that MOEA/D with local search works much better than MOEA/D without local search. In the future, we will study the performance of MOEA/D on other optimization problems.

References

1. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)
2. Coello, C.A., Veldhuizen, D.A., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer, Norwell (2002)
3. Tan, K.C., Khor, E.F., Lee, T.H.: Multiobjective Evolutionary Algorithms and Applications. Springer, New York (2005)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation 6, 182–197 (2002)
5. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Trans. on Evolutionary Computation 11, 712–731 (2007)

6. Johnson, D.S., McGeoch, L.A.: The traveling salesman problem: a case study. In: Aarts, E., Lenstra, J.K. (eds.) Local Search in Combinatorial Optimization, pp. 215–310. John Wiley & Sons, Chichester (1997)
7. 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, vol. 2632, pp. 479–493. Springer, Heidelberg (2003)
8. Yan, Z., Zhang, L., Kang, L., Lin, G.: A new MOEA for multi-objective TSP and its convergence property analysis. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 342–354. Springer, Heidelberg (2003)
9. García-Martínez, C., Cordón, O., Herrera, F.: A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. European Journal of Operational Research 180, 116–148 (2007)
10. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Norwell (1999)

Integrating Cross-Dominance Adaptation in Multi-Objective Memetic Algorithms

Andrea Caponio[1] and Ferrante Neri[2]

[1] Department of Electrotechnics and Electronics,
Technical University of Bari, Italy
caponio@deemail.poliba.it
[2] Department of Mathematical Information Technology,
University of Jyväskylä, Finland
neferran@cc.jyu.fi

This chapter proposes a novel adaptive memetic approach for solving multi-objective optimization problems. The proposed approach introduces the novel concept of cross-dominance and employs this concept within a novel probabilistic scheme which makes use of the Wigner distribution for performing coordination of the local search. Thus, two local searchers are integrated within an evolutionary framework which resorts to an evolutionary algorithm previously proposed in literature for solving multi-objective problems. These two local searchers are a multi-objective version of simulated annealing and a novel multi-objective implementation of the Rosenbrock algorithm.

Numerical results show that the proposed algorithm is rather promising and, for several test problems, outperforms two popular meta-heuristics present in literature. A real-world application in the field of electrical engineering, the design of a control system of an electric motor, is also shown. The application of the proposed algorithm leads to a solution which allows successful control of a direct current motor by simultaneously handling the conflicting objectives of the dynamic response.

1 Introduction

Many optimization problems in engineering and applied science, due to their nature, require the satisfaction of necessities of various kinds i.e. the desired candidate solution should perform well according to various objectives. In the vast majority of these cases, the objectives are in mutual conflict and a compromise must be accepted. More specifically, as these objectives are usually conflicting, it is not possible to find a single solution that is optimal with respect to all objectives. Which solution is the best depends on the users' utility function, i.e., how the different criteria are weighted. Unfortunately, it is usually rather difficult to formally specify user preferences before the alternatives are known. One way to solve this predicament is by searching for the whole Pareto-optimal front of solutions i.e., all solutions that can not be improved in any criterion without at least sacrificing another criterion.

For example, in control engineering, when a control system is designed, it is desirable that the speed response is very reactive to the input and, at the same time, contains no overshoot and oscillations. Under-dumped responses are usually very reactive but

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 325–351.
springerlink.com

contain overshoot and oscillations in the settling process; on the contrary, over-dumped responses do not contain overshoot or oscillations but usually perform rather poorly in terms of reactivity. It is thus necessary to partially give up both the objectives and find a compromise, that is, a solution which is fairly reactive without excessive overshoot and oscillations.

The well-known scalarized approach [1] i.e. to associate a weight factor (on the basis of their importance) to each objective and then optimize the weighted sum, though in some cases rather efficient, implicitly accepts that a ranking of the importance of the objectives and the related proportion of how much each objective should be taken into account with respect to the others, is known beforehand. Moreover, the weighted sum approach has the main disadvantage that it implicitly excludes some solutions from the search since the a priori determination of the weighted value may assign a low fitness value to some solutions which could on the contrary be interesting in the application viewpoint. Therefore, in many cases it is preferable to employ a multi-objective approach [1]. Since the latter considers the objectives simultaneously and leads to a set of solutions, the user can choose by means of a decision making process the most suitable solution amongst those that are actually available.

Due to their structure, Evolutionary Algorithms (EA) have been proven to be very promising in multi-objective optimization and have been intensively used during the last two decades (see the implementation proposed in [2]). As shown in [3] and [4], Multi-objective Optimization Evolutionary Algorithms (MOEA) are very efficient in finding the Pareto-optimal or near Pareto-optimal solutions. Several algorithms have been designed for such a purpose, for example the Non-dominated Sorting Genetic Algorithm II (NSGA II) [5] and the Strength Pareto Evolutionary Algorithm-2 (SPEA-2) [6].

Memetic Algorithms (MAs) are population based meta-heuristics which combine local search components within an evolutionary framework [7]. For single-objective optimization problems MAs, if well-designed for specific applications by taking into account features of the fitness landscape, have been proven to outperform classical meta-heuristics e.g. Genetic Algorithms (GAs), Evolution Strategy (ES), Particle Swarm Optimization (PSO) etc. [8] [9]. One crucial problem in the algorithmic design of MAs is coordination among the evolutionary framework and local search and amongst the various local searchers [10]. The problem of local search coordination has been widely discussed over the years. In [7] the concept of coordination and cooperation of local searchers has been introduced, after being developed in [11]. In [10] the use of multiple local search operators having different features in order to explore the decision space under different perspectives has been proposed. In recent years, several kinds of adaptation and self-adaptation for coordinating the local search have been designed. In [12] a classification of adaptive MAs is given while a tutorial which organizes the basic concepts of MAs including the coordination of the local search is given in [13].

MAs have been recently applied to multi-objective optimization, as discussed in [14] and several Multi-Objective Memetic Algorithms (MOMA) have therefore been designed. In such design two crucial problems arise: the first is the proper definition of local search in a multi-objective environment, the second is the balance between global and local search in presence of many simultaneous objectives [15], [16]. This balance,

which is strictly related to the local search coordination, is extremely difficult to be performed and, as highlighted in the empirical study reported in [17], an adaptation is so difficult to be defined that it might be preferable in several cases to employ simple time-dependant heuristic rules.

This chapter proposes an adaptation scheme based on the mutual dominance between non-dominated solutions belonging to subsequent generations uncoupled with a probabilistic criterion in order to coordinate and balance the global and local search within a MOMA. The proposed algorithm is called Cross Dominant Multi-Objective Memetic Algorithm (CDMOMA). Section 2 gives a detailed description of the algorithmic components and their interaction, Section 3 shows the behavior of the proposed algorithm in an extensive amount of test cases, Section 4 analyzes a real-world engineering problem, Section gives the conclusion of our work.

2 Cross Dominant Multi-Objective Memetic Algorithm

Let us consider a classical multi-objective optimization problem:

$$\left.\begin{array}{cll} \text{Minimize/Maximize} & f_m(x), & m = 1,2,\ldots,M \\ \text{subject to} & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1,2,\ldots,n \end{array}\right\} \quad (1)$$

where f_m is the m^{th} single objective function, a solution x is a vector of n decision variables. Each decision variable is limited to take a value within a lower $x_i^{(L)}$ and an upper $x_i^{(U)}$ bound. These bounds define the decision space D.

In order to solve the problem in eq. (1), the CDMOMA has been designed. The CDMOMA is composed of an evolutionary framework resorting the NSGA-II and two local searchers, a multi-objective implementation of the Rosenbrock algorithm and of the Simulated Annealing respectively, adaptively coordinated by criterion based on mutual dominance amongst the individuals of two populations at two consecutive generations and a probabilistic scheme.

For the sake of completeness and better understanding of the CDMOMA the classical definitions of dominance [3] are given. Without a generality loss, the following definitions refer to the minimization of all the objective functions.

Definition 1. A solution $x^{(1)}$ is said to dominate the other solution $x^{(2)}$ ($x^{(1)} \preceq x^{(2)}$), if both conditions 1 and 2 are true:

1. The solution $x^{(1)}$ is no worse than $x^{(2)}$ in all objectives,
2. The solution $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective.

Definition 2. A solution $x^{(1)}$ is said to strictly dominate the other solution $x^{(2)}$ ($x^{(1)} \prec x^{(2)}$), if solution $x^{(1)}$ is strictly better than $x^{(2)}$ in all the M objectives.

2.1 The Evolutionary Framework

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) introduced in [5], is the second and improved Version of the Non-dominated Sorting Genetic Algorithm proposed in [18], it is an elitist multi-objective evolutionary algorithm which proves to

have high performance in terms of both quality and distribution of the detected non-dominated solutions.

Briefly, an initial sampling is performed pseudo-randomly within the decision space, thus generating S_{pop} individuals. At each generation, $S_{pop}/2$ parents are selected according to a binary tournament selection. For each pairwise comparison the winner is the dominating individual (for definition of dominance see [1] or [3]). If the individuals are non-dominant (to each other), the individual having a higher value in the crowding distance is selected. The crowding distance of an individual is a measure of the distance between the individual under examination and the other individuals belonging to the same set of non-dominated solutions (see [5] for details).

Then, for $S_{pop}/2$ times a pseudo-random value is generated. Each time, if this value is lower than 0.1, one individual is pseudo-randomly selected and then mutated; if, on the contrary, it is higher than 0.1, two parents are pseudo-randomly paired and undergo crossover.

Polynomial mutation [19] and simulated binary crossover [20], [21] are employed. Since mutation generates one child and the simulated binary crossover two children, an offspring population composed of a number of individuals between $S_{pop}/2$ and S_{pop} is thus generated.

This offspring population is merged to the population produced from the previous generation. Then, according to an elitist logic, S_{pop} individuals are selected for survival to the subsequent generation. The survivor selection scheme sorts individuals according to their rank i.e. divides the individuals into subsets according to their level of dominance. Thus, the subset of rank 1 is the set of non-dominated solutions, the subset of rank 2 is the set of non-dominated solutions if we remove those individuals belonging to the first subset, the subset of rank 3 is the set of non-dominated solutions after having removed the individuals of the first and second subset and so on. Within each subset, the individuals are then sorted on the basis of their crowding distance. More formally, for a given pair of individuals i and j, and indicating with i_r and i_{cd} the rank and crowding distance of i respectively, the partial order (here indicated with $\prec_n$) is defined as:

$$i \prec_n j \; IF \; (i_r < j_r) \; OR \; ((i_r = j_r) AND (i_{cd} > j_{cd}))$$

The sorting performed amongst a set of solutions by employing the formula above is called non-dominated sorting. The selected individuals compose the new population for the subsequent generation.

2.2 Local Searchers

The CDMOMA employs two local searchers within the generation loop of the NSGA-II evolutionary framework. These algorithms are a novel multi-objective implementation of the Rosenbrock algorithm and the Pareto Domination Multi-Objective Simulated Annealing (PDMOSA) proposed in [22]. In the following subsections a description of these two algorithms is given.

2.2.1 The Multi-Objective Rosenbrock Algorithm

The classical Rosenbrock Algorithm [23] is a single objective algorithm that works on a solution and attempts to improve upon it by means of a steepest descent pivot rule.

A novel implementation of the Multi-Objective Rosenbrock Algorithm (MORA) is proposed here. The MORA consists of the following. Starting from x, a trial is made in all the n orthogonal directions of the n-dimensional decision space. A trial over the i^{th} decision variable is performed by checking the value of $y = [x_1, x_2 ... x_i + stepLength, x_{i+1}, .., x_n]$ where the $stepLength$ is the step length i.e. the length of the exploratory step. When a new point y is generated, it is compared with the old one x. If the new point is not dominated by the old one we have a success. In such a case, the new point is retained ($x = y$) and the step length is multiplied by a positive factor α. If the new point is dominated by the old one we have a failure. In this case, the vector of variables is left unchanged and the step length is multiplied by a negative factor

```
i = 1;
initialize stepLength;
initialize SuccessAndFailure;
while budget condition
   generate next point y from point x:
      y_j = x_j for j = 1,...,n and j ≠ i;
      y_i = x_i · stepLength_i;
   if y is out of bounds
      f_k(y) = ∞ ∀ k = 1,...,M;
   else
      evaluate y;
   end-if
   if x ≼ y
      stepLength_i = stepLength_i · β;
      if SuccessAndFailure_i == success
         SuccessAndFailure_i = successFailure;
      else
         SuccessAndFailure_i = failure;
      end-if
   else
      x = y;
      stepLength_i = stepLength_i · α;
      SuccessAndFailure_i = success;
   end-if
   if SuccessAndFailure_j == successFailure ∀ j = 1,...,n;
      rotate base by Gram and Schmidt procedure;
      initialize stepLength;
      initialize SuccessAndFailure;
   else-if i < n
      i = i + 1;
   else
      i = 1;
   end-if
end-while
```

Fig. 1. MORA pseudo-code

$-1 < \beta < 0$. According to Rosenbrock's suggestions $\alpha = 3$ and $\beta = -0.5$ have been set [23]. As in the single objective Rosenbrock algorithm, the process is repeated until at the least a success is followed by a failure in each direction. When such a condition is satisfied, the orthogonolization procedure of Gram and Schmidt (see [24]) is executed and the search, along the new set of directions, begins again. The algorithm is stopped when a budget condition is exceeded.

According to the given definitions of "success" and "failure", the MORA accepts a new point only when it does not decrease performance in each of the objective functions; if even one worsens, the point is discarded. Thus, the MORA handles the various objective functions without performing a scalarization.

It must be highlighted that when, during a MORA step, a solution outside the decision space is generated, the algorithm assigns an infinite value to every one of its objectives.

Fig. 1 shows the pseudo-code of the proposed MORA. It should be noted that the dominance condition is represented by the symbol $\prec$ e.g. x dominates y is expressed by $x \prec y$; analogously, x does not dominate y is expressed by $x \succeq y$ (see [3]). With reference to Fig. 1, the variable *SuccessAndFailure* is a vector of three valued flag variables which records for each of its elements $SuccessAndFailure_i$ the behavior of the algorithm during the previous two steps. More specifically, it records the value $failure$ if the trial failed twice, it records the value *success* if the trial either succeeded twice or succeeded after having failed, it records the value *successFailure* if the trial failed after having succeeded. The latter condition determines the activation of the Gram and Smith procedure.

2.2.2 Pareto Domination Multi-Objective Simulated Annealing

The multi-objective simulated annealing algorithm implemented here is based on the Pareto Domination Multi-Objective Simulated Annealing PDMOSA (PDMOSA) proposed in [22]. The PDMOSA works on a solution x and an auxiliary population in order to improve upon the starting point. At each step, the current best solution is perturbed by means of a Gaussian distribution and a perturbed solution y is thus generated. For

while budget condition
  initialize y;
  while y is out of the bounds of the decision space
    generate y by perturbing $x$by means of a Gaussian distribution;
  end-while
  calculate all the single objective values of y;
  $dx =$ number of the population individuals dominated by x;
  $dy =$ number of the population individuals dominated by y;
  replace x with y with a probability $p = e^{\frac{dx-dy}{T}}$;
  decrease temperature T by means of an hyperbolic law;
end-while

Fig. 2. MOSA pseudo-code

both, current best and perturbed solution, the number of individuals of the population which are dominated by x and y respectively are calculated. In the fashion of simulated annealing the new solution y replaces x with a time-dependant probability. The temperature is decreased by means of a hyperbolic law as suggested in [25]. For the sake of clarity, the PDMOSA pseudo-code which highlights the working principles is shown in Fig. 2.

2.3 Adaptation

Definition 3. Let us consider two sets of candidate solutions, namely X and Y respectively. Without a generality loss, let's assume that the cardinality of both sets is N. By scrolling all the elements of set Y, let's enumerate the dominance occurrences with each element of set X. N^2 comparisons are thus performed. Let us assign Λ to be this number of dominance occurrences. The set Y is said to **cross-dominate** the set X **with a grade**:

$$\lambda = \frac{\Lambda}{N^2} \tag{2}$$

Fig. 3 gives a graphical representation of the concept of cross-dominance. The solid lined arrow represents the dominance of the point under examination while the dash lined arrow represents non-dominance.

This chapter proposes to use the concept of cross-dominance in order to perform an adaptive coordination of the local search. More specifically, at the end of each generation the parameter λ is calculated:

$$\lambda = \frac{\Lambda^{t+1}}{N^2} \tag{3}$$

where Λ^{t+1} is the number of dominance occurrences obtained by the comparison of the population at generation $t+1$ (which plays the role of the set Y in the definition above) with respect to the population at generation t (which plays the role of X).

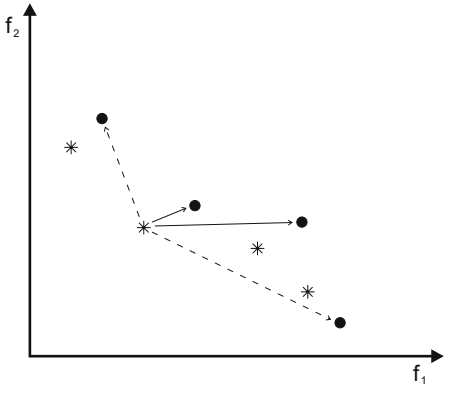


Fig. 3. Graphical Representation of the Cross-dominance

In this way, the algorithm can monitor the overall improvements of the population by means of a parameter which acquires values between 0 and 1. More specifically, if $\lambda = 1$ the algorithm is making excellent improvements and all individuals of the population at generation $t+1$ strictly dominate all individuals at generation t. On the contrary, if $\lambda = 0$ the algorithm is not leading to any improvement and the new population is equivalent to the old one in terms of dominance. It must be remarked that this adaptation index should be integrated within a fully elitist system (as the NSGA-II), thus a temporary worsening is not allowed. In addition, it should be observed that even though λ can acquire values between 0 and 1, most likely it will acquire values around zero ($\lambda = 0.05$ means that the population is still significantly better than the previous one).

The main idea is to design an adaptive system which automatically coordinates evolutionary framework and local search components by estimating algorithmic improvements, thus the necessity of the search during the optimization process.

2.4 Coordination of the Local Search

In order to perform coordination of the local search, λ is employed in a novel way. More specifically, for each local searcher, a generalized Wigner semicircle distribution is generated:

$$p(\lambda) = \frac{2}{\pi R^2}\sqrt{R^2 - (\lambda - a)^2}\,\frac{c}{\left(\frac{2}{\pi R}\right)} \tag{4}$$

where R is the radius of the distribution (the shape of the distribution depends on R), a determines the shift of the distribution, c is the maximum value of the distribution. For the MORA, we consider a distribution that has its maximum value equal to 0.8 for $\lambda = 0$, and that is 0 for $\lambda > 0.007$ (we consider just a half of the semi-elliptic Wigner distribution). For the PDMOSA, we consider a function that has its maximum value equal to 0.1 for $\lambda = 0.0125$, while it is 0 for $\lambda < 0.005$ and $\lambda > 0.02$. Thus, these two distributions return the probability of the local search activation dependent upon the adaptive parameter λ. Furthermore, the MORA is applied to 25 individuals while the PDMOSA is applied to 5 individuals pseudo-randomly selected respectively. Fig. 4 graphically shows the probability distribution for the CDMOMA adaptive local search coordination.

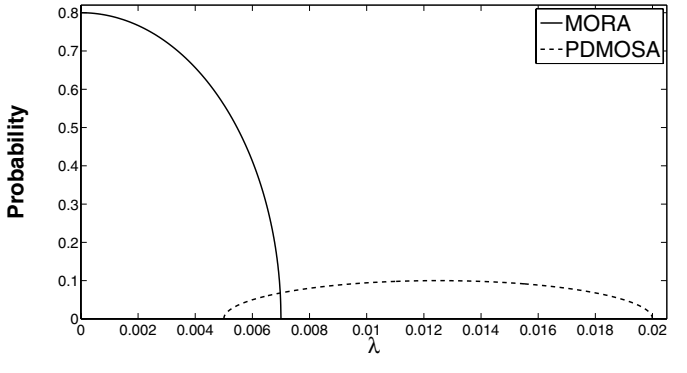


Fig. 4. Graphical Representation of the Probabilistic Scheme for the Local Search Coordination

```
generate initial population pseudo-randomly;
compute the fitness values of the individuals of the initial population;
perform the non-dominated sorting;
while budget condition
   execute NSGA-II generation;
   apply the cross-dominance procedure and compute λ = Λ^{t+1}/N^2;
   compute p(λ);
   generate pseudo-randomly ε ∈ [0,1]
   if ε < p(λ)
      execute PDMOSA on 5 individuals pseudo-randomly selected,
      for 3000 fitness evaluations;
      replace the 5 individuals with the results of the PDMOSA;
   end-if
   if ε < p(λ)
      execute RA on 25 individuals pseudo-randomly selected,
      for 1000 fitness evaluations;
   end-if
end-while
```

Fig. 5. CDMOMA pseudo-code

The two employed local searchers have clearly different structures in terms of pivot rule and neighborhood generating function. It should be noted that the MORA is a steepest descent local searcher which explores the neighborhood of a promising solution. On the contrary, the PDMOSA employs a simulated annealing logic which attempts to achieve a global property during the exploration of the decision space. According to our algorithmic philosophy, a decrease of the parameter λ during the optimization process corresponds to a settlement of the population over a set of non-dominated solutions. These solutions will most likely be better spread out by the evolutionary framework without any improvement, in terms of quality, of the detected solutions. In such conditions, the PDMOSA has the role of providing a new perspective into the search and hopefully detects new non-dominated solutions in still unexplored areas of the decision space. When, notwithstanding this action (by the PDMOSA) the populations' new generation seems to have negligible improvements, the MORA attempts to further improve the solutions by exploring their neighborhood. In other words, the CDMOMA attempts, at first, to generate a set of non-dominated solutions by the NSGA-II, then combines the actions of the evolutionary components and local searchers for improving the performance of the non-dominated set and eventually employs the local search to further improve the solutions and the NSGA-II to assure a good spread to the set.

Fig 5 shows the pseudo-code of the proposed CDMOMA.

3 Numerical Results

The CDMOMA has been tested on eight popular test problems: FON (from Fonseca and Fleming's study [26]), POL(from Poloni's study [27]), KUR (from Kursawe's study

[28]) and five ZDT problems (from Zitzler, Deb and Tiel) selected from [29] formulated according to the study in [30].

Table 1 lists all the test problems under examination and the related details.

Table 1. Test Problems

Prob.	n	Bounds	Objective functions	Solutions	Comments
ZDT1	30	$[0,1]$	$f_1(x) = x_1$ $f_2(x) = g(x)\left[1-\sqrt{\frac{x_1}{g(x)}}\right]$ $g(x) = 1+9\frac{\sum_{i=2}^{n} x_i}{n-1}$	$x_1 \in [0,1]$, $x_i = 0$, $i = 2,..,n$	convex
ZDT2	30	$[0,1]$	$f_1(x) = x_1$ $f_2(x) = g(x)\left[1-\left(\frac{x_1}{g(x)}\right)^2\right]$ $g(x) = 1+9\frac{\sum_{i=2}^{n} x_i}{n-1}$	$x_1 \in [0,1]$, $x_i = 0$, $i = 2,..,n$	nonconvex
ZDT3	30	$[0,1]$	$f_1(x) = x_1$ $f_2(x) = g(x)\left[1-\sqrt{\frac{x_1}{g(x)}}-\frac{x_1}{g(x)}\sin(10\pi x_1)\right]$ $g(x) = 1+9\frac{\sum_{i=2}^{n} x_i}{n-1}$	$x_1 \in [0,1]$, $x_i = 0$, $i = 2,..,n$	convex, disconnected
ZDT4	10	$x_1 \in [0,1]$ $x_i \in [-5,5]$ $i = 2,..,n$	$f_1(x) = x_1$ $f_2(x) = g(x)\left[1-\sqrt{\frac{x_1}{g(x)}}\right]$ $g(x) = 1+10(n-1)+\sum_{i=2}^{n}\left[x_i^2-10\cos(4\pi x_i)\right]$	$x_1 \in [0,1]$, $x_i = 0$, $i = 2,..,n$	nonconvex
ZDT6	10	$[0,1]$	$f_1(x) = 1-\exp(-4x_1)\sin^6(4\pi x_1)$ $f_2(x) = g(x)\left[1-\left(\frac{f_1(x)}{g(x)}\right)^2\right]$ $g(x) = 1+9\left[\frac{\sum_{i=2}^{n} x_i}{n-1}\right]^{0.25}$	$x_1 \in [0,1]$, $x_i = 0$, $i = 2,..,n$	nonconvex, nonuniformly spread
FON	3	$[-4,4]$	$f_1(x) = 1-\exp\left(-\sum_{i=1}^{3}\left(x_i-\frac{1}{\sqrt{3}}\right)^2\right)$ $f_1(x) = 1-\exp\left(-\sum_{i=1}^{3}\left(x_i+\frac{1}{\sqrt{3}}\right)^2\right)$	$x_1 = x_2 =$ $= x_3 \in$ $\left[\frac{-1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right]$	nonconvex
POL	2	$[-\pi,\pi]$	$f_1(x) = \left[1+(A_1-B_1)^2+(A_2-B_2)^2\right]$ $f_2(x) = \left[(x_1+3)^2+(x_2+1)^2\right]$ $A_1 = 0.5\sin 1-2\cos 1+\sin 2-1.5\cos 2$ $A_2 = 1.5\sin 1-\cos 1+2\sin 2-0.5\cos 2$ $B_1 = 0.5\sin x_1-2\cos x_1+\sin x_2-1.5\cos x_2$ $B_2 = 1.5\sin x_1-\cos x_1+2\sin x_2-0.5\cos x_2$		nonconvex, disconnected
KUR	3	$[-5,5]$	$f_1(x) = \sum_{i=1}^{n-1}\left(-10\exp\left(-0.2\sqrt{x_i^2+x_{i+1}^2}\right)\right)$ $f_2(x) = \sum_{i=1}^{n}\left(\lvert x_i\rvert^{0.8}+5\sin x_i^3\right)$		nonconvex

The CDMOMA performance has been compared with the SPEA-2 [6] and the NSGA-II [5]. The three algorithms have been executed with a population size of 150 individuals, with a total budget of 800000 fitness evaluations. For each test problem 50 initial populations have been pseudo-randomly sampled within the respective decision space. For each of these 50 populations the three algorithms have been independently run. Thus, for each test problem each algorithm has been run 50 times. Fig.'s 6, 7, 8, 9, 10, 11, 12, 13 show the results obtained on selected runs.

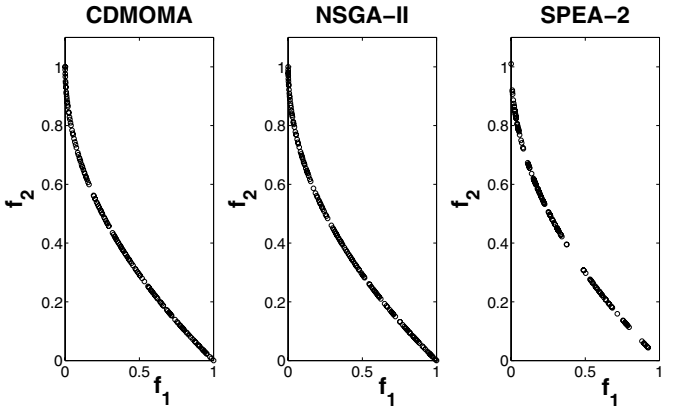


Fig. 6. ZDT1, selected solutions

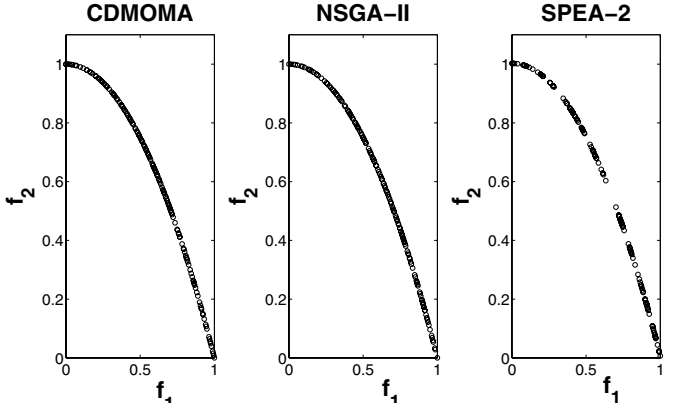


Fig. 7. ZDT2, selected solutions

Numerical results on selected runs qualitatively show that the CDMOMA is able to detect very good sets of non-dominated solutions in terms of fitness values and spreading.

In order to also give a graphical representation of the average algorithmic performance, for the single run of each algorithm, the final population has been sorted on the basis of the first objective function. For each algorithm, the sorted objective function values are averaged over each objective. Fig.'s 14, 15, 16, 17, 18, 19, 20, 21 show the average algorithmic performance.

Numerical results indicate that the CDMOMA seems to have a promising behavior with most of the problems under examination. In particular, in the case of the POL, the CDMOMA has a performance comparable to that of the NSGA-II and better than the SPEA-2; in the case of the FON, KUR, ZDT1, and ZDT3, the CDMOMA seems to be significantly more efficient than the SPEA-2 in detecting a good set of solutions

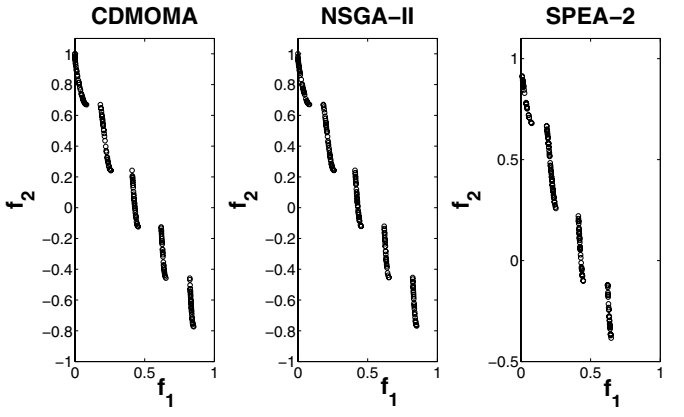


Fig. 8. ZDT3, selected solutions

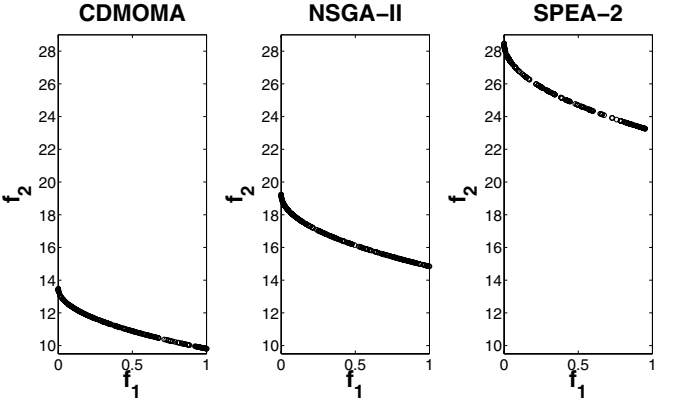


Fig. 9. ZDT4, selected solutions

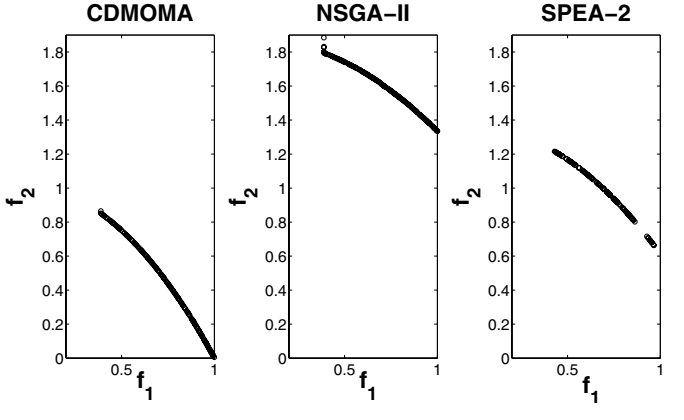


Fig. 10. ZDT6, selected solutions

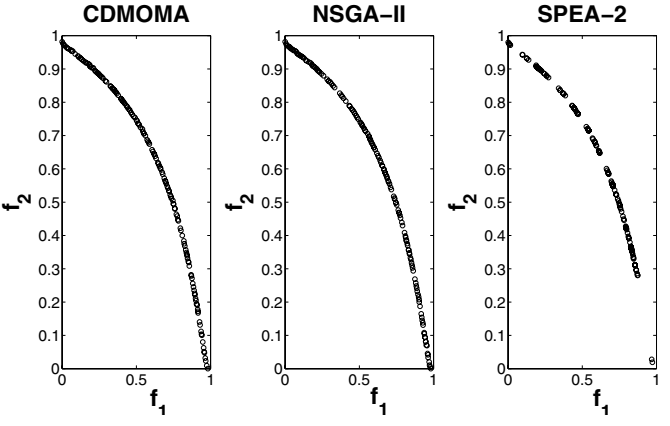


Fig. 11. FON, selected solutions

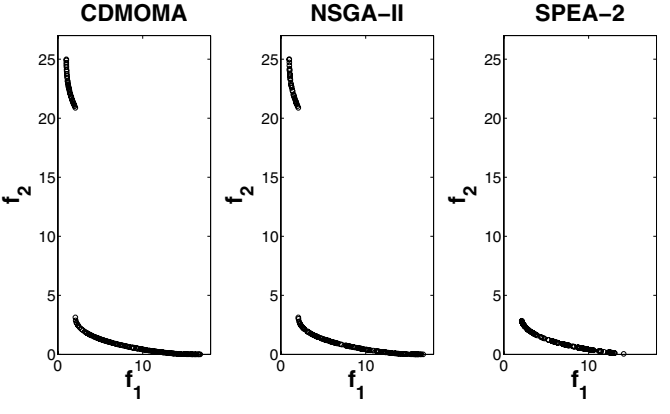


Fig. 12. POL, selected solutions

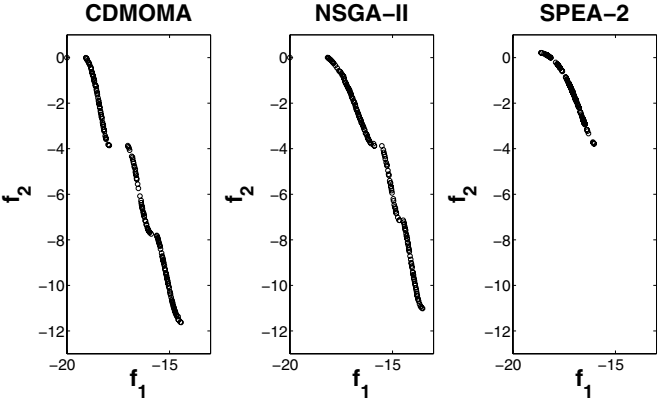


Fig. 13. KUR, selected solutions

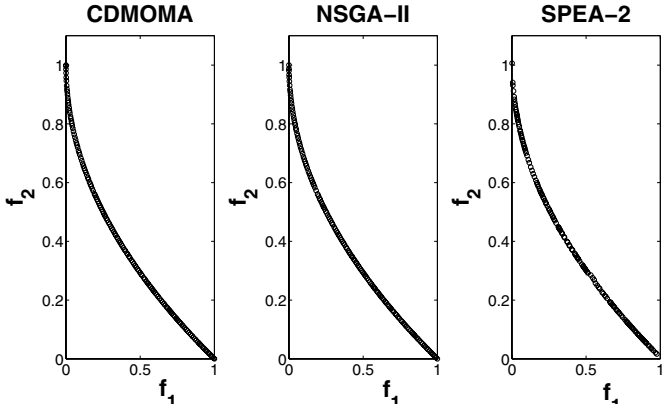


Fig. 14. ZDT1, average performance

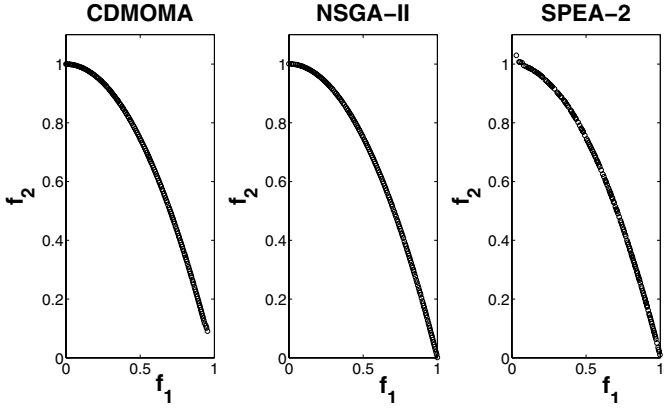


Fig. 15. ZDT2, average performance

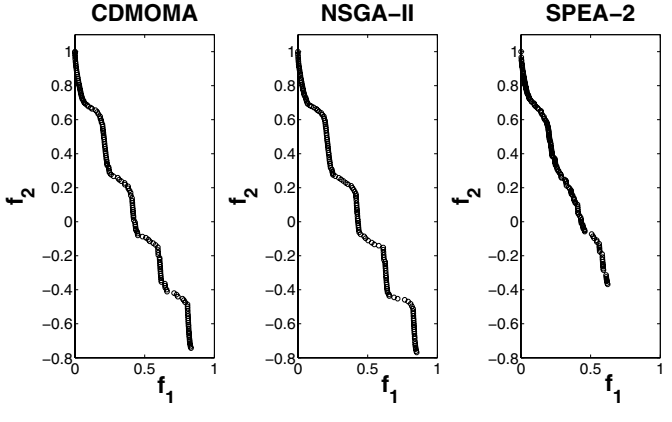


Fig. 16. ZDT3, average performance

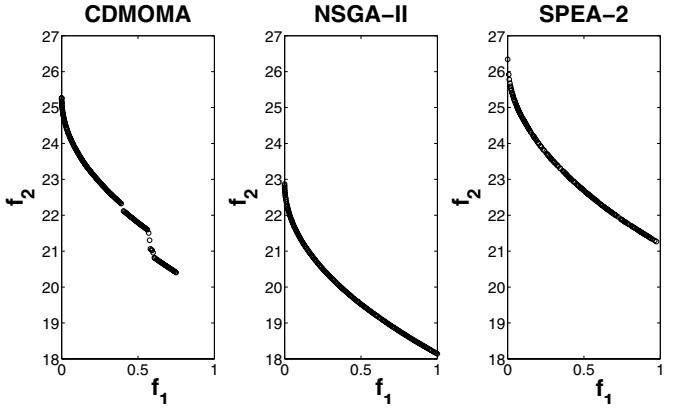


Fig. 17. ZDT4, average performance

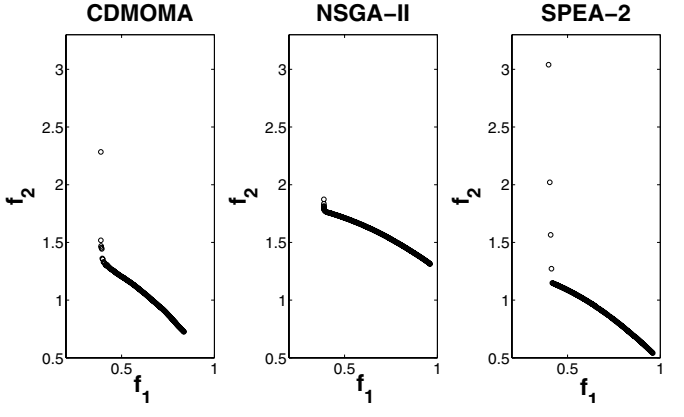


Fig. 18. ZDT6, average performance

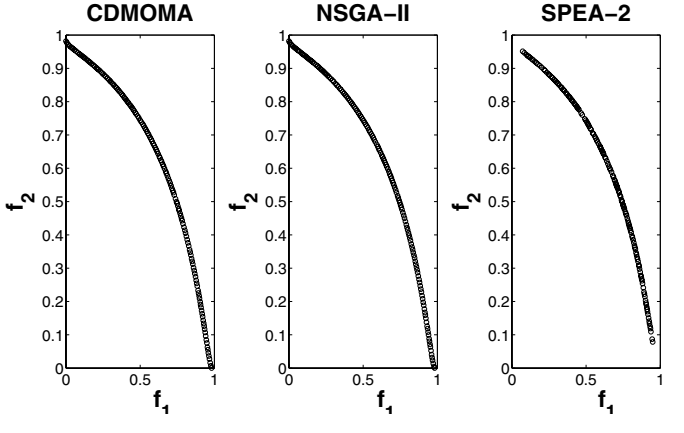


Fig. 19. FON, average performance

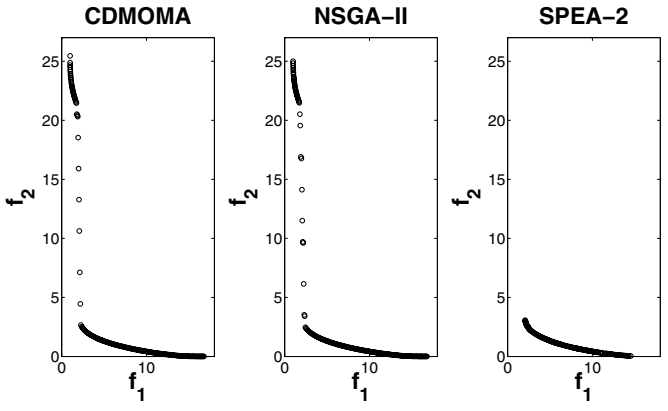


Fig. 20. POL, average performance

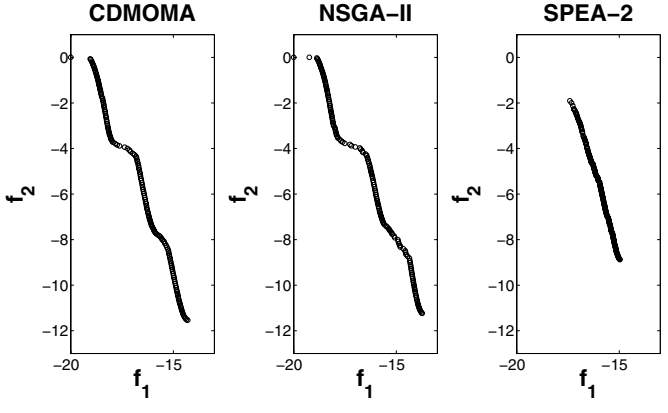


Fig. 21. KUR, average performance

and slightly more efficient than the NSGA-II; in the case of of ZDT2, the CDMOMA behaves slightly worse than the NSGA-II and comparably to the SPEA-2; in the case of the ZDT6, the CDMOMA seems to behave better than the NSGA-II but worse than the SPEA-2; in the case of the ZDT4, the CDMOMA is definitely worse than the NSGA-II and globally comparable to the SPEA-2. Thus, it can be stated that, except in the case of the ZDT4 test problem, the CDMOMA detects on average a set of solutions with high performance and good spreading features.

In order to have a more quantitative comparison by means of the performance measures, Υ and Δ (see [5] and [31]) has been carried out. The first metric Υ measures the extent of convergence to a known set of Pareto-optimal solutions. First, a set of 500 uniformly spaced solutions from the true Pareto-optimal front is detected. For each solution obtained with an algorithm, the minimum Euclidean distance it has from the 500 chosen solutions on the true Pareto-optimal front is computed. The average of these

Table 2. Υ values

	NSGA-II		CDMOMA		SPEA-2	
	$\bar{\Upsilon}$	σ^2_{Υ}	$\bar{\Upsilon}$	σ^2_{Υ}	$\bar{\Upsilon}$	σ^2_{Υ}
ZDT1	0.0012	$8.1314 \cdot 10^{-9}$	0.0011	$9.7175 \cdot 10^{-9}$	0.0111	$8.6124 \cdot 10^{-6}$
ZDT2	0.0014	$3.9939 \cdot 10^{-6}$	0.0008	$1.0723 \cdot 10^{-7}$	0.0136	$1.3993 \cdot 10^{-5}$
ZDT3	0.0014	$4.7059 \cdot 10^{-9}$	0.0013	$2.8252 \cdot 10^{-7}$	0.0128	$2.3767 \cdot 10^{-5}$
ZDT4	19.1313	$4.1036 \cdot 10^{1}$	21.7563	$1.0508 \cdot 10^{2}$	23.3591	$5.5884 \cdot 10^{1}$
ZDT6	0.8279	$1.2301 \cdot 10^{-1}$	0.4678	$5.7394 \cdot 10^{-1}$	0.4748	$5.1439 \cdot 10^{-2}$
FON	0.0061	$4.5006 \cdot 10^{-8}$	0.0061	$3.4269 \cdot 10^{-8}$	0.0071	$1.7740 \cdot 10^{-7}$

distances is used as the first metric Υ. In other words, Υ known also as the convergence metric, is a measurement of deviation of the detected set of solutions from the true Pareto-optimal front. Thus, it can be concluded that if $\Upsilon \approx 0$ algorithm is efficient. It should be remarked that this metric can be employed only when the true set of Pareto-optimal solutions is known. Thus, it is obvious that this metric cannot be used for any arbitrary problem.

The second metric Δ measures the extent of spread achieved among the obtained solutions, since one of the goals in multi-objective optimization is to acquire a set of solutions that spans the entire Pareto-optimal region. In order to compute Δ, the Euclidean distance d_i (in the multi-dimensional codomain) between consecutive solutions (with respect to the sorting according to one arbitrary objective function) in the obtained non-dominated set of solutions is calculated. The average of these distances $\bar{d}$ is then calculated. Then, if the true Pareto-optimal front is known, the Euclidean distances d_f and d_l between the extreme solutions and the boundary solutions of the obtained non-dominated set are calculated. The non-uniformity metric Δ is given by:

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \overline{d} \right|}{d_f + d_l + (N-1)\,\overline{d}}, \tag{5}$$

where N is the cardinality of the non-dominated set. If the true Pareto-optimal front is not known, d_f and d_l are ignored by imposing $d_f = d_l = 0$. Formula (5) is thus modified:

$$\Delta = \frac{\sum_{i=1}^{N-1} \left| d_i - \overline{d} \right|}{(N-1)\,\overline{d}}. \tag{6}$$

For further details see [5] and [31]. Since a high spreading, in the non-dominated set of solutions, is desired, $\Delta \approx 0$ characterizes a good set of solutions.

For each algorithm and each test problem, the average values $\bar{\Upsilon}$ and $\bar{\Delta}$ have been calculated over the 50 runs available. A graphical representation of the $\bar{\Upsilon}$ and $\bar{\Delta}$ values are given in Fig. 22 and 23 respectively. Since the calculation of Υ requires the a priori knowledge of the actual Pareto front which is unknown for POL and KUR test problems, the Υ values related to these two problems are missing in the following analysis.

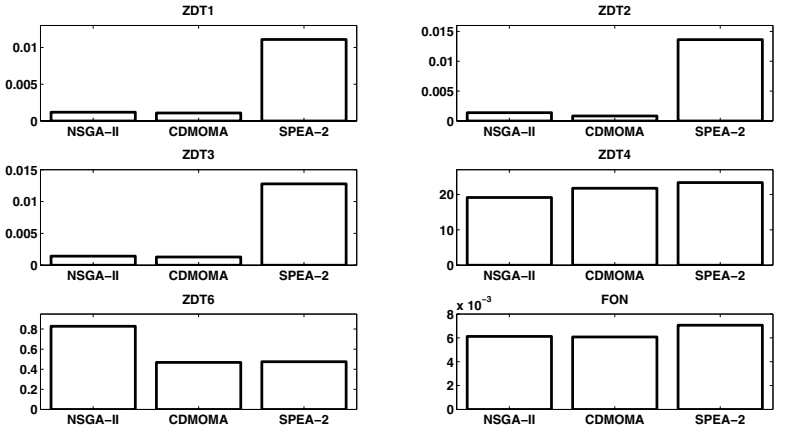


Fig. 22. $\bar{\Upsilon}$ values

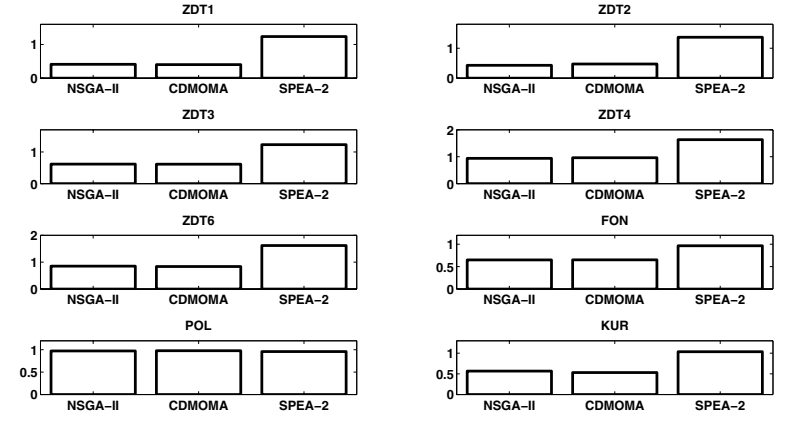


Fig. 23. $\bar{\Delta}$ values

Table 2 lists average and variance values of Υ and Table 3 lists average and variance values of Δ.

The quantitative analysis of the results shows that, regarding the convergence property of the algorithms, the CDMOMA seems to have a very promising capability of detecting a set of solutions which is similar to the true Pareto-optimal front. In particular, Fig. 22 and Table 2 show that the CDMOMA obtained the best convergence metric Υ for all the available test problems except the ZDT4. In the latter case, the CDMOMA still performs better than the SPEA-2. Regarding spreading of the solutions within the set, the CDMOMA is also rather promising. Results in Fig. 23 and Table 3 show that the CDMOMA has a better performance than the SPEA-2 (except for the POL) and comparable to the NSGA-II. This finding was somehow expectable since the CDMOMA

Table 3. Δ values

	NSGA-II		CDMOMA		SPEA-2	
	$\bar{\Delta}$	σ^2_{Δ}	$\bar{\Delta}$	σ^2_{Δ}	$\bar{\Delta}$	σ^2_{Δ}
ZDT1	0.4108	$5.6038 \cdot 10^{-4}$	0.4003	$3.5590 \cdot 10^{-4}$	1.2347	$2.8580 \cdot 10^{-3}$
ZDT2	0.42834	$6.0626 \cdot 10^{-4}$	0.4723	$7.5048 \cdot 10^{-3}$	1.3670	$2.9583 \cdot 10^{-3}$
ZDT3	0.6147	$4.1059 \cdot 10^{-4}$	0.6126	$8.1888 \cdot 10^{-4}$	1.2306	$2.8785 \cdot 10^{-3}$
ZDT4	0.9395	$2.4384 \cdot 10^{-4}$	0.9620	$1.6705 \cdot 10^{-3}$	1.6331	$9.9375 \cdot 10^{-3}$
ZDT6	0.8521	$3.6873 \cdot 10^{-3}$	0.8359	$1.1110 \cdot 10^{-1}$	1.6178	$9.1970 \cdot 10^{-3}$
FON	0.6491	$1.7710 \cdot 10^{-4}$	0.6520	$2.7579 \cdot 10^{-4}$	0.9660	$1.0791 \cdot 10^{-3}$
POL	0.9722	$2.5309 \cdot 10^{-4}$	0.9775	$6.3631 \cdot 10^{-4}$	0.9583	$2.7690 \cdot 10^{-3}$
KUR	0.5659	$3.7139 \cdot 10^{-3}$	0.5313	$2.4347 \cdot 10^{-3}$	1.0343	$2.5964 \cdot 10^{-2}$

employs the the NSGA-II logic in its evolutionary framework and thus both algorithms have the same sorting structure, this being an algorithmic component that heavily affects the spreading in the population.

In conclusion the results from the set of benchmark problems allow the authors to state that the proposed CDMOMA is a rather promising algorithm for multi-objective optimization problems. According to our interpretation, employment of the local search algorithms allows an improvement upon the evolutionary framework (NSGA-II) in detection of a non-dominated set which performs highly in terms of fitness values. On the other hand, the evolutionary framework guarantees an efficient spreading of the solutions. The proposed adaptation seems, also, to be efficient in the coordination of local search components. Finally, the cross-dominance criterion defined in this chapter is an instrument for comparing two sets of solutions and thus monitor the algorithmic improvements. This information can be generally useful since it can be employed as a feedback in the design of an adaptive algorithm for multi-objective optimization problems.

4 Real World Application: Design of a DC Motor Speed Controller

Nowadays most motion actuators are set up with electric motors since they offer high performance in terms of power density, efficiency, compactness and lightness. On the other hand, in order to have satisfactory functioning of the motor, an effective control is needed. Basically, an efficient motor control can be achieved either by applying a complex and expensive control system (see [32], [33], [34], [35]) or by using a simple and cheap control system, e.g. Proportional Integral (PI) based [36], which requires a design often very difficult to perform. In the latter case, the control design of an electric motor consists of detecting those system parameters that ensure a good system response in terms of speed and current behavior. This leads to a multi-objective optimization problem too complex for analytical solution [37]. Moreover, the application of classical design strategy [38], [39], [40] likely leads to unsatisfactory results. Thus, during recent years, interest in computational intelligence techniques has increased (see [41], [42] and [43]).

This chapter attempts to apply the CDMOMA to the control design of the Direct Current (DC) Motor whose electrical and mechanical features are shown in Table 4.

Table 4. DC Motor Nameplate

Parameter	Value
Armature resistance	2.13 Ω
Armature induction	0.0094 H
Moment of inertia	$2.4e^{-6}$ $Kg \cdot m^2$
Rated armature voltage	12 V
Rated armature current	1.2 A
Rated load torque	0.0213 Nm
Rated speed	400 rad/s

Fig. 24 shows the block diagram of the control scheme.

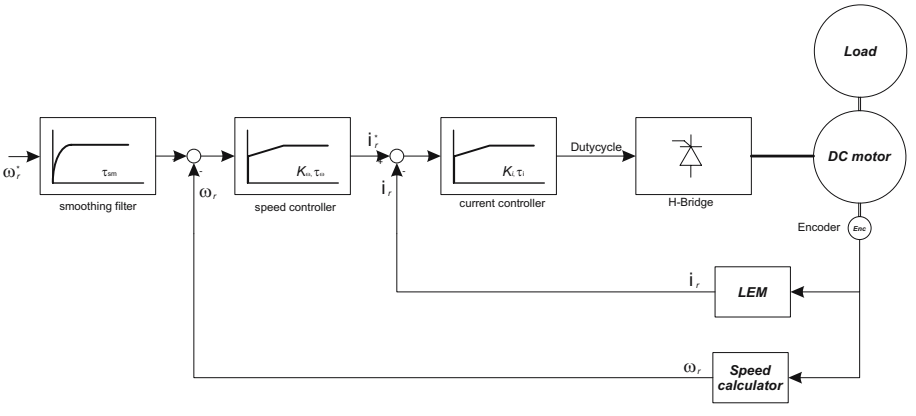


Fig. 24. Block diagram of a DC motor control

The control scheme is based on dynamic equations of the motor:

$$v_a = R_a \cdot i_a + L_a \cdot \frac{di_a}{dt} + e \tag{7}$$

$$v_f = R_f \cdot i_f + L_f \cdot \frac{di_f}{dt} \tag{8}$$

$$e = K\Phi \cdot \omega \tag{9}$$

$$T = K\Phi \cdot i_a \tag{10}$$

$$J \cdot \frac{d\omega}{dt} = T - T_r \tag{11}$$

where v_a is the voltage applied to the armature circuit, v_f is the voltage applied to the excitation circuit, R_a, R_f, L_a, L_f, i_a and i_f are the resistance, inductance and current for the armature and the excitement circuits respectively, T and T_r are the electromagnetic and load torque respectively, $K\Phi$ is the torque constant, ω is the rotor speed, J is the moment of inertia and e is the voltage generated by the rotor of the electric machine while rotating.

The DC motor control system is composed of two PI controllers. The first is used to control current and the second speed. The PIs transfer functions of the current and the speed controls are respectively $K_{pi} + \frac{K_{ii}}{s}$ and $K_{p\omega} + \frac{K_{i\omega}}{s}$. The speed reference is pre-filtered through a smoothing filter to reduce overshoot and the current required by the control in response to a speed step. The transfer function of the smoothing filter is $\frac{1}{(1+\tau_s m)}$.

With reference to Fig. 24, the control design consists of determining the parameters K_{pi}, K_{ii}, $K_{p\omega}$, $K_{i\omega}$ and τ_{sm} which guarantee very small values in rise and settling time, steady state error and overshoots. The decision space $H \subset \Re^5$ is a five dimensional hyper-rectangle given by the Cartesian product constructed around solution x_0 obtained by applying the classical symmetrical optimum (SO) criterion to design the speed regulator and the absolute value optimum (AVO) criterion to design the current regulator [44]. The lower and upper bounds of each interval have been set according to the following equations:

$$x_{lb}(i) = 10^{-6} \cdot x_0(i) \tag{12}$$

$$x_{ub}(i) = 3 \cdot x_0(i) \tag{13}$$

In order to evaluate the performance of each candidate solution, the four speed and load torque step training test shown in Fig. 25 is simulated by means of Matlab/Simulink as a discrete time control drive in order to realistically emulate an industrial digital drive. The control design of the DC Motor consists of determining a solution

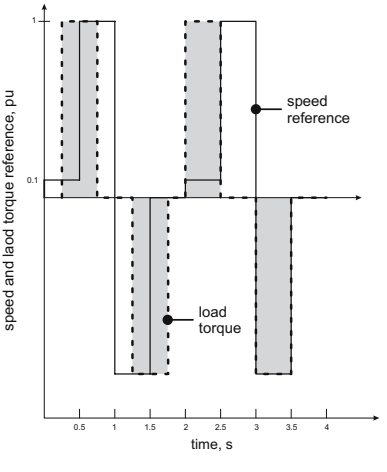


Fig. 25. Training test is a combination of speed commands and load torque

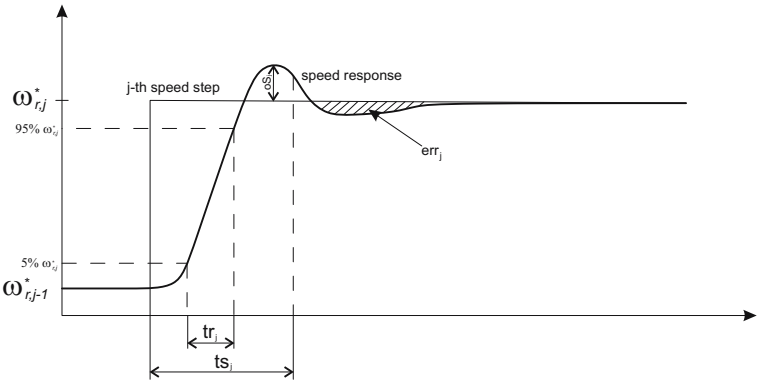


Fig. 26. j_{th} speed step of the training and values for objective function evaluation

$x = [K_{pi}, K_{ii}, K_{p\omega}, K_{i\omega}, \tau_{sm}]$ which satisfies the following multi-objective optimization problem:

$$\left.\begin{array}{l} \text{Minimize } \sum_{j=1}^{4} oS_j, \sum_{j=1}^{4} tr_j, \sum_{j=1}^{4} ts_j, \sum_{j=1}^{4} err_j \\ \text{Within } H \end{array}\right\} \tag{14}$$

where oS_j is the overshoot, tr_j the rise time, ts_j the settling time and err_j the sum of the absolute values of the speed error in settling condition during the j_{th} trial step.

Fig. 26 illustrates oS_j, tr_j, ts_j and err_j for the generic j_{th} step of the training test. Finally, it must be remarked that, since each fitness evaluation requires a computationally expensive simulation test (8 s for each evaluation, see [45]), the problem is very demanding in terms of computational overhead.

The CDMOMA has been applied and its performance compared with the SPEA-2 and the NSGA-II. For each algorithm, 25 runs have been performed with a population size equal to 40. The average and variance values of Δ are listed in Table 5. The values related to Υ are obviously missing since the actual Pareto is unknown.

The results in Table 5 show that for the problem under study, the SPEA-2 seems to have slightly better performance than the other algorithms in terms of spreading of the solutions.

In order to detect the most suitable control design the following decision making process has been implemented. For each algorithm, all the final populations related to the 25 runs have been merged. At first, all the individuals having an error $\sum_{j=1}^{4} err_j$ above a threshold value (200 rad) are discarded. This condition means that during the entire

Table 5. Δ values for the DC Motor Control Design

NSGA-II		CDMOMA		SPEA-2	
$\bar{\Delta}$	σ^2_Δ	$\bar{\Delta}$	σ^2_Δ	$\bar{\Delta}$	σ^2_Δ
0.8951	$2.5601 \cdot 10^{-2}$	0.8375	$1.4176 \cdot 10^{-2}$	0.6858	$1.5762 \cdot 10^{-2}$

Table 6. Single objective values after the decision making

	$\sum_{j=1}^{4} oS_j$	$\sum_{j=1}^{4} tr_j$	$\sum_{j=1}^{4} ts_j$	$\sum_{j=1}^{4} err_j$
NSGA-II	8	0.1780	0.2640	8.3330
CDMOMA	10	0.1750	0.2610	7.9150
SPEA-2	145	0.2250	0.2920	19.4240

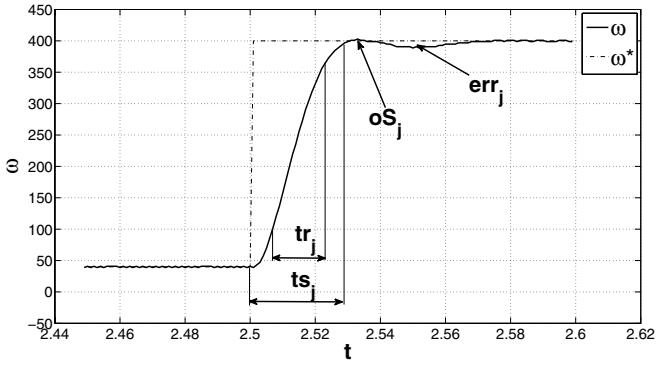


Fig. 27. Zoom detail of the speed response of NSGA-II solution

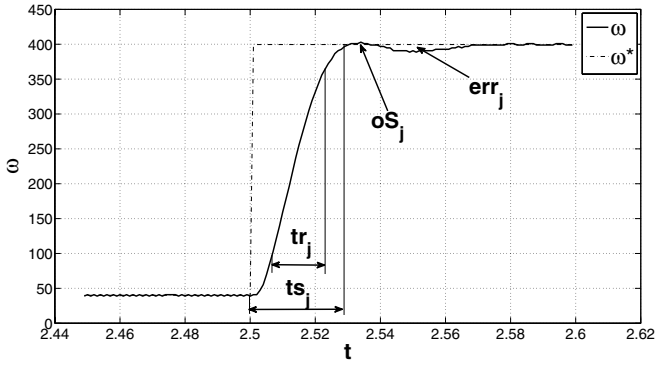


Fig. 28. Zoom detail of the speed response of CDMOMA solution

training test, the overall deviation (the sum of all the deviations) of the rotor position from the reference axis should not be more than 200 rad. Amongst the remaining solutions, all the individuals having a settling time $\sum_{j=1}^{4} ts_j$ above 0.35 s are discarded; amongst the remaining solutions, all the individuals having a rise time $\sum_{j=1}^{4} tr_j$ above 0.2

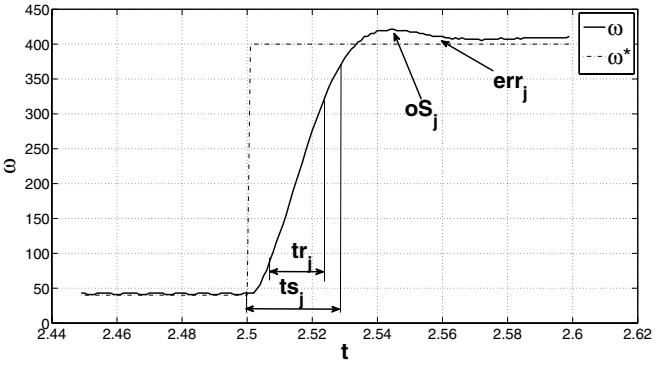


Fig. 29. Zoom detail of the speed response of SPEA-2 solution

s are discarded; amongst the remaining solutions, the solution having a minimal value in the overshoot $\sum_{j=1}^{4} oS_j$ is eventually selected.

The single objective values given by solutions obtained at the end of the decision making process are listed in Table 6.

It can be noticed that the solution returned by the SPEA-2 is dominated by the solutions returned by both NSGA-II and CDMOMA. The performance of the NSGA-II and CDMOMA solutions are, on the contrary, rather similar; both the algorithms seem to have high performance for this problem.

For the sake of clarity, a zoom detail of the speed response which graphically highlights the difference in performance is shown in Fig. 27, 28 and 29 for the NSGA-II, the CDMOMA and the SPEA-2 respectively.

5 Conclusion

This chapter proposes the Cross-Dominant Multi-Objective Memetic Algorithm (CDMOMA), which is a memetic algorithm composed of the NSGA-II as an evolutionary framework and two local searchers adaptively integrated within the framework. The adaptation is based on a criterion which attempts to coordinate the local search by monitoring improvements in the set of non-dominated solutions. Novel contributions of this chapter are: the implementation proposed here for the Multi-Objective Rosenbrock Algorithm, the concept of Cross-Dominance and its employment within a Memetic Framework, and the probabilistic scheme based on the Wigner semicircle distribution.

The CDMOMA seems very promising in several test cases by either reaching the theoretical Pareto or outperforming the popular NSGA-II and SPEA-2. In only one test case (ZDT4) out of eight test problems the CDMOMA failed in detecting a good set of solutions. Numerical results related to the real-world problem analyzed here seem also to conclude that the CDMOMA can be a promising approach.

A further improvement in the proposed approach will be in the detection of tailored local search components for some specific applications, in the design of efficient local

searchers which take into account the spreading property of the locally improved solution with respect to other individuals of the population and, finally, to propose a modification of the evolutionary framework in order to enhance its robustness over multiple runs.

Acknowledgement

The Authors wish to express their deep gratitude to Anna V. Kononova for the helpful discussions and precious suggestions.

References

1. Miettinen, K.: Nonlinear Multiobjective Optimization. Springer, Berlin (1998)
2. Schaffer, J.D.: Multiple objective optimization with vector evaluated genetic algorithms. In: Proc. of 1st International Conference on Genetic Algorithms and Their Applications, pp. 93–100 (1985)
3. Deb, K.: Multi-objective Optimization using Evolutionary Algorithms, pp. 147–149. John Wiley and Sons LTD, Chichester (2001)
4. Coello, C.A.C., Veldhuizen, D.A.V., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, New York (2002)
5. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evolutionary Computation 6(2), 182–197 (2002)
6. Zitzler, E., Laumanns, M., Thiele, L.: SPEA-2: Improving the Strength Pareto Evolutionary Algorithm. Tech. Rep. 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland (2001)
7. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Tech. Rep. 790, Caltech Concurrent Computation Program (1989)
8. Neri, F., Toivanen, J., Mäkinen, R.A.E.: An adaptive evolutionary algorithm with intelligent mutation local searchers for designing multidrug therapies for HIV. In: Applied Intelligence, vol. 27, pp. 219–235. Springer, Heidelberg (2007) (special issue on Computational Intelligence in Medicine and Biology)
9. Neri, F., Toivanen, J., Cascella, G.L., Ong, Y.S.: An adaptive multimeme algorithm for designing HIV multidrug therapies. IEEE/ACM Transactions on Computational Biology and Bioinformatics, Special Issue on Computational Intelligence Approaches in Computational Biology and Bioinformatics 4(2), 264–278 (2007)
10. Krasnogor, N.: Toward robust memetic algorithms. In: Hart, W.E., Krasnogor, N., Smith, J.E. (eds.) Recent Advances in Memetic Algorithms, pp. 185–207. Springer, Berlin (2004)
11. Ong, Y.S., Keane, A.J.: Meta-lamarkian learning in memetic algorithms. IEEE Transactions on Evolutionary Computation 8(2), 99–110 (2004)
12. 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 36(1), 141–152 (2006)
13. Krasnogor, N., Smith, J.: A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Transactions on Evolutionary Computation 9, 474–488 (2005)
14. Knowles, J., Corne, D.: Memetic algorithms for multiobjective optimization: issues, methods and prospects. In: Hart, W.E., Krasnogor, N., Smith, J.E. (eds.) Recent Advances in Memetic Algorithms, pp. 313–352. Springer, Heidelberg (2004)
15. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between genetic search and local search in hybrid evolutionary multi-criterion optimization algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, New York, USA, pp. 1301–1308 (July 2002)

16. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between genetic search and local search in memetic algorithms for multiobjective permutation flow shop scheduling. IEEE Transactions on Evolutionary Computation 7, 204–223 (2003)
17. Ishibuchi, H., Hitotsuyanagi, Y., Nojima, Y.: An empirical study on the specification of the local search application probability in multiobjective memetic algorithms. In: Proc. of the IEEE Congress on Evolutionary Computation, pp. 2788–2795 (September 2007)
18. Srinivas, N., Deb, K.: Multi-objective function optimization using non-dominated sorting genetic algorithms. Evolutionary Computation 2, 221–248 (1995)
19. Deb, K., Goyal, M.: A combined genetic adaptive search (GeneAS) for engineering design
20. Deb, K., Agrawal, R.B.: Simulated binary crossover for continuous search space. Complex SYstems 9(2), 115–148 (1995)
21. Deb, K., Kumar, A.: Real-coded genetic algorithms with simulated binary crossover: Studies on multi-modal and multi-objective problems. Complex SYstems 9(6), 431–454 (1995)
22. Suman, B.: Study of simulated annealing based algorithms for multiobjective optimization of a constrained problem. Computers and Chemical Engineering 28
23. Rosenbrock, H.H.: An automatic method for findong the greatest or least value of a function. The Computer Journal 3(3), 175–184 (1960)
24. Birkho, G., Lane, S.M.: A Survey of Modern Algebra. Macmillan, Basingstoke (1953)
25. Szu, H., Hartley, R.: Fast simulated annealing. Physiscs Letters A 122, 157–162 (1987)
26. Fonseca, C.M., Fleming, P.J.: Multiobjective optimization and multiple constraint handling with evolutionary algorithms part-ii aplication example. IEEE Transctions on Systems,Man and Cyberntics, PartA, Systems and Humans, 38–47 (1998)
27. Poloni, C.: Hybrid GA for multi-objective aerodynamic shape optimization. In: Winter, G., Periaux, J., Galan, M., Cuesta, P. (eds.) Gentic algorithms in egineering and computer science, pp. 397–414. Wiley, Chichester (1997)
28. Kursawe, F.: A variant of evolution strategies for vector optimization. In: Schwefel, H.-P., Mäner, R. (eds.) Parallel Problem Solving from Nature, vol. I. Springer, Berlin
29. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms:empirical results. Evolutionary Computaion Journal 8, 173–195 (2000)
30. Multi-objective genetic algorithms:problem difficulties and construction of test functions. Evolutionary Computaion Journal 7, 205–230 (1999)
31. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Schoenauer, M., et al. (eds.) Proceedings of the Parallel Problem Solving from Nature VI Conference. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
32. Åström, K.J., Wittenmark, B.: Adaptive Control, 2nd edn. Prentice Hall, Englewood Cliffs (1994)
33. Slotine, J.-J., Li, W.: Applied Nonlinear Control. Prentice Hall, Englewood Cliffs (1990)
34. Khorrami, H.M.F., Krishnamurthy, P.: Modeling and Adaptive Nonlinear Control of Electric Motors. Springer, Heidelberg (2003)
35. Jain, L.C., de Silva, C.W.: Intelligent Adap.tive Control: Industrial Applications. CRC, Boca Raton (1998)
36. Åström, K.J., Hägglund, T.: The future of PID control. Control Engineering Practice 9(13), 1163–1175 (2001)
37. Panagopoulos, H., Åström, K.J., Hägglund, T.: Design of PID controllers based on constrained optimisation. IEE Proceedings - Control Theory & Applications 149, 32–40 (2002)
38. Dury, W.: Control Techniques Drives & Controls Handbook. Institution Electrical Engineers (2001)
39. Leonhard, W.: Control of Electrical Drives, 2nd edn. Springer, Heidelberg (2001)
40. Krishnan, R.: Electronic Motor Drives: Modeling, Analysis and Control. Prentice-Hall, Englewood Cliffs (2001)

41. Zanchetta, P., Sumner, M., Cupertino, F., Marinelli, M., Mininno, E.: On-line and off-line control design in power electronics and drives using genetic algorithms. In: Proceedings of the IEEE Industry Applications Conference, pp. 864–871 (2004)
42. Cascella, G.L., Salvatore, N., Sumner, M., Salvatore, L.: On-line simplex-genetic algorithm for self-commissioning of electric drives. In: Proceedings 11th EPE, pp. 277–283 (2005)
43. Caponio, A., Cascella, G.L., Neri, F., Salvatore, N., Sumner, M.: A fast adaptive memetic algorithm for on-line and off-line control design of pmsm drives. IEEE Transactions on System Man and Cybernetics-part B, special issue on Memetic Algorithms 37(1), 28–41 (2007)
44. Szklarski, K.J.L., Horodecki, A.: Electric Drive Systems Dynamics. Elsevier Science Ltd., Amsterdam (1990)
45. del Toro Garcia, X., Neri, F., Cascella, G.L., Salvatore, N.: A surrogate assisted hooke-jeeves algorithm to optimize the control system of a pmsm drive. In: Proceedings of IEEE International Symposium on Industrial Electronics, vol. 1, pp. 347–352 (2006)

A Memetic Algorithm for Dynamic Multiobjective Optimization

Tapabrata Ray, Amitay Isaacs, and Warren Smith

University of New South Wales, Australian Defence Force Academy,
ACT 2600, Australia
{t.ray,a.isaacs,w.smith}@adfa.edu.au

Summary. Dynamic multi-objective optimization (DMO) is one of the most challenging class of multi-objective optimization problems where the objective function(s) change over time and the optimization algorithm is required to identify the corresponding Pareto optimal solutions with minimal time lag. DMO has received very little attention in the past and none of the existing multi-objective algorithms perform too well on the set of newly proposed DMO test problems. In this chapter, we introduce a memetic algorithm (MA) to deal with such class of problems. The memetic algorithm employs an orthogonal epsilon-constrained formulation to deal with multiple objectives and a sequential quadratic programming (SQP) solver is embedded as its local search mechanism to improve the rate of convergence. The performance of the memetic algorithm is compared with an evolutionary algorithm (EA) embedded with a Sub-EA on two benchmarks FDA1 and modified FDA2. The memetic algorithm consistently outperforms the evolutionary algorithm with Sub-EA for both FDA1 and modified FDA2 for all problem sizes. A variational study on the effect of the number of SQP iterations on the behavior of MA is included to highlight the benefits offered by MA for such class of problems.

Keywords: dynamic multi-objective optimization, memetic algorithm, evolutionary algorithm, orthogonal epsilon-constrained method.

1 Introduction

Over the years, a number of optimization algorithms such as evolutionary algorithms (EA), particle swarm optimization (PSO), differential evolution (DE) and hybrids such as memetic algorithms have been successfully used to solve a number of real life multi-objective optimization problems. However, most of the reported applications deal with static multi-objective optimization problems, where the objective functions do not change over time. For such classes of problems, the aim of an optimization algorithm is to identify a diverse set of non-dominated solutions that are close to the Pareto optimal front.

Dynamic multi-objective optimization is far more challenging as compared to its static counterpart. In order to solve a DMO problem effectively and efficiently, the optimization algorithm should posses mechanisms to (a) identify a change

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 353–367.
springerlink.com

in the objective function(s), (b) converge to the Pareto optimal set with a high rate of convergence, and (c) allow migration from a converged Pareto optimal set to another.

In terms of identifying a change in the objective function, there are basically two approaches reported in literature:- proactive and reactive. In proactive approach, a forecasting model is developed based on objective function behavior over time and a portion of the population is evolved based on an objective function derived through the forecasting model (1). Such an approach is suitable if the objective function follows a pattern over time as in many DMO test problems but may not be the best choice for all classes of problems. The reactive approach on the other hand responds to a change in the objective function only when it is detected and is a more generic approach to solving DMO problems. Performance of reactive model on some test cases under unpredictable parameter variations is reported in (2).

In order to solve a DMO problem efficiently, the underlying optimization algorithm should have a high rate of convergence. A hybrid search i.e. a combination of a global and a local search is particularly attractive for such problems as the rate of convergence of a hybrid search is better than a global search alone. Memetic algorithm (MA) (3; 4) is one such hybrid which is known for its high rate of convergence. MA integrates the global search properties of a population based approach with the exploitation properties of a local search algorithm. An excellent review of memetic algorithms has been presented by (5). However, the performance of MA is largely dependent on the correct choice of the local search strategies (memes), identification of the subset undergoing local improvements and the convergence criterion used in local search strategies. Furthermore, for MO problems in general, the MA should posses appropriate mechanisms to maintain diversity and an acceptable level of spread along the Pareto optimal front. Epsilon constraint formulation (6) is an attractive choice to deal with the above problem but needs a prudent choice of the constraints to uncover regions of the non-convex front. Adaptive GA with dynamic fitness function guided by fuzzy inference system to control crossover and mutation rates has been reported in (7).

The third aspect of DMO algorithm is particularly difficult to incorporate as migration from a converged Pareto optimal set to another may require substantial traversal in the variable space and thus lead to a larger time lag. A generic approach would be to use a random population instead of the converged Pareto solutions whenever a change in the objective function is detected. However, such an approach is likely to have larger time lag as compared to a migration from the converged Pareto set for problems where there is gradual migration of the variables. The recombination and mutation schemes employed within an EA are also expected to influence the convergence behavior.

In this chapter, we present a memetic algorithm that consists of an EA coupled with a SQP solver to deal with DMO problems. An orthogonal epsilon-constrained formulation is used to deal with multiple objectives where orthogonal sets of constraints are used to uncover all parts of the Pareto optimal front. The

underlying EA is embedded with a SQP solver for faster convergence. The MA with orthogonal epsilon constraint formulation attempts to solve a series of single objective constrained optimization problems to yield the set of non-dominated solutions. The details of the algorithm are outlined in Section 2 while its performance on the set of mathematical benchmarks FDA1 and modified FDA2 are presented in Section 3 followed by a summary of our findings in Section 4.

2 Memetic Algorithm for Dynamic MO Optimization

The framework for the proposed memetic algorithm is outlined in Algorithm 1. The structure is identical to an evolutionary algorithm framework with the only exception that MA utilizes gradient evolve mechanism while an EA uses Sub-EA evolve mechanism.

Algorithm 1. Memetic / Evolutionary algorithm for DMO problems

Require: $N_G > 1$ *(* Number of Generations *)*
Require: $N > 0$ *(* Population size *)*
1: P_1 = Initialize()
2: Evaluate(P_1)
3: **for** $i = 2$ to N_G **do**
4: **if** the function has changed **then**
5: Evaluate(P_{i-1})
6: **end if**
7: C_{i-1} = GradientEvolve(P_{i-1}) or C_{i-1} = SubEAEvolve(P_{i-1})
8: Evaluate(C_{i-1})
9: P_i = Reduce($P_{i-1} + C_{i-1}$)
10: **end for**

2.1 Initialization

The solutions in the initial population are initialized by selecting individual variable values in the specified range as given in Eq. 1.

$$x_i = \underline{x_i} + \mathcal{U}[0,1]\ (\overline{x_i} - \underline{x_i}) \qquad 1 \leq i \leq n \tag{1}$$

where x_i denotes the initialized variable, $\underline{x_i}$ and $\overline{x_i}$ are the lower and upper limits for ith variable respectively, n is the size of the design variable vector and $\mathcal{U}[0,1]$ is a uniform random number lying between 0 and 1.

2.2 Evaluation

The solutions in a population are evaluated using disciplinary analysis to calculate the objectives and the constraints. For dynamic problems, the function (objectives and/or constraints) behavior changes with time. The objective and

the constraint values of a candidate solution calculated in the previous generation may not be valid for the next generation. To check if the objective and/or constraint functions have changed with time, a solution is picked randomly from the population and re-evaluated. If the objective or the constraint values have changed, then the function behavior is assumed to have changed and the entire population is re-evaluated to obtain new values of the objectives and the constraints.

2.3 Gradient Evolve Method

The gradient evolve mechanism employs a sequential quadratic programming (SQP) solver to solve a series of single objective optimization problems generated through an orthogonal epsilon-constrained formulation of the multi-objective problem. Consider a bi-objective optimization problem as presented in Eq. 2.

$$\text{Minimize } f_1(\mathbf{x}),\ f_2(\mathbf{x}) \quad \text{subject to} \quad g_i(\mathbf{x}) \leq 0,\quad i = 1, \ldots, m \tag{2}$$

The extent of the Pareto optimal front is established by 2 separate single objective optimization problems given in Eq. 3.

$$\begin{aligned} &\text{Minimize } f_1(\mathbf{x}) \quad \text{subject to} \quad g_i(\mathbf{x}) \leq 0,\quad i = 1, \ldots, m \\ &\text{Minimize } f_2(\mathbf{x}) \quad \text{subject to} \quad g_i(\mathbf{x}) \leq 0, \\ &\quad i = 1, \ldots, m \end{aligned} \tag{3}$$

The solutions of the single optimization problems ($f_1^{\min}$ and $f_2^{\min}$) correspond to the extremities of the Pareto optimal front as shown in Figure 1. The rest of the Pareto optimal front is obtained by orthogonal epsilon-constrained reformation for each objective. The range of each objective is sub-divided into small intervals and a single objective optimization problem is solved for each of those

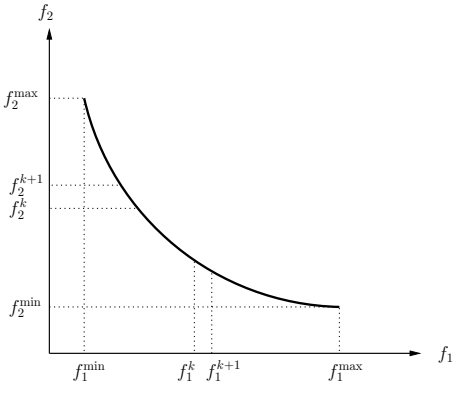


Fig. 1. Pareto optimal front for bi-objective optimization problem and formulation of orthogonal epsilon-constrained formulation

intervals. For $f_1(\mathbf{x})$, the range of values is $[f_1^{\min}, f_1^{\max}]$ and is sub-divided into equal intervals $[f_1^k, f_1^{k+1}]$ as shown in Figure 1. For each interval $[f_1^k, f_1^{k+1}]$, a single objective optimization problem is solved as given in Eq. 4.

$$\begin{aligned} \text{Minimize} \quad & f_2(\mathbf{x}) \\ \text{subject to} \quad & f_1^k \leq f_1(\mathbf{x}) \leq f_1^{k+1} \\ & g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m \end{aligned} \tag{4}$$

Similarly, the range for $f_2(\mathbf{x})$ is $[f_2^{\min}, f_2^{\max}]$ and the single optimization problem is solved on each interval $[f_2^k, f_2^{k+1}]$ (Eq. 5).

$$\begin{aligned} \text{Minimize} \quad & f_1(\mathbf{x}) \\ \text{subject to} \quad & f_2^k \leq f_2(\mathbf{x}) \leq f_2^{k+1} \\ & g_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m \end{aligned} \tag{5}$$

Both, the convex and the non-convex parts of the Pareto optimal front can be captured using such an approach.

Solution of a single objective optimization problem requires a starting point. For each of the single optimization problems, a solution is picked up from the parent population randomly as the starting point. The number of function evaluations are dictated by the number of SQP iterations.

2.4 Sub-EA Evolve Method

In the Sub-EA evolve mechanism, the parent population of EA is allowed to evolve using an embedded evolutionary algorithm (Sub-EA) with conventional crossover and mutation operators. The main steps of the Sub-EA evolve method are given in Algorithm 2. In the present study, simulated binary crossover (SBX) and polynomial mutation operators are used (8). The population size for sub-EA is the same as that of the EA. For each generation of EA, multiple generations are evolved using sub-EA. The number of function evaluations is dictated by the number of generations of sub-EA.

Algorithm 2. Sub-EA Evolution Algorithm

Require: $N'_G > 1$ /* *Number of Sub-EA Generations* */
Require: P_j /* *Parent Population of jth generation* */
1: $P'_1 = P_i$
2: Evaluate(P'_1)
3: **for** $i = 2$ to N'_G **do**
4: C'_{i-1} = Crossover(P'_{i-1})
5: C'_{i-1} = Mutation(C'_{i-1})
6: Evaluate(C'_{i-1})
7: P'_i = Reduce($P'_{i-1} + C'_{i-1}$)
8: **end for**
9: $C_j = P'_{N'_G}$ /* *Sub-EA evolved population is offspring population for EA* */

2.5 Reduction

The reduction process retains N elite solutions for the next generation from a set of $2N$ solutions (parent and offspring population). Non-dominated sorting is used to sort $2N$ solutions into non-dominated fronts and within each front, the solutions are ranked using crowding distance sort (9). The reduction procedure is as follows.

1. If there are N or more feasible solutions,
   - N feasible solutions are selected in the order of non-dominated fronts and decreasing order of crowding distance in each front.
2. If there are less than N feasible solutions,
   - all the feasible solutions are selected, and
   - remaining solutions are selected from infeasible solutions in the increasing order of maximum constraint violation value.

Non-dominated sorting and crowding distance based sorting helps maintain the diversity in the population and spreads the solutions along the Pareto front.

3 DMO Test Problems

A static multi-objective optimization problem has fixed Pareto Optimal Front (POF) and corresponding solutions in the variable space denoted by POS. For dynamic multi-objective optimization problems POF and/or POS vary with time. A classification of multi-objective optimization problems are presented by Farina *et. al.* (10). Test problems FDA1 (10) and modified FDA2 (11) are used for this study. The FDA1 test problem is as defined by Eq. 6.

$$\text{FDA1:} \quad \begin{cases} f_1(\mathbf{x}_I) = x_1, \\ f_2(\mathbf{x}_{II}) = g \times h, \\ g(\mathbf{x}_{II}) = 1 + \sum\limits_{x_i \in \mathbf{x}_{II}} (x_i - G(t))^2 \\ h(f_1, g) = 1 - \sqrt{f_1/g} \\ G(t) = \sin(0.5\pi t), \quad t = \dfrac{1}{n_t} \left\lfloor \dfrac{\tau}{\tau_T} \right\rfloor \\ \mathbf{x}_I = (x_1) \in [0, 1], \\ \mathbf{x}_{II} = (x_2, \ldots, x_n) \in [-1, 1]. \end{cases} \tag{6}$$

where τ is the generation counter, τ_T is the number of generations for which t remains fixed, and n_t is the number of distinct steps in t. The values used for the variables are: $n = 11$, $n_t = 10$, and $\tau_T = 5$.

Every time t changes, the POS changes but the POF remains same corresponding to $f_2 = 1 - \sqrt{f1}$. At the first time instant $t = 0$, the POS solutions correspond to $x_i = 0$ for $x_i \in \mathbf{x}_{II}$. With changes in t, each variable in $\mathbf{x}_{II}$ changes in a sinusoidal manner corresponding to a change in $G(t)$ as seen in Figure 2.

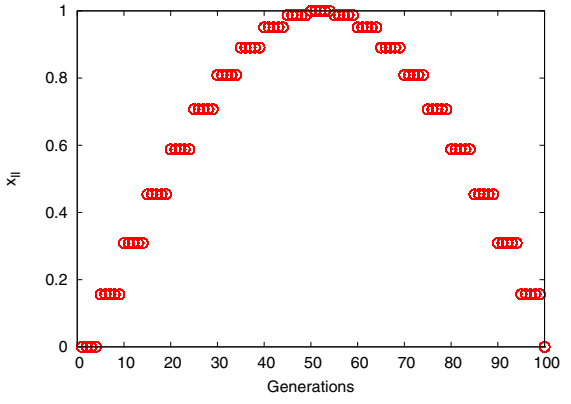


Fig. 2. The variation of variable values in x_{II} with time for FDA1

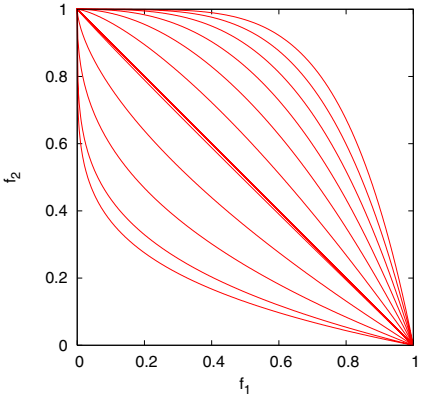


Fig. 3. FDA2$_{mod}$ POF changes from a convex shape to a concave shape with time

The definition for modified FDA2 (FDA2$_{mod}$) is given in Eq. 7.

$$\text{FDA2}_{mod}: \begin{cases} f_1(\mathbf{x}_I) = x_1, \\ f_2(\mathbf{x}_{II}, \mathbf{x}_{III}) = g \times h \\ g(\mathbf{x}_{II}, \mathbf{x}_{III}) = 1 + \sum\limits_{x_i \in \mathbf{x}_{II}} x_i^2 + \sum\limits_{x_i \in \mathbf{x}_{III}} (x_i + 1)^2 \\ h(f_1, g) = 1 - \left(\dfrac{f_1}{g}\right)^{H(t)} \\ H(t) = 0.2 + 4.8t^2, \quad t = \dfrac{1}{n_t}\left\lfloor \dfrac{\tau}{\tau_T} \right\rfloor \\ \mathbf{x}_I = (x_1) \in [0, 1], \\ \mathbf{x}_{II} = (x_2, \ldots, x_{r_1}) \in [-1, 1], \\ \mathbf{x}_{III} = (x_{r_1+1}, \ldots, x_n) \in [-1, 1]. \end{cases} \tag{7}$$

where τ is the generation counter, τ_T is the number of generations for which t remains fixed, and n_t is the number of distinct steps in t. The values used for the variables are: $n = 31$, $r_1 = 16$, $n_t = 10$, and $\tau_T = 5$. The POF for FDA2$_{mod}$ changes from a convex shape to a concave shape as t varies as shown in Figure 3.

A series of performance metrics for multi-objective optimization have been reported in literature (12; 13). These performance metrics use various measures on the non-dominated set of solutions such as cardinality, distance, volume, spread, *etc.* For the test problems used in this study, the POF and the POS variations are known and performance metrics based on absolute measures can be used. One such metric is the generational distance. The generational distance is the measure of the distance between the Pareto optimal front and the non-dominated solution set (14). The metric is defined as follows:

Let F^* denote the Pareto Optimal Front (POF) and S^* denote the Pareto Optimal Set (POS). The generational distance of a set of non-dominated solutions (F) with respect to the POF $F*$ is given by,

$$G(F, F^*) = \frac{1}{|F|} \left(\sum_{i=1}^{n} (d_i)^p \right)^{\frac{1}{p}},$$

where d_i is the Euclidean distance between the ith non-dominated solution of F and the nearest member of the Pareto front F^*, and $|F|$ is the number of elements in the non-dominated set. Most often $p = 2$. The same metric can also be used in the variable space to measure the generational distance of the non-dominated solutions in the variable space (S) with respect to the POS S^*.

$$G(S, S^*) = \frac{1}{|S|} \left(\sum_{i=1}^{n} (d_i)^p \right)^{\frac{1}{p}},$$

4 Results

4.1 Experimental Setup

A population size of 40 is evolved over 100 generations for both MA and EA. Each generation corresponds to a single time step. Both the algorithms are run for 100 time steps. For EA, the parameters of sub-EA (crossover and mutation parameters) are varied along with the random seed, resulting in a total of 32

Table 1. Parameters Values for 32 experimental runs of EA

Random Seed	10, 20
Crossover Probability	0.8, 0.9
Crossover Distribution Index	10, 20
Mutation Probability	0.05, 0.1
Mutation Distribution Index	20, 50

experimental runs. Since there are no additional parameters for MA, 32 different values of random seed are chosen. Shown in Table 1 are the parameter values used for various runs for EA. The random seed values for MA used are $10, 20, \ldots, 320$.

To ensure that EA and MA use the same number of function evaluations, first an estimate of number of function evaluations used by MA is obtained based on few runs. Gradient based algorithm uses a tolerance criterion to terminate the algorithm, or alternately, it can use a fixed iteration count. For MA the number of SQP iterations is fixed at 2 and the corresponding number of function evaluations is used to determine the number of SubEA generations. For FDA1, function evaluations corresponding to 2 iterations of SQP are equivalent to 70 generations of sub-EA. For FDA2$_{mod}$, function evaluations corresponding to 2 iterations of SQP are equivalent to 190 generations of SubEA.

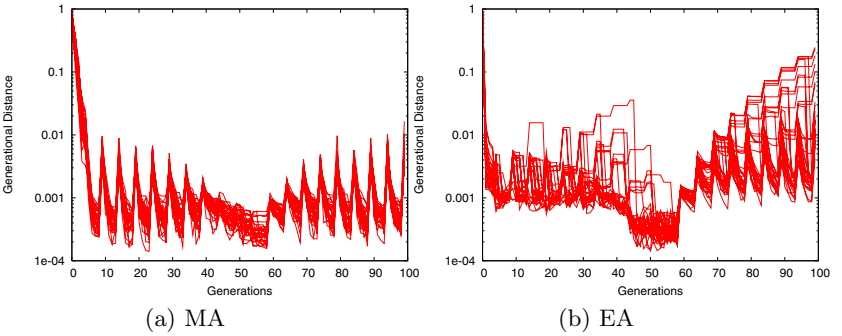
4.2 Results of FDA1

The statistics of multiple runs for MA and EA are presented in Table 2. EA uses a fixed number of function evaluations as dictated by the population size and the number of generations. As MA uses gradient based search, the number of function evaluations vary across multiple runs. The difference in the minimum and maximum number of function evaluations for MA is 384 (283,167-282,783) which is approximately 0.1% of the average number of function evaluations. Thus the number of function evaluations for MA and EA are considered equivalent.

The performance of MA and EA is compared using the generational distance metric. Variation in the generational distance for POF across multiple runs of

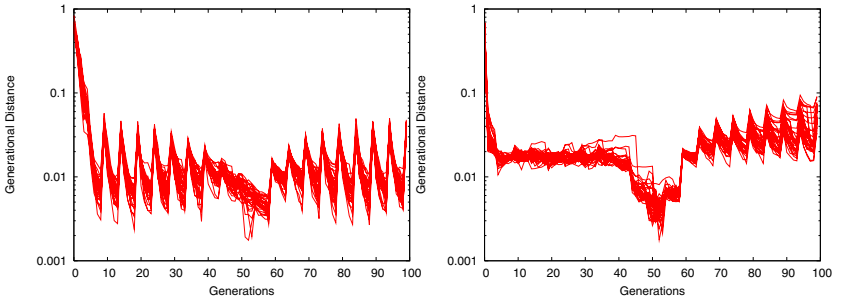
Table 2. Function evaluations statistics for MA and EA for FDA1

	Min.	Avg.	Max.
MA	282,783	283,001	283,167
EA	282,099	282,099	282,099



(a) MA (b) EA

Fig. 4. FDA1: Generational Distance metric for the Pareto optimal fronts (POF) obtained by MA and EA for all the experimental runs



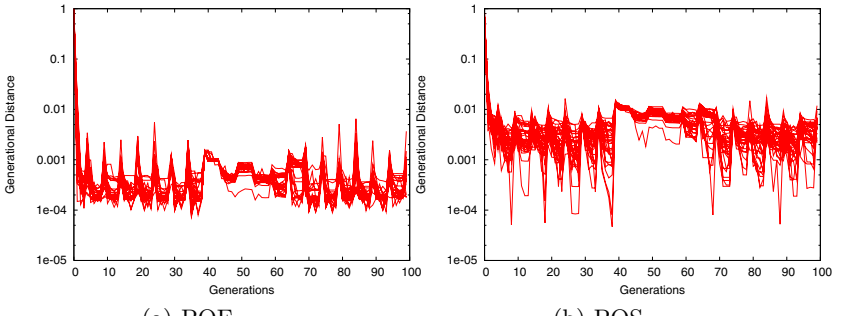
(a) MA (b) EA

Fig. 5. FDA1: Generational Distance metric for the Pareto optimal sets(POS) obtained by MA and EA for all the experimental runs

MA and EA are shown in Figure 4. The POF generational distance is quite high in the beginning and reduces in next few generations. At every interval of $t = 5$, there is a sudden increase in the generational distance as seen in Figure 4(a). This increase corresponds to the change in the function form for $t = 5$. The POF generational distance using MA is consistently lower than EA across the generations. Since the movement of POS is very small between time steps 40 and 60, the POF generational distance for EA and MA shows a dip as in Figure 4(b).

Shown in Figure 5 is the variation in the POS generational distance across multiple runs of MA and EA. The POS generational distance in the variable space is calculated for the solutions that are non-dominated in objective space. The POS generational distance using MA (Figure 5(a)) shows a similar trend as that of POF generational distance (Figure 4(a)). Small movement of POS during the time steps 40 and 60 is visible prominently in the POS generational distance using EA as shown in Figure 5(b).

When the SQP search in MA is allowed to run for more number of iterations, the solutions converge closer to the Pareto front as seen in Figure 6(a) and



(a) POF (b) POS

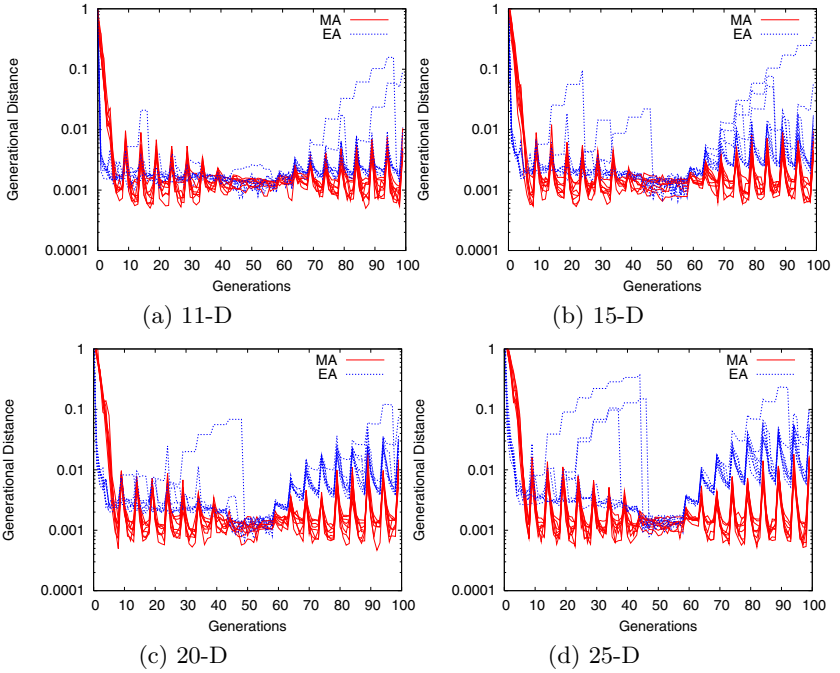
Fig. 6. FDA1: Generational Distance metric for the POF and the POS obtained by MA when the SQP search is run for 10 iterations

Table 3. Function evaluations used by MA with SQP search executed for 2 and 10 iterations for FDA1

SQP Iterations	Function evaluations		
	Min.	Avg.	Max.
2	282,783	283,001	283,167
10	492,500	497,568	501,678

Table 4. Function Evaluations used by MA and EA for FDA1 with different number of design variables

	MA			EA	
dim.	Min	Avg.	Max	Avg.	SubEA gens.
11	282,849	282,996	283,091	282,099	70
15	375,434	375,561	375,644	361,299	90
20	491,083	491,271	491,446	480,099	120
25	606,786	606,995	607,240	598,899	150



(a) 11-D (b) 15-D

(c) 20-D (d) 25-D

Fig. 7. FDA1: Generational Distance metric for the POF obtained by MA and EA for different number of design variables

(Figure 6(b)). The improvement in performance of MA is at the cost of increased number of function evaluations as seen from Table 3.

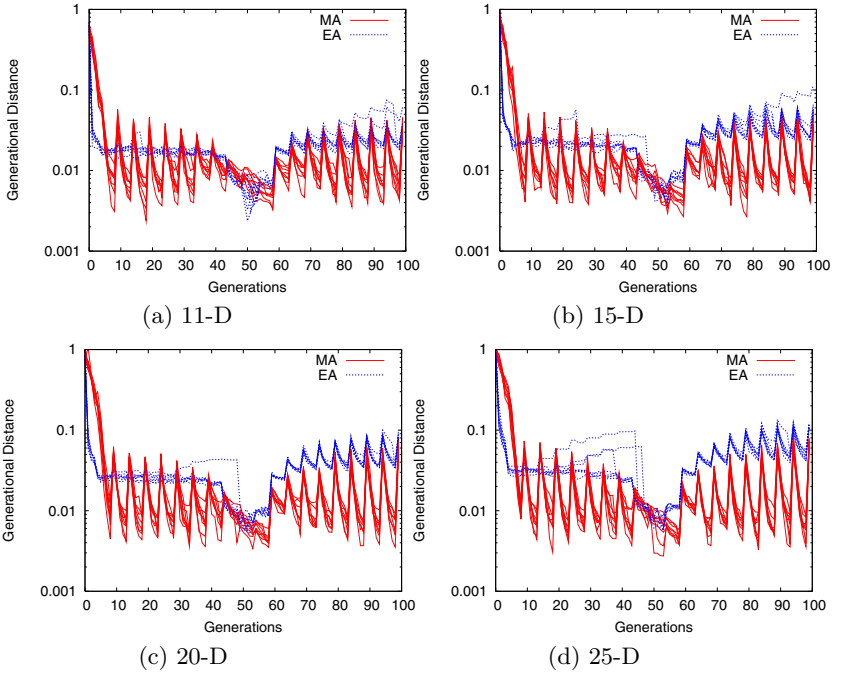
4.3 Results of FDA1 Problem Size Variation

Increasing the problem size has an immediate effect on the number of function evaluations used by MA as the gradient calculation requires additional evaluations. The number of function evaluations used by MA for different number of design variables (change in the dimension of x_{II}) is given in Table 4. The equivalent number of function evaluations for EA are obtained by increasing the number of sub-EA generations as given in Table 4.

The effects of increase in the number of design variables on the POF generational distance can be observed through a comparison between Figure 4 with Figure 7 and on the POS generational distance can be seen through a comparison between Figure 5 with Figure 8. MA consistently outperforms EA for all the problem sizes in both POF and POS.

4.4 Results of FDA2

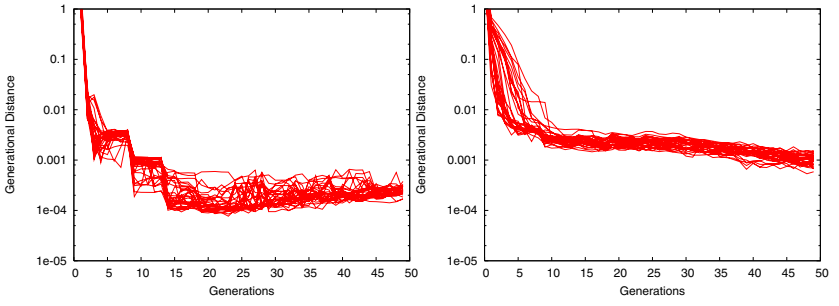
The POF generational distance variation by MA and EA for FDA2$_{mod}$ is presented in Figure 9. Its clear that MA is able to maintain lower generational



(a) 11-D (b) 15-D

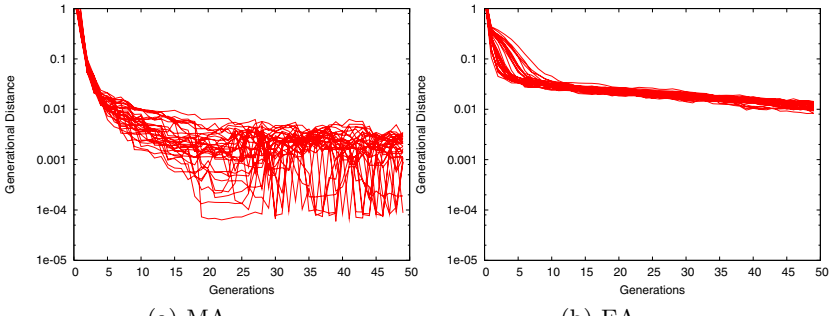
(c) 20-D (d) 25-D

Fig. 8. FDA1: Generational Distance metric for the POS obtained by MA and EA for different number of design variables



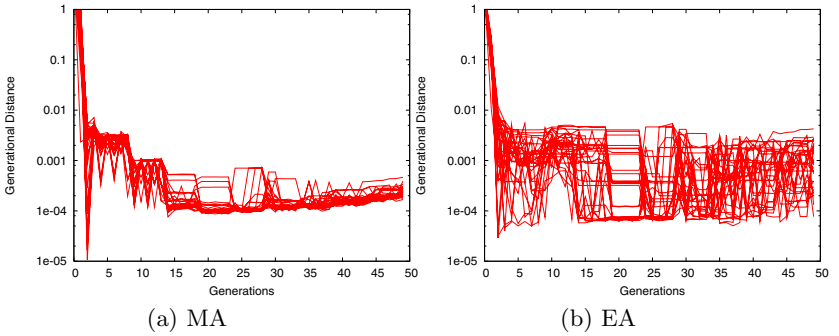
(a) MA (b) EA

Fig. 9. FDA2$_{mod}$: Generational Distance metric for the Pareto optimal fronts (POF) obtained by MA and EA for all the experimental runs



(a) MA (b) EA

Fig. 10. FDA2$_{mod}$: Generational Distance metric for the Pareto optimal sets (POS) obtained by MA and EA for all the experimental runs



(a) MA (b) EA

Fig. 11. FDA2$_{mod}$: Generational Distance metric for the POF and the POS obtained by MA when the SQP search is run for 10 iterations

distance as compared to EA over the generations. The corresponding POS generational distance for MA and EA is presented in Figure 5. It is interesting to observe that although POS generational distance for MA is lower than EA, it exhibits a significant fluctuation whereas EA presents a relatively flat behavior.

The generational distance variation (POF and POS) for MA with an increase in the number of SQP iterations from 2 to 10 is presented in Figure 11. An increase in the number of SQP iterations, improves the generational distance in both POF and POS as observed for FDA1.

5 Summary and Conclusion

In this chapter, a memetic algorithm is presented for dynamic multi-objective optimization. Currently, DMO problems are considered intractable and there is considerable interest to develop optimization methods to solve such classes of problems. A memetic algorithm is proposed to solve DMO problems and the results on the test functions clearly indicate its superiority over EA based approaches. The memetic algorithm is embedded with a sequential quadratic programming (SQP) solver as a local search mechanism for an improved rate of convergence. An orthogonal epsilon-constrained formulation is used to reformulate a multi-objective optimization problem as series of single objective optimization problems and uncover both convex and non-convex part of the Pareto optimal front. The algorithm is designed as a reactive model, where it acts on migration only when a change is detected in the objective function. Upon detecting a change in the objective function, existing solutions are used as starting points for SQP search although a random start points for SQP is also a possibility. The performance of the proposed MA is compared with an evolutionary algorithm undergoing secondary evolution using Sub-EA for two standard benchmarks FDA1 and modified FDA2 over a range of problem dimensions and the number of SQP iterations. The results clearly indicate that MA outperforms EA for all problem sizes for the same computational cost delivering solutions that are closer to the Pareto optimal front in both the objective and variable space and stands out as a promising alternative for DMO problems. The algorithm is currently being tested for online UAV system identification and control.

References

[1] Hatzakis, I., Wallace, D.: Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach, Seattle, Washington, USA, pp. 1201–1208. ACM, New York (2006)
[2] Amato, P., Farina, M.: An ALife-inspired evolutionary algorithm for dynamic multiobjective optimization problems (2003)
[3] Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical report, Caltech (1989)
[4] Ong, Y.S., Keane, A.: Meta-lamarckian learning in memetic algorithms. IEEE Transactions on Evolutionary Computation 8, 99–110 (2004)

[5] Krasnogor, N., Smith, J.: Multimeme algorithms for the structure prediction and structure comparison of proteins. In: GECCO, pp. 42–44 (2002)
[6] Chankong, V., Haimes, Y.Y.: Multiobjective Decision Making: Theory and Methodology. North-Holland, Amsterdam (1983)
[7] Bingul, Z., Sekmen, A., Zein-Sabatto, S.: Adaptive genetic algorithms applied to dynamic multiobjective problems. In: Artifiticial Neural Networks Engineering Conference, pp. 273–278 (2000)
[8] Deb, K., Agrawal, S.: Simulated binary crossover for continuous search space. Complex Systems 9, 115–148 (1995)
[9] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Proceedings of the Parallel Problem Solving from Nature VI, pp. 849–858 (2000)
[10] Farina, M., Deb, K., Amato, P.: Dynamic multiobjective optimization problems: test cases, approximations, and applications. IEEE Transactions on Evolutionary Computation 8, 425–442 (2004)
[11] Mehnen, J., Wagner, T., Rudolph, G.: Evolutionary optimization of dynamic multi-objective test functions. In: Proceedings of the Second Italian Workshop on Evolutionary Computation (GSICE2), Siena, Italy (September 2006)
[12] Veldhuizen, D.V., Lamont, G.: On measuring multiobjective evolutionary algorithm performance. In: Proceedings of the Congress on Evolutionary Computation (CEC) 2000, vol. 1, pp. 204–211 (2000)
[13] Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8, 173–195 (2000)
[14] Van Veldhuizen, D.A., Lamont, G.B.: Multiobjective evolutionary algorithm test suites. In: Proceedings of the 1999 ACM Symposium on Applied Computing, pp. 351–357 (1999)

A Memetic Coevolutionary Multi-Objective Differential Evolution Algorithm

Omar Soliman, Lam T. Bui, and Hussein Abbass

The Artificial Life and Adaptive Robotics Laboratory, School of ITEE, University of New South Wales, Canberra, Australia
o.solinman@adfa.edu.au, lam.bui07@gmail.com, h.abbass@adfa.edu.au

In this chapter, we propose a co-evolutionary differential evolution algorithm, which combines ideas from co-evolution and local search to guide the search process towards the Pareto optimal set. Experimental results are carried out on fifteen test problems. The analysis demonstrates the effectiveness of the proposed algorithms and generates solution sets that are highly competitive in terms of convergence, diversity, and distribution.

1 Introduction

Many real-world optimization problems can be modelled as (MOPs). As the objectives are often in conflict with each others, no single solution can optimize all the objectives simultaneously. Since the pioneering work of Schaffer's [1], a significant number of evolutionary algorithms (EAs) has been developed for MOPs [2],[3]. These multi-objective algorithms range from conventional evolutionary algorithms (EAs) such as genetic algorithms (GAs), evolution Strategies (ES), and genetic programming (GP), to a relatively newly-developed paradigms such as differential evolution (DE) [4], artificial immune systems (AIS) [5], ant colony optimization (ACO) [6], particle swarm optimization (PSO) [7], and estimation of distributions algorithms (EDA) [8]. (MOEAs) work with a population of candidate solutions and thus can produce a set of non-dominated solutions to approximate the Pareto optimal set in a single run.

There have been several issues that attracted attention in the literature. The first issue is fitness assignment and diversity maintenance [9, 10]. The second issue, is how to balance diversity and convergence with a single population, since genetic drift may cause the population to lose some representative solutions found during the search due to its finite size. To overcome this shortcoming, an external population (archive) is often used in MOEAs for maintaining those non-dominated solutions found during the search. Efforts have been made to study how to maintain and utilize such an external population [11]. A third issue is the combination of MOEA and local search heuristics to perform local fine-tuning using Memetic Algorithms (MAs) [12]. A several (MOMAs) has been developed over the past decade [3]. MOMAs need to consider how to evaluate the quality of solutions for their local search operators.

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 369–388.
springerlink.com

In this paper, we propose a cooperative co-evolutionary approach which co-evolves the populations of solutions and the search directions using multi-objective differential evolution. Further, we also incorporate local search into the algorithm to fine-tune the non-dominated set. A comparative study was carried out to validate the algorithm. The obtained results show a quick convergence to the Pareto optimal set. Also, the obtained non-dominated solutions from the proposed algorithm are diverse in comparison to those generated by other methods.

In the rest of this chapter is organized as follows: an overview of Multi-objective Evolutionary Algorithms, memetic algorithms and the DE algorithm are presented in Section 2. The proposed algorithms are introduced in Section 3. The experimental study, results, and discussions are presented in Section 4. The last section is devoted to the conclusion and future work.

2 Background

2.1 Multi-Objective Evolutionary Algorithms (MOEAs)

Real-world problems often have multiple conflicting objectives. For example, one may like to have a good quality car, but to also want to spend less money on buying it. The question becomes what is an optimal solution for a multi-objective problem (MOP)? In general, it is defined as follows:

"*A solution to a MOP is Pareto optimal if there exists no other feasible solution which would decrease some criterion without causing a simultaneous increase in at least one other criterion.*" [13].

Using this definition of optimality, we usually find several trade-off solutions (called *Pareto optimal set*, or *Pareto optimal front* (POF) for the plot of the vectors of decision variables corresponding to these solutions) that will be further explained later in this section. In that sense, the search has fundamentally changed from finding a single optimal solution as in single-objective optimization to finding a set of non-dominated solutions. MOEAs are stochastic optimization techniques used to find Pareto optimal solutions for a particular problem [2].

Mathematically, in a k-objective optimization problem, a vector function $\overrightarrow{f}(\overrightarrow{x})$ of k objectives is defined as:

$$\overrightarrow{f}(\overrightarrow{x}) = [f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), ..., f_k(\overrightarrow{x})] \tag{1}$$

in which $\overrightarrow{x}$ is a vector of decision variables in the n-dimensional space $\mathbf{R^n}$; n and k are not necessarily the same. Each individual is assigned a vector $\overrightarrow{x}$ and therefore the corresponding vector $\overrightarrow{f}$. The decision to select an individual is partially dependent on all objectives (as in Eq. 1). MOEAs offer a set of solutions; however, which solution to be selected at the end of the day is decision-maker's dependent.

An individual x_1 is said to dominate x_2 if x_1 is better than x_2 when measured on all objectives. If x_1 does not dominate x_2 and x_2 also does not dominate x_1, they are said to be non-dominated. if we use $\preceq$ between x_1 and x_2 as $x_1 \preceq x_2$ to denote that x_1 dominates x_2 and $\prec$ between two scalars a and b as $a \prec b$ to denote that a is better than b (similarly,

$a \succ b$ to denote that a is worse than b, and $a \not\succ b$ to denote that a is not worse than b), then the domination concept is formally defined as follows.

Definition 1: $x_1 \preceq x_2$ *if the following conditions are held*:

1. $f_j(x_1) \not\succ f_j(x_2), \forall\, j \in [1, 2, ...,k]$
2. $\exists\, j \in [1, 2, ...,k]$ *in which* $f_j(x_1) \prec f_j(x_2)$

In general, if an individual in a population is not dominated by any other individual in the population, it is called a non-dominated individual. All non-dominated individuals in a population form the non-dominated set or the Pareto front (as formally described in *definition 2*).

Definition 2: *A set S is said to be the non-dominated set of a population* P *if the following conditions are held:*

1. $S \subseteq P$
2. $\forall\, s \in S, \nexists\, \mathrm{x} \in \mathrm{P} \mid x \not\preceq s$

When the set P represents the entire search space, the set of non-dominated solutions S is called the *global Pareto optimal set*. If P represents a sub-space, S will be called the *local Pareto optimal set*. There is only one global Pareto optimal set, but there could be multiple local ones. However, in general, we simply refer to the global Pareto optimal set as the Pareto optimal set.

The key steps of a MOEA are as follows: At each iteration, the objective values are calculated for every solution and are then used to determine the dominance relationships within the population in order to select potentially better solutions for the next generation. Following this procedure, the population is expected to converge to the POF. Generally, the MOEA has to overcome two major problems [14]. The first problem is how to get as close as possible to the Pareto optimal front. This is not an easy task when considering that the search needs to progress across multiple dimensions simultaneously. The second is how to keep diversity among the solutions in the obtained set. These two problems become common criteria for most current algorithmic performance comparisons.

To date, many MOEAs have been developed. Generally speaking, they are classified into two broad categories: non-elitism and elitism. With the elitism approach, MOEAs employ an external set (the archive) to store the non-dominated solutions after each generation. This set will then be a part of the next generation. With this method, the best individuals in each generation are always preserved, and this way helps the algorithm

Table 1. The common framework for Evolutionary Multi-objective Optimization Algorithms

Step 1: Initialize a population ***P***
Step 2: (optional):Select elitist solutions from ***P*** to create an external set ***FP*** (For non-elitism algorithms, ***FP*** is empty).
Step 3: Create mating pool from one or both of ***P*** and ***FP***
Step 4: Perform reproduction based on the pool to create the next generation P
Step 5: Possibly combine ***FP*** into ***P***
Step 6: Go to step 2 if the termination condition is not satisfied.

gets closer to the POF. Algorithms such as SPEA2 [15], PDE [16] and NSGA-II [10] are examples of this category. In contrast, the non elitism approach has no concept of elitism when it does selection of individuals for the next generation from the current population [14]. Examples of this category include VEGA [1] and NSGA [2]. Although MOEAs are different from each other, the common steps of these algorithms can be summarized as shown below (Table 1).

2.2 Differential Evolution

A (DE) algorithm is a floating-point encoded evolutionary algorithm for global optimization problems over continuous spaces. In order to generate a new offspring, DE uses one main parent and two supportive parents [17, 18, 4]. Basically, the main parent is disturbed by adding a fixed step length multiplied by the difference between the two supportive parents. The resultant solution is called the trial/prototype solution. The prototype is then crossed-over with another pre-selected solution to generate an offspring. The offspring is inserted into the population if it outperforms the pre-selected solution. By using difference vectors, DE takes into account direction information. In some cases, good directions will be generated and DE will generate good solutions. In other cases, bad directions will be generated which will deteriorate the solution quality. There are several variants of the DE algorithm [17, 18] *DE/rand/1*, *DE/Best/1*, *DE/rand-to-Best/1* and *DE/Best/2* all variants are the same except in the way to generate mutant vectors. In this paper, we focus on the *DE/rand/1* variant of DE, explained as a pseudo code in Figure 1. It is also used as a basis for comparison with the proposed strategy. According to Storn [17, 18] the DE algorithm has three operators: mutation, crossover and selection. Mutation and crossover are used to generate the new trial solution, while the selection operator determines which of the vectors will insert into the population and survive to the next generation.

- *Mutation Operator:*
  For each target solution vector $(X_{i,t})$, $i = 1,2,...,NP$, a mutant vector $(V_{i,t+1})$ is generated based on one of the following equation:
  - DE/rand/1:
  $$V_{i,t+1} = X_{r1,t} + F(X_{r2,t} - X_{r3,t}), \tag{2}$$
  - DE/Best/1:
  $$V_{i,t+1} = X_{Best,t} + F(X_{r1,t} - X_{r2,t}), \tag{3}$$
  - DE/rand to Best/1:
  $$V_{i,t+1} = X_{i,t} + F(X_{Best,t} - X_{i,t}) + F(X_{r1,t} - X_{r2,t}), \tag{4}$$
  - DE/Best/2:
  $$V_{i,t+1} = X_{Best,t} + F(X_{r1,t} - X_{r2,t}) + F(X_{r3,t} - X_{r4,t}), \tag{5}$$

  Where: $r1, r2, r3, r4 \in (1...NP)$are three mutually distinct random number and distinct from i. NP is the number of solution vectors and F is a real number representing the step length, $F \in (0,2)$ controls the amplification of the difference vectors. $X_{Best,t}$ is the best solution vector at generation t.

- *Crossover Operator:*
  The mutant vector($V_{i,t+1}$) and the target solution vector ($X_{i,t+1}$ are crossed over to generate a trial solution vector ($U_{i,t+1}$ according to the following equation:

$$U_{i,t+1} = (u_{1i,t+1}, u_{2i,t+1}, ..., u_{Di,t+1}) \tag{6}$$

  where :

$$u_{ji,t+1} = \begin{cases} v_{ji,t+1}, & \text{if } r_j \leq CR \text{ or } j = rn(i) \\ x_{ji,t}, & \text{if } r_j > CR \text{ and } j \neq rn(i). \end{cases} \tag{7}$$

  Where: $j = 1, 2, ..., D$, with D as the dimension of the solution vector, r_j is a uniform random number generated from the interval $\in [0, 1]$, CR is a crossover probability, $rn(i)$ is a randomly chosen index $\in (1, 2, ...D)$ which ensures that $U_{i,t+1}$ gets at least one parameter from ($V_{i,t+1}$).
- *Selection Operator:*
  The trial vector $U_{i,t+1}$ is compared to the target vector ($X_{i,t}$) using the following criterion:

$$X_{i,t+1} = \begin{cases} U_{i,t+1}, & iff(U_{i,t+1}) < f(X_{i,t}) \\ X_{i,t}, & \text{otherwise} \end{cases} \tag{8}$$

***Input**: population P, crossover rate CR, a real value F*
***Evolve** P:*
For i=1 to the population size(NP)
Select randomly three different values r1,r2,r3:
r1 ≠ r2 ≠ r3 ≠ i
For each variable j
if Random ≤ CR
X = P[r1][j] + F(P[r2][j]-P[r3][j])*
else X = P[i[j]
If X is better than P[i], replace P[i] by X
Loop
***Output**: new population P*

Fig. 1. The pseudo code for the DE/rand/1

To date, many MOEAs have been introduced in the literature using DE such as PDE [16], GDE [19], and NSDE [20].

2.3 Memetic Algorithms

[12] are population-based heuristic search approaches for optimization problems based on cultural evolution. The majority of memetic approaches are based on finding local improvements of candidate solutions obtained by the evolutionary search mechanism. This is performed by using dedicated local search methods such as Back-propagation when training artificial neural networks [21].

There has been a significant number of (MOMAs). Here, we summarize some of the most popular ones. In [22], Ishibuchi and Murata proposed to assign fitness for

individuals using a randomly selected linear utility function. The resulting offspring is improved by a local search operator using the same utility function by which the parents were selected. The local search procedure is terminated when k neighbors of the current solution have been examined with no improvement. An elitist strategy is incorporated in the procedure. Also, in [23], the authors introduced a local search probability to MOMAs [22] for decreasing the computation time spent by local search. In their modified MOMA, local search is not applied to all solutions in the current population, but to selected solutions with a predefined probability.

In [24] two novel algorithms were proposed. The first was based on a hybrid of EAs with simulated annealing, while the second was quite similar to the one proposed by Ishibuchi and Murata [22]. The difference is that they introduced a variant of restricted mating so that only the N best solutions measured using a random utility function are allowed to mate. The two proposed algorithms were tested and compared with Ishibuchi and Murata's on a set of multi-objective travelling salesman problems. On these problems, global convexity may be exploited and so restricted mating is advantageous.

Knowles and Corne [25] combined their Pareto archived evolution strategy (PAES [26] with a crossover operation for designing a memetic PAES (M-PAES). In their M-PAES, the Pareto-dominance relation and the grid-type partitioning of the objective space were used for determining the acceptance (or rejection) of new solutions generated in genetic and local search. The performance of M-PAES was examined for multi-objective knapsack problems and for degree-constrained multi-objective MST (minimum-weight spanning tree) problems. In those studies, the M-PAES was compared with the PAES, the MOGLS of Jaszkiewicz [24], and a MOEA.

More recently in [27] a new memetic algorithm for multi-objective optimization was proposed, which combines the global search ability of particle swarm optimization with a synchronous local search heuristic for directed local fine-tuning. A new particle updating strategy is proposed based upon the concept of fuzzy global-best to deal with the problem of premature convergence and diversity maintenance within the swarm. The proposed features are examined to show their individual and collective behaviors in MOEA optimization.

3 Methodology

As indicated in Section 1.2.2, the use of direction in guiding the search engine in DE is a vital component. However, in some cases, good directions will be generated and therefore DE will generate good solutions. In other cases, bad directions will be generated which will deteriorate the solution quality. This poses a question of whether or not we can control as well as maintain good directions? Our proposal is to employ *co-evolution* to do the task where two populations are co-evolved during the optimization process. One is for the population of the solutions, while the other is for the population of directions. We introduce two approaches to control the interaction between these two populations: one serves as a main engine for the co-evolution and search, while the other uses *local-search* for further improvement of the solutions found from the former one. We employ the non-dominated sorting DE algorithm (called NSDE) of [20] as a

base for the implementation of our approaches. NSDE is a version of NSGA-II using DE operators instead of its original crossover and mutation operators.

3.1 Co-evolutionary Multi-Objective Differential Evolution - CMODE

Here, we describe our (CMODE) - allowing evolution of directions side-by-side with the population of solutions. This is considered as where the direction is used to make improvement for the population of solutions. The fitness of a direction is based on whether it helps to make improvement or not. The detailed steps of the algorithm are presented as follow:

1. **Initialize**
   - Randomly initialize the population of solutions P using an uniform distribution.
   - Evaluate all individuals of P and then apply non-dominated sorting to P.
   - Generate a population of directions D by iterating the following:
     - Choose an individual i from the top 20% of P.
     - Choose an individual j from the lowest 80% of P.
     - Generate a direction d: $d = U(0,1) * (i - j)$, in which $U(0,1)$ is an uniform distributed random value.
     - Change each variable of the direction d to a zero value with probability 0.5
     - Add the newly generated direction d to D.
2. **Generate** population of offspring $P1$ of the same size as P by iterating the process of selecting an individual from the top 20% of P and add to it a randomly selected direction from the direction population D.
3. **Evaluate** all solutions in $P1$ against objective functions. Also, assign fitness values for individuals in D. The fitness value of a selected direction in D is determined based on the improvement of the newly generated child. If the child dominates its parent, the fitness of the direction is increased by one; and if it is dominated by its parent, the fitness of the direction is decreased by one; and otherwise, no changes of the fitness are made.
4. **Combine** P and P1 in order to form a new population P2 of the same size as P by selecting the best of P + P1 based on non-dominated sorting.
5. **Replace** P with P2.
6. **Perform** non-dominated sorting for P
7. **Perform** sorting for population D based on individual fitness
8. **Replace** the worst 25% directions in the direction population D with a newly generated ones as in Step 1.
9. **Check stopping-conditions**: Stop process if termination conditions are satisfied. Otherwise, go to *Step 2*.

3.2 Memetic Implementation (CMODE-MEM)

The implementation of a memetic procedure within CMODE is straightforward. After each generation, a predefined portion of solutions are randomly selected to undergo local search. In general, it is described as follows:

1. **Define** the number of steps using local search and the neighborhood of the solutions
2. **Select** a solution S from the population
3. **Perform** local search
   - Perturb S to get a new solution $S1$ within its neighborhood
   - Evaluate $S1$
   - If $S1$ dominates S, replace S by $S1$
   - Repeat this process for a number of predefined steps
4. **Goto** Step 1 until the number of selected solutions is reached.

3.3 Computational Complexity

Computational complexity of an algorithm is usually seen as the worst-case computational time that the algorithm spends on solving a problem instance of an input size m [28]; and it is usually presented as $O(.)$. When $f(m)$ and $g(m)$ are two functions of the input size m. $f(m)$ is $O(g(m))$ if there exists a constant c such that $|f(m)| \leq c|g(m)|$. If $g(m)$ is a polynomial function of the input size, the algorithm is a polynomial-time algorithm, otherwise an exponential-time one.

In general, for the evolution of the population of solutions, the complexity is still as that of NSGA-II $O(MN^2)$ in which M is the number of objectives and N is population size (Sept 6 in our algorithm is also required in NSGA-II). For the evolution of direction, it requires sorting of the population based on fitness of individuals. This process requires (assume using quick-sort) $O(NlogN)$ on average; but on the worst case, it still requires $O(N^2)$. So, overall complexity of the framework is still $O(MN^2)$.

4 Experimental Studies

In order to validate the proposed approach, we carried out a comparative study on a set of test problems. Algorithms taking part in this studies are:

- **NSDE**
- **NSGA-II**
- **CMOED**
- **CMOED-MEM**

4.1 Test Problems

A summary of a collection of 15 test problems that is used in this paper is given in Table 2. This list is divided into two sets:

- The first set includes 7 low-dimension problems (from BINH to REN2) that has 2D decision search space and a property of uni-modality.
- The second set (from KUR to ZDT6) contains more difficult problems with larger search space. It has either uni-modal or multi-modal problems.

Table 2. Lists of test problems used for experiments in this paper

Problems	Dim	Range	POF features
BINH	2	$x_i \in [-5,10]$	Convex
		$f_1(x) = x_1^2 + x_2^2$	Uniform
		$f_2(x) = (x_1-5)^2 + (x_2-5)^2$	Uni-modal
			Connected
POL	2	$x_i \in [-\pi,\pi]$	Nonconvex
		$f_1(x) = [1 + (A1 - B1)^2 + (A2 - B2)^2]$	Disconnected
		$f_2(x) = [(x_1+3)^2 + (x_2+1)^2]$	Uniform
		$A1 = 0.5sin1.0 - 2cos1.0 + sin2.0 - 1.5sin2.0$	
		$A2 = 1.5sin1.0 - cos1.0 + 2sin2.0 - 0.5sin2.0$	
		$B1 = 0.5sinx_1 - 2cosx_1 + sinx_2 - 1.5sinx_2$	
		$B2 = 1.5sinx_1 - cosx_1 + 2sinx_2 - 0.5sinx_2$	
LAU	2	$x_i \in [-50,50]$	Convex
		$f_1(x) = x_1^2 + x_2^2$	Disconnected
		$f_2(x) = (x_1+2)^2 + x_2^2$	Uni-modal
			Uniform
LIS	2	$x_i \in [-5,10]$	Nonconvex
		$f_1(x) = \sqrt[8]{x_1^2 + x_2^2}$	Uniform
		$f_2(x) = \sqrt[4]{(x_1-0.5)^2 + (x_2-0.5)^2}$	Uni-modal
			Disconnected
MUR	2	$x_1 \in [1,4]$ and $x_2 \in [1,2]$	Nonconvex
		$f_1(x) = 2\sqrt{x_1}$	Uni-modal
		$f_2(x) = x_1(1-x_2) + 5$	Connected
			Uniform
REN1	2	$x_i \in [-3,3]$	Convex
		$f_1(x) = \frac{1}{x_1^2 + x_2^2 + 1}$	Uniform
		$f_2(x) = \frac{1}{x_1^2 + 3x_2^2 + 1}$	Uni-modal
			Connected
REN2	2	$x_i \in [-3,3]$	Convex
		$f_1(x) = x_1 + x_2 + 1$	Uniform
		$f_2(x) = x_1^2 + 2x_2 - 1$	Uni-modal
			Connected
KUR	3	$x_i \in [-5,5]$	Nonconvex
			Disconnected
		$f_1(x) = \sum_{i=1}^{n-1}(-10exp(-0.2\sqrt{x_i^2 + x_{i+1}^2}))$	Uniform
		$f_2(x) = \sum_{i=1}^{n}(\lvert x_i\rvert^{0.8} + 5sin(x_i)^3)$	
FON	10	$x_i \in [-4,4]$	Nonconvex
			Connected
		$f_1(x) = 1 - exp(-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{n}})^2)$	Uniform
		$f_2(x) = 1 - exp(-\sum_{i=1}^{n}(x_i + \frac{1}{\sqrt{n}})^2)$	Uni-modal
QUA	16	$x_i \in [-5.12,5.12]$	Non-convex
		$f_1(x) = \sqrt{\frac{A_1}{n}}$	Connected
		$f_2(x) = \sqrt{\frac{A_2}{n}}$	Uniform
			Multi-modal
ZDT1	30	$x_i \in [0,1]$	Convex
		$f_1(x) = x_1$, $f2(x) = g * h$	Connected
		$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	Uniform
		$h(f_1,g) = 1 - \sqrt{\frac{f_1}{g}}$	Uni-modal
ZDT2	30	$x_i \in [0,1]$	Nonconvex
		$f_1(x) = x_1$, $f2 = g * h$	Connected
		$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	Uniform
		$h(f_1,g) = 1 - (\frac{f_1}{g})^2$	Uni-modal
ZDT3	30	$x_i \in [0,1]$	Convex
		$f_1(x) = x_1$, $f2 = g * h$	Disconnected
		$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$	Uniform
		$h(f_1,g) = 1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g}sin(10\pi f1)$	Uni-modal
ZDT4	10	$x_1 \in [0,1]$	Convex
		$x_i \in [-5,5]$, i=2..10	Connected
		$f_1(x) = x_1$, $f2 = g * h$	Uniform
		$g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10cos(4\pi x_i))$	Multi-modal
		$h(f_1,g) = 1 - (\frac{f_1}{g})^2$	
ZDT6	10	$x_i \in [0,1]$	Nonconvex
		$f_1(x) = 1 - exp(-4x_1)sin^6(6\pi x_1)$, $f2 = g * h$	Connected
		$g(x) = 1 + 9[\frac{\sum_{i=2}^{10} x_i}{9}]^{0.25}$	Non-uniform
		$h(f_1,g) = 1 - (\frac{f_1}{g})^2$	Uni-modal

4.2 System Settings

For all experiments, the total population size was set as 200. Further, for DE, the crossover rate was set as 0.7 and the step length was randomly generated between 0 and 1 similar to [16]. Local search was triggered after every generations with 2% of populations for easy problems, and 5% for hard ones. All cases were tested in 30 separate runs using 30 different random seeds. The results have been analyzed within these 30 runs for each model and on each problem.

In all experiments, for NSGA-II, crossover rate was 0.9 and mutation rate was 0.1. The distribution indexes for crossover and mutation operators were $\eta_m = 20$ and $\eta_c = 15$ as recommended by its authors. Further, all algorithms ran with the same number of evaluations in order to make a fair comparison.

4.3 Performance Measurement Methods

Performance metrics are usually used to compare algorithms in order to form an understanding of which one is better and in what aspects. However, it is hard to define a concise definition of algorithmic performance. In general, when doing comparisons, a number of criteria are employed [14]. We will look at two of these criteria. The first measure is the generation distance, *GD*, which is the average distance from the set of solutions found by evolution to the POF [29]

$$GD = \frac{\sqrt{\sum_{i=1}^{N} d_i^2}}{N} \tag{9}$$

where d_i is the Euclidean distance (in objective space) from solution i to the nearest solution in the POF. If there is a large fluctuation in the distance values, it is also necessary to calculate the variance of the metric. Finally, the objective values should be normalized before calculating the distance.

As recommended in [29], it is considered better to use the , *HR*, that is measured by the ratio between the hyper–volumes of hyper–areas covered by the obtained POF and the true POF, called H_1 and H_2 respectively. *HR* is calculated as in Eq. 10. For this metric, the greater the value of *HR*, the better convergence the algorithm has.

$$HR = \frac{H1}{H2} \tag{10}$$

There are some discussions on how to determine the reference point for the calculation of the hyper-volume, for example, it can be the origin [29]. However, generally it is dependent on the area of the objective space that is visited by all comparing algorithms. In this paper, as suggested elsewhere [2], the reference point is the one associated with all the worst values of objectives found by all the algorithms under investigation.

4.4 Results and Discussions on the Effect of Evolving Directions

4.4.1 Performance Analysis

Convergence is one of the most important characteristics of an optimization technique. However, the ways of looking at convergence of single objective and multi-objective

Table 3. The predefined level of the hyper-volume ratio that algorithms need to reach within a time limit. They are kept as high as possible, but majority of approaches can achieve.

Problems	Predefined level
BINH	0.999
LAU	0.999
LIS	0.999
MUR	0.990
POL	0.995
REN1	0.995
REN2	0.999
KUR	0.940
FON	0.800
QUA	0.900
ZDT1	0.999
ZDT2	0.950
ZDT3	0.999
ZDT4	0.999
ZDT6	0.999

Table 4. The number of evaluations that algorithms needed to reach the certain levels of hyper-volume ratio (mean and standard deviation from 30 runs). NA means the algorithms could not reach the levels within an evaluation limit.

	NSGA-II		NSDE		CMODE	
Problems	Mean	Std	Mean	Std	Mean	Std
BINH	1833	320†	1387	104†	**1280**	**186**
LAU	1047	286†	840	259†	**633**	**140**
LIS	6300	2678	4513	933	4013	1143
MUR	1800	197†	1007	98†	**627**	**69**
POL	693	208†	493	101†	**427**	**69**
REN1	993	98 †	667	109†	**607**	**64**
REN2	2747	484†	1247	114†	**767**	**92**
KUR	1473	249†	2100	389†	**1247**	**215**
FON	6073	326†	3413	117†	**1840**	**161**
QUA	**62107**	**26139**†	NA	NA†	75053	22411
ZDT1	21080	527†	18487	542†	**11807**	**1157**
ZDT2	5913	422†	8380	308†	**4333**	**415**
ZDT3	23407	1110†	25853	948†	**15480**	**1603**
ZDT4	**18633**	**1889**†	78300	5560†	NA	NA
ZDT6	38807	1099†	10633	1717†	**3167**	**417**

optimizations are different [30]. If some measurements of the objective function, with regard to the number of generations, are experimentally considered as an indication for convergence in single objective optimization, it is not a suitable method for multi-objective optimizations since they do not involve the mutual assessment on all objective functions.

Table 5. The average GD and HR recorded after 20000 evaluations for all algorithms. Symbol † indicates that the difference between algorithms and CMODE is significant.

	GD			HR		
Problems	NSGA-II	NSDE	CMODE	NSGA-II	NSDE	CMODE
BINH	0.007±0.000	0.006±0.000†	0.007±0.000	1.000±0.000	1.000±0.000	1.000±0.000
LAU	0.089±0.002†	0.089±0.002†	0.091±0.002	1.000±0.000	1.000±0.000	1.000±0.000
LIS	0.002±0.000†	0.001±0.000†	0.001±0.000	1.077±0.016	1.065±0.010	1.063±0.015
MUR	0.000±0.000	0.000±0.000	0.000±0.000	0.999±0.000	0.999±0.000	0.999±0.000
POL	0.002±0.005	0.001±0.000	0.002±0.004	1.000±0.000	1.000±0.000	1.000±0.000
REN1	0.001±0.000	0.001±0.000	0.001±0.000	0.998±0.000	0.999±0.000†	0.998±0.000
REN2	0.001±0.000	0.001±0.000	0.001±0.000	1.000±0.000	1.000±0.000	1.000±0.000
KUR	0.001±0.000	0.001±0.000	0.001±0.000	1.000±0.000	0.999±0.001†	1.000±0.000
FON	0.001±0.000†	0.000±0.000	0.000±0.000	0.962±0.004†	0.994±0.000†	0.992±0.000
QUA	0.001±0.000	0.000±0.000†	0.001±0.001	0.872±0.014†	0.799±0.015†	0.859±0.014
ZDT1	0.001±0.000†	0.001±0.000†	0.000±0.000	0.999±0.000†	0.999±0.000†	1.000±0.000
ZDT2	0.001±0.000†	0.001±0.000†	0.000±0.000	0.998±0.000†	0.999±0.000†	1.000±0.000
ZDT3	0.001±0.000†	0.001±0.000†	0.000±0.000	0.998±0.000†	0.997±0.000†	1.000±0.000
ZDT4	0.063±0.045†	5.784±1.492†	0.825±3.274	1.000±0.001†	0.962±0.007†	0.986±0.012
ZDT6	0.022±0.004	0.001±0.000	0.001±0.000	0.973±0.003†	1.001±0.000	1.001±0.000

Further, convergence is not only related to how close the obtained POF after a period of time is in comparison to the true Pareto optimal front, but also the rate of convergence which can be represented by convergence over time. We consider both issues in this section. For the closeness of the obtained POF, we use the generation distance GD (as well as the hyper–volume ratio HR). While GD is an indication for the closeness of an obtained set of solution to the true POF, HR is for both closeness as well as diversity of the obtained set.

Firstly, for the convergence rate, the number of evaluations that each approach spends to reach a certain high level of hyper-volume ratio (Table 3) was recorded at each run. The mean and standard deviation of the 30 different runs are calculated and reported. These were derived from the highest level of *HR* that almost all algorithms could achieve within an evaluation limit. This measurement is to provide a quantitative indication of how fast the algorithms are in converging to the true POF (by using the same high level of *HR*). These values are reported in Table 4. *t-test* with 0.05 level of significance was used to test the difference between results of the comparing approaches.

The performance comparison is divided into two categories: Comparing between NSGA-II and DE approaches (NSDE and CMODE) and between DE approaches themselves. It is clear from the table that DE approaches were quicker than NSGA-II in almost all problems (except QUA and ZDT4, two multi-modal problems). This indicates that the use of direction is obviously helps the algorithms to move quickly to the area of the POF. This is consistent with the finding in [20].

Regarding the performance between NSDE and CMODE, we can see that the improvement was made by the use of co-evolution where CMODE was statically-significantly faster than NSDE in most of problems. These results clearly support our

argument that the evolution of directions helps CMODE finds and maintains the good directions. However, it is interesting to see that for QUA and ZDT4 (two multi-modal problems), the DE approaches was worse than NSGA-II. This means that quick convergence of DE approaches might make them lose diversity and hardly converge to the global POF. This might be the case also when using local search. This will be analyzed further in the next section.

Secondly, in order to confirm the convergence of the algorithms, we recorded the generation distance GD and hyper–volume ratio HR after 20 000 evaluations for each algorithm. They are all reported in Table 5. It is quite clear that after 20 000 evaluations, all algorithms converged to the true POF for all problems except ZDT4 which converged with very small GD and high HR. For ZDT4, NSDE seemed to be the worst with very large value of GD. The multi-modality made it hard to converge to the global POF. Note that, for ZDT4, although CMODE was better than NSDE, it was still worse than NSGA-II.

In general, CMODE shows a great deal of employing the evolution of directions during the optimization process. However, in some cases of multi-modality, this advantage seems to be deteriorated since the exploration might be less focussed and the balance between exploration and exploitation might be violated. This opens an opportunity for using memetic methods for CMODE. This will be analyzed in the next section.

4.4.2 Dynamics of the Co-evolutionary Approach

We now move to an analysis on the dynamics of the proposed approach during the optimization process. We measure the diversification of the direction population over time (up to 100 generations), which is measured by the average Euclidean distances

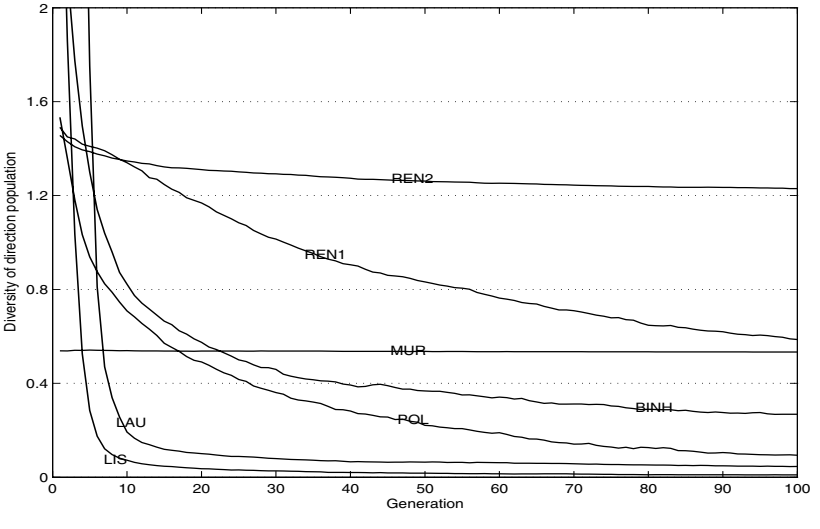


Fig. 2. Diversity of direction population for first seven test problems(2D)

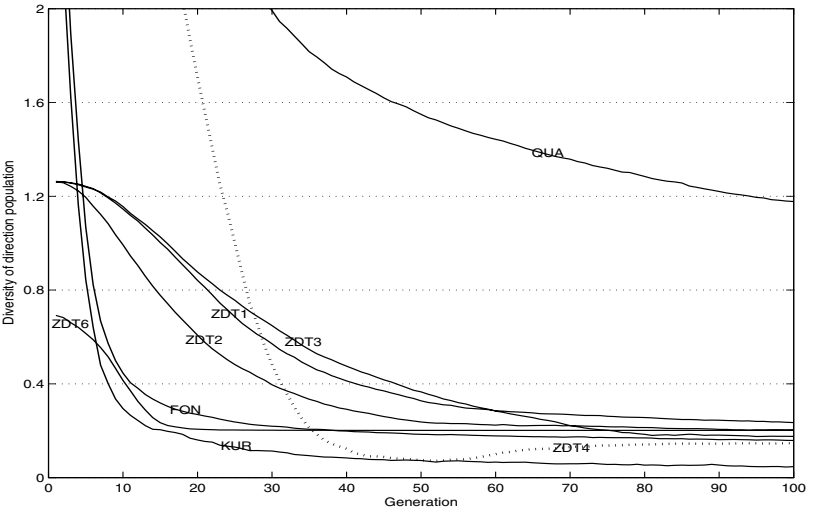


Fig. 3. Diversity of direction population for the last eight test problems(high-dimension Problems)

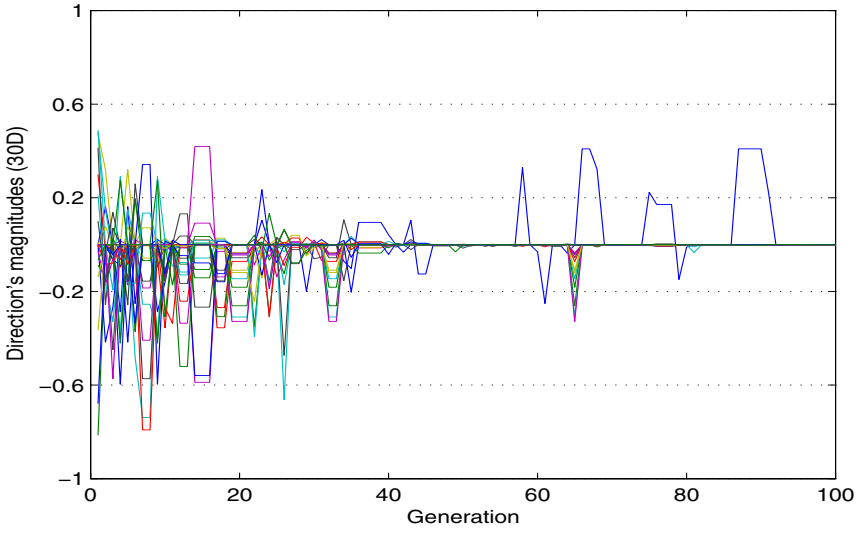


Fig. 4. The best direction for ZDT1 problem (one run) obtained over time. Each curve (among 30 curves) is for one single variable).

among individuals in a population, in order to see how the improvement of the direction is made over time. For this, the average diversity over 30 runs was plotted in Figures 2 (for the first set of 2D problems) and 3 (for the second set of difficult problems).

It is clear that the diversity of the direction population reduced over time as the CMODE was converging to the POF. That is understandable since the population focused on several good directions over time. We use ZDT4 as an example where the diversity of the direction population dropped as the algorithm progressed. However, after about 50 generations, its diversity increased again. This indicates that the population of directions converged to some good directions for a local POF. However, after a while, it started to realize the situation and tried to introduce the new direction to overcome the local POF. Therefore, the diversity increased. Once more, it motivates the use of local search for CMODE for multi-modal problems.

We plot the best direction obtained over time for ZDT1 and ZDT4 in Figures 4 and 5. Again, we see the trend that the best direction changed its position frequently at the start of the process, but over time it stayed more stable. This means that early in the process, the algorithm focused on exploring the space, then it focused more on exploitation. Later on, there were few changes of the best direction, but these changes are much less than earlier in evolution.

4.5 Results and Discussions on the Effect of Memetic

4.5.1 Performance Analysis

In this section, we investigate how the memetic procedure helps CMODE to improve its performance. Again, we measure the convergence rate of CMODE-MEM in order to compare with CMODE. All results are reported in Table 6. It is very clear that for problems that CMODE was already good in converging to the POF, there was no improvement in CMODE-MEM. This means that when CMODE maintains well the balance between exploration and exploitation, local search does not make more sense in

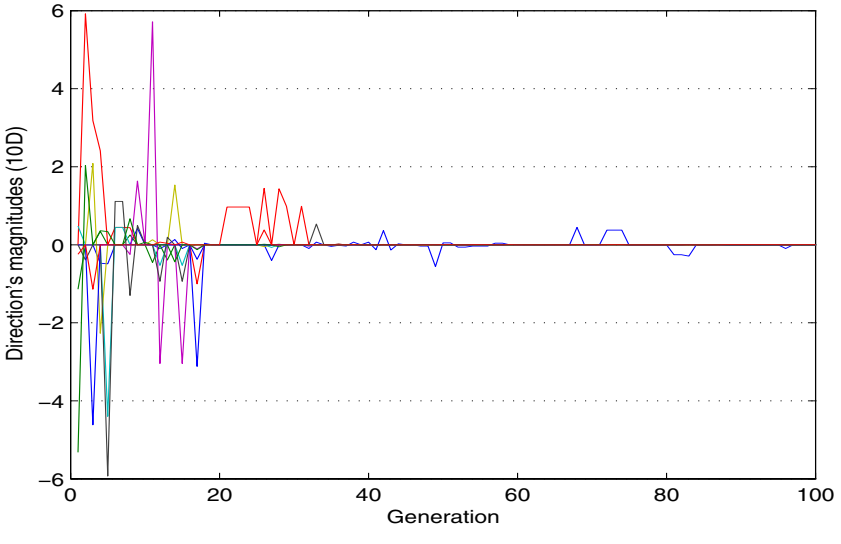


Fig. 5. The best direction for ZDT4 problem (one run) obtained over time. Each curve (among 10 curves)is for one single variable).

terms of exploration. This leads to a question of how CMODE-MEM behaves in some problems (such as QUA and ZDT4) where the balance seems not to be well-maintained?

From the results on two problems QUA and ZDT4, we can see that there was a significant improvement of CMODE when using memetic method. Local search certainly helped CMODE to quickly overcome the traps caused by local POFs. This finding indicates that local search should be used for CMODE in solving multi-modal problems.

Further supporting evidences for the performance of CMODE-MEM is given in Table 7 where all GD and HR, recorded after 20 000 evaluations, are given. Once again, we see the good performance of CMODE and CMODE-MEM over the test problems. They all obtained very small GD and large HR (near 1.0). For QUA and ZDT4, local search helped CMODE to find the global POF and cover all parts of the POF.

4.5.2 Success Rates of Local Search in CMODE-MEM

We investigate the dynamics of CMODE-MEM via the success rates of local search during the optimization process which is the percentage that the local search makes successful steps or an improvement. The results are plotted in Figures 6 and 7.

As shown in Table 6, for the first set of problems, both approaches converged after about five generations (1000 evaluations); illustrating why the success rates of CMODE-MEM dropped to zero after five generations. Obviously, after converging, the use of local search is not useful any more. Therefore, when the success of local search vanishes, we can use this as a criterion to stop using local search.

Table 6. The number of evaluations that CMODE with/without memetic needed to reach the certain levels of hyper-volume ratio (mean and standard deviation from 30 runs). NA means the algorithms could not reach the levels within an evaluation limit. Symbol † indicates that the difference between algorithms.

	CMODE		CMODE-MEM	
Problems	Mean	Std	Mean	Std
BINH	1280	186	1213	173
LAU	633	140	659	155
LIS	4013	1143†	3363	1038
MUR	627	69	645	63
POL	427	69†	444	72
REN1	607	64	624	55
REN2	767	92	742	118
KUR	1247	215	1262	250
FON	1840	161	1872	181
QUA	75053	22411	**58372**	**43792**
ZDT1	11807	1157	12036	1542
ZDT2	4333	415†	4853	351
ZDT3	15480	1603	15510	1680
ZDT4	NA	NA†	**31992**	**41823**
ZDT6	3167	417†	3390	464

Table 7. The average GD and HR (the mean and standard deviation) recorded after 20 000 evaluations. Symbol † indicates that the difference between algorithms.

	GD		HR	
Problems	CMODE	CMODE-MEM	CMODE	CMODE-MEM
BINH	0.007±0.000	0.007±0.000	1.000±0.000	1.000±0.000
LAU	0.091±0.002	0.091±0.002	1.000±0.000	1.000±0.000
LIS	0.001±0.000	0.001±0.000	1.063±0.015	1.065±0.019
MUR	0.000±0.000	0.000±0.000	0.999±0.000	0.999±0.000
POL	0.002±0.004	0.001±0.000	1.000±0.000	1.000±0.000
REN1	0.001±0.000	0.001±0.000	0.998±0.000	0.998±0.000
REN2	0.001±0.000	0.001±0.000	1.000±0.000	1.000±0.000
KUR	0.001±0.000	0.001±0.000	1.000±0.000	1.000±0.000
FON	0.000±0.000	0.000±0.000	0.992±0.000	0.992±0.001
QUA	0.001±0.001	0.001±0.000	0.859±0.014	0.865±0.014
ZDT1	0.000±0.000	0.000±0.000	1.000±0.000	1.000±0.000
ZDT2	0.000±0.000	0.000±0.000	1.000±0.000	1.000±0.000
ZDT3	0.000±0.000	0.000±0.000	1.000±0.000	1.000±0.000
ZDT4	0.825±3.274	0.101±0.114	0.986±0.012†	0.998±0.002
ZDT6	0.001±0.000	0.001±0.000	1.001±0.000	1.001±0.000

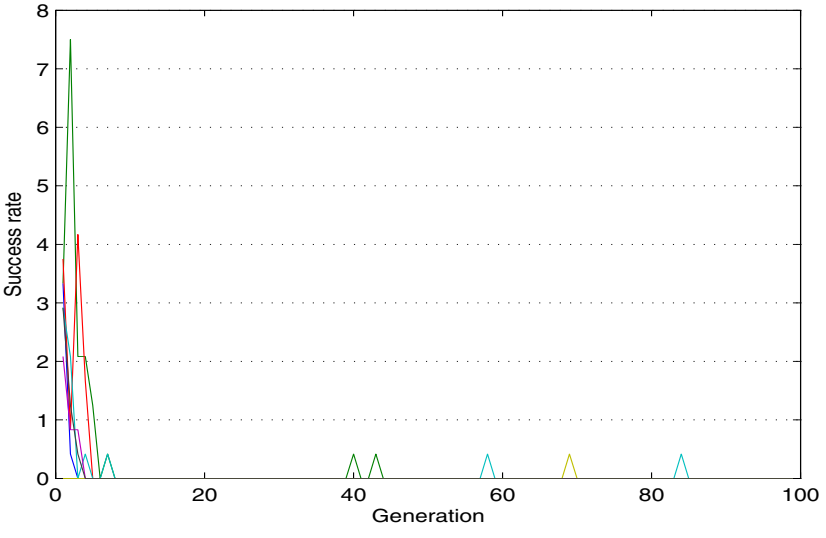


Fig. 6. The success rate of CMODE-MEM algorithm for the 2D test problems

However, for the second set of problems, since the algorithm took more time to converge, the success rates were slowly reduced. For ZDT1, ZDT2, and ZDT3, the search space are very large (30D). Therefore, the algorithm takes longer time to cover up all parts of the POF.

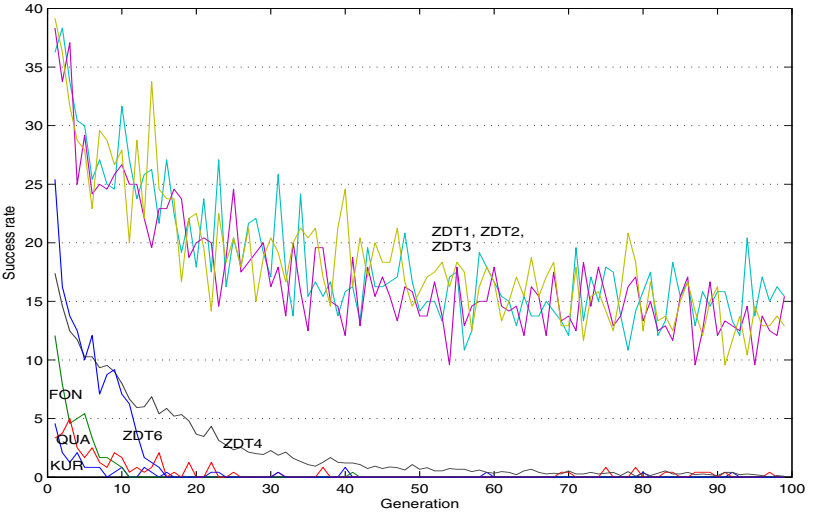


Fig. 7. The success rate of CMODE-MEM algorithm for the high dimension test problems (last eight problems)

5 Conclusion

In this chapter, we proposed an approach to incorporate co-evolution and local search into DE in order to improve the performance of DE in solving multi-objective problems. For the co-evolutionary mechanism, a population of direction is co-evolved with the population of solutions. The improvement in fine-tuning the obtained non-dominated solutions is made by using local search during the optimization process. Experimental results were carried out on wide range of test problems. The comparative study among CMODE, CMODE-MEM, NSDE, and NSGA-II showed the effectiveness of the proposed approach in obtaining diverse non-dominated solutions. For future work, we intend to employ a self-adaptation strategy for local search to adapt and control the local improvements based on historical knowledge.

Acknowledgements

This work is supported by the Australian Research Council (ARC) Centre for Complex Systems grant number CEO0348249.

References

1. Schaffer, J.: Multiple objective optimization with vector evaluated genetic algorithms. In: Proceedings of the First International Conference on Genetic Algorithms, Hillsdale, New Jersey, pp. 93–100 (1985)

2. Deb, K.: Multiobjective Optimization using Evolutionary Algorithms. John Wiley and Son Ltd., New York (2001)
3. Knowles, J., Corne, D.: Approximating the nondominated front using the pareto archived evolution strategy. Evolutionary Computation 8(2), 149–172 (2000)
4. Price, K., Storn, R., Lampinen, J.: Differential Evolution - A Practical Approach to Global Optimization. Springer, Berlin (2005)
5. Dasgupta, D.: Artificial Immune Systems and Their Applications. Springer, Berlin (1998)
6. Dorigo, M., Stutzle, T.: Ant Colony Optimization. MIT Press, USA (2004)
7. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks 4, 1942–1948 (1995)
8. Larraanaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Norwell, MA, USA. Kluwer Academic Publishers, Dordrecht (2002)
9. Srinivas, N., Deb, K.: Multi-objective function optimization using non-dominated sorting genetic algorithm. Evolutionary Computation 2(3), 221–248 (1994)
10. 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)
11. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation 3(4), 257–271 (1999)
12. Hart, W., Krasnogor, N., Smith, J. (eds.): Recent Advances in Memetic Algorithms. Studies in Fuzziness and Soft Computing, vol. 166. Springer, Heidelberg (2005)
13. Coello, C.A.C.: Evolutionary multi-objective optimization: A historical view of the field. IEEE Computational Intelligence Magazine 1(1), 28–36 (2006)
14. Zitzler, E., Thiele, L., Deb, K.: Comparision of multiobjective evolutionary algorithms: Emprical results. Evolutionary Computation 8(1), 173–195 (2000)
15. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm. tech. rep., ETH in Zurich, Swiss (2001)
16. Abbass, H.A., Sarker, R., Newton, C.: PDE: A Pareto frontier differential evolution approach for multiobjective optimization problems. In: Proceedings of CEC 2001, vol. 2, pp. 971–978. IEEE Press, Los Alamitos (2001)
17. Storn, R., Price, K.: Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. technical report tr-95-012. tech. rep., ICSI (1995)
18. Corne, D., Dorigo, M., Glover, F.: New Ideas in OPtimization. McGraw Hill, Cambridge (1999)
19. Kukkonen, S., Lampinen, J.: An extension of generalized differential evolution for multiobjective optimization with constraints. 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. 752–761. Springer, Heidelberg (2004)
20. Iorio, A.W., Li, X.: Solving rotated multi-objective optimization problems using differential evolution. In: Webb, G.I., Yu, X. (eds.) AI 2004. LNCS (LNAI), vol. 3339, pp. 861–872. Springer, Heidelberg (2004)
21. Abbass, H.: A memetic pareto evolutionary approach to artificial neural networks. In: Stumptner, M., Corbett, D.R., Brooks, M. (eds.) Canadian AI 2001. LNCS (LNAI), vol. 2256, pp. 1–12. Springer, Heidelberg (2001)
22. Ishibuchi, H., Murata, T.: Multi-objective genetic local search algorithm. In: Proceedings of the Conference on Evolutionary Computation, pp. 119–124. IEEE Press, Los Alamitos (1996)
23. 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)

24. Jaszkiewicz, A.: Genetic local search for multi-objective combinatorial optimization. European Journal on Operation Research 137(1), 50–71 (2002)
25. Knowles, J.D., Corne, D.: m-paes: A memetic algorithm for multi-objective optimization. In: Proceedings of the Congress on Evolutionary Computation. IEEE Press, Los Alamitos (2000)
26. Knowles, J., Corne, D.: Approximating the nondominated front using the pareto archived evolution strategy. Evolutionary Computation 8(2), 149–172 (2000)
27. Dasheng Liu, C.K.G., Tan, K.C., Ho, W.K.: A multiobjective memetic algorithm based on particle swarm optimization. IEEE Transactions on Systems, Man and Cybernetics - Part B: Applications and Reviews 27(1), 42–50 (2007)
28. Garey, M.R., Johnson, D.S.: Computers and Intractability: A guide to the theory of NP-Completeness. W. H. Freeman and Company, New York (1979)
29. Veldhuizen, D.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovation. PhD thesis, Department of Electrical Engineering and Computer Engineering, Airforce Institue of Technology, Ohio (1999)
30. Tan, K., Lee, T., Khor, E.: Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization. IEEE Transactions on Evolutionary Computation 5(6), 565–588 (2001)

Multiobjective Memetic Algorithm and Its Application in Robust Airfoil Shape Optimization

Wenbin Song

Institute of Aerospace Science and Technology, Shanghai Jiao Tong University, Shanghai, China
wenbin.song@gmail.com

This chapter describes the use of a multiobjective memetic algorithm in aerodynamic shape optimization applications using computational fluid dynamics(CFD). A systematic overview on multiobjective optimization is given. It covers parameterized scalarization, population-based Pareto front forming using evolutionary algorithms, formulation of Pareto front improvement using local search methods, and various possible heuristics that can be applied in formulating efficient memetic algorithms. A constrained local search, combined with a non-dominated sorting genetic algorithm with niching in both objective and design space is proposed in the paper. Test function results are given and finally the method is used to solve the robust airfoil design problem using a multi-point formulation.

Numerical optimization is increasingly deployed in the development of products and services, to improve performance and reduce costs. Examples of these can be found in science, engineering and financial sectors. With the increasing accuracy of simulation models and decreasing cost of growing computing power, this trend is set to continue and will grow evermore sophisticated. However, traditional single objective numerical optimization methods based on the use of gradients to identify search directions often fail to locate global optimal solutions. The reasons for such failures have been well studied and various alternatives and remedies have been recommended by many researchers. These failures comes from the facts the many real-world problems are far more complex than test functions that are often used to develop and test algorithms. And these problems often have multiple objectives with various degrees of continuity, subject to multiple equality and inequality constraints. The number of variables are great and computational cost of each evaluation of objective or constraint functions are huge. Among various methods tackling these challenges, multiobjective memetic algorithms (MOMAs) appear to have provided a flexible framework, which, when properly designed and implemented, can find good solutions for complex problems whose objective and constraint functions can be efficiently evaluated.

The aerospace industry has been the driving force of many technological developments that later find application in other areas such as the design of Formula-1 cars. The multidisciplinary nature and complexity of aircraft design has motivated the use of numerical optimization methods in the drive to improve performances, reduce weights

C.-K. Goh, Y.-S. Ong, K.C. Tan (Eds.): Multi-Objective Memetic Alg., SCI 171, pp. 389–402.
springerlink.com

and costs. One of the important areas that optimization has been used is shape optimizations using computational fluid dynamics (CFD).[1] Initially, the problem was often set up to optimize one target that indicates the most important performance criterion such as the drag of the wing or the weight of structural components. This practice is expected to continue at the level of part or component design, but it proves to be less effective on the subsystem or system level when multiple operating conditions or disciplines are included in the equation.

Problems with multiobjectives are traditionally solved by converting multiple objectives into a single objective function using various scalarization methods. The most commonly used method is to apply some prior weighting coefficients to each objective and then to lump all the weighted objectives together to form a single, balanced objective function. The disadvantage of this approach is that it produces a single solution and it is sometimes difficult to decide in advance what is the most appropriate set of weights and as a result of any possible shift in the coefficients, the optimization process will have to be repeated. Furthermore, it often becomes necessary to scale different objective functions and different sets of weights could lead to the same results.

Another category of methods for solving multiobjective optimization problems focuses on generating an approximation to the entire Pareto front. Pareto front is consisted of solutions that are nondominated. The formal definition of nondominance and Pareto front will be given later, but roughly speaking, one solution nondominates another means that moving from one to the other, at least one objective function will degrade. The nature of Pareto front suggests that population-based evolutionary optimization methods seem to be a natural choice as these methods work on a population of solutions instead of a single one as in the scalarized methods described in the previous paragraph.

Memetic algorithms are a category of methods that combine the robustness of population based evolutionary search in finding global or near-global optimal points with the efficiency of a gradient descent methods in locating local improvements. Initially developed to solving single objective optimization problems, memetic algorithms have seen growing popularity in solving problems with multiple objectives, leading to the development of multiobjective memetic algorithms. The main concerns in multiobjective memetic algorithms are the heuristics used to combine the local search with population-based Pareto forming.

The rest of the chapter contains an overview on multiobjective evolutionary (MOEA) methods in the next section; A modified MOEA method is proposed based on the NSGA-II method and its application to some test functions and a real-world aerodynamic shape optimization problem is also presented.

1 Multiobjective Evolutionary Optimization

A generic multiobjective optimization problem can be defined as follows assuming all objective functions are to be minimized[3]:

$$\begin{aligned} &\text{Minimize} && f_i(\mathbf{x}), i = 1, \cdots, P, \\ &\text{subject to} && g_j(\mathbf{x}) \geq 0, j = 1, \cdots, J, \\ & && h_k(\mathbf{x}) = 0, k = 1, \cdots, K. \end{aligned} \tag{1}$$

where $\mathbf{x}$ is a vector containing n design variables, and P, J and K are the number of objectives, inequality and equality constraints, respectively.

The following definitions define relations between solution vectors, dominance, and Pareto front. These definitions form the basis for subsequent discussions.

Definition 1. *Vector* $\mathbf{f}(\mathbf{x}_i) \leq \mathbf{f}(\mathbf{x}_j)$, *if* $f_k(\mathbf{x}_i) \leq f_k(\mathbf{x}_j), k = 1, \cdots, P$ *and* $\mathbf{f}(\mathbf{x}_i) \neq \mathbf{f}(\mathbf{x}_j)$.

Definition 2. *Solution* $\mathbf{x}_i$ *dominates* $\mathbf{x}_j$ *if* $\mathbf{f}(\mathbf{x}_i) \leq \mathbf{f}(\mathbf{x}_j)$.

Definition 3. ***Pareto front (PF)*** *is the set of all nondominated solutions that no other solution can be found which dominates it.* ***Pareto optimization*** *is the process of finding an approximation to the* ***PF***.

1.1 Parameterized Scalarization for Multiobjective Optimization

Early attempts on multiobjective optimization problems focus on ways to convert it into single optimization problems so that the multitude of existing methods can be readily applied. The weighted sum method is probably the most commonly used among this type of approaches. A vector of objective functions $\mathbf{f}$ is turned into a congregated single value using the following equation

$$\sum_{i=1}^{P} w_i \cdot f_i(\mathbf{x}) \tag{2}$$

where $\sum_i w_i = 1$ and $w_i \geq 0$ $(i = 1, \cdots, P)$ are usually specified. The weight vector can be viewed as a measure of preferences for each objective function, and may be determined from domain knowledge. By varying the values of the weight vector, either randomly or systematically, a series of solutions can be found which can be seen as an approximation to the Pareto front. The drawback of the approach is that different weight vectors might lead to the same value in scaled objective space for different solutions, an issue caused by lack of mechanisms to compare the relative dominance between solutions.

1.2 Population-Based Pareto Front Forming

The parameterized scalarization method described in previous section can be used with any of the existing single objective optimization methods including population based evolutionary search such as genetic algorithms. Applying population-based methods and parallelization will no doubt increase the chance of achieving global optimization more efficiently, it does not address the inherent weaknesses in those methods. Instead of converting multiple objectives into a single value, the concept of dominance is used to rank the individuals in the population in order to give them a "overall" fitness value which reflects the relative dominance of each individual in the population. Two solutions that would otherwise have the same objective function in the weighted sum approach would be better differentiated based on their dominance against each other.

Population-based evolutionary optimization methods have inherent advantages when applied to multiobjective optimization problems since it deals with a population of solutions at each iteration. Individuals in the population are ranked according to their

dominance to determine their relative fitness values. One of the popular methods is nondominance sorting which was suggested by Goldberg[2] and later implemented in the popular NSGA[3]. Another popular procedure Strength Pareto EA (SPEA) was proposed by Zitzler and Thiele.[4] A tutorial of using various techniques for fitness ranking with genetic algorithms is offered by Kodak.[5]

As with classical evolutionary algorithms, diversity in population needs to be encouraged in order to avoid premature convergence of the procedure to less optimal solutions. This is often balanced with elitism which means the retention of good parents in the population from one generation to the next. In multiobjective EAs, this means the retention of nondominated solutions found so far, often in a separate or archive population. The differences among various methods for elitism are in the way that the main population and archive population interacts and how fitness values are assigned.

An issue unique to multiobjective EAs is fitness sharing among nondominated solutions. The aim of fitness sharing is to spread nondominated solutions across the Pareto front. The effective fitness of a solution within a niche of solutions is reduced to encourage diversity in the objective space. The niche is often defined in terms of distance of solutions to one another in the objective space. A survey on several approaches used for fitness sharing in the literature was provided by Knowles and Corne[6]. Several popular multiobjective evolutionary optimization methods are compared in Table 1.

Table 1. Feature comparisons of various MOEA methods

name	ranking	fitness sharing	elitism	Pareto archive	selection
NSGA-II[7]	population	crowding	yes	no	ranking based
MOGA[8]	population	yes	no	no	ranking based
NPGA[9]	binary	yes	no	no	tournament
PAES[10]	binary	hyper grid	yes	yes	-
SPEA[4]	population	Pareto domi-nacne	yes	yes	tournament
PESA[11]	population	hyper grid	yes	yes	ranking and crowding

This work uses a sightly modified implementation of NSGA-II[7] as the main framework of the multiobjective memtic algorithm. In addition to fitness sharing method in objective space as used in original NSGA-II, fitness sharing in design space is also implemented. Another modification adopted is regarding the selection process, this paper makes no change to the original selection process used in the base genetic algorithm. Instead, the fitness values are manipulated based on the ranking, crowdedness in both objective and design space. The average distance between neighboring points is normalized so that it does not exceed the difference in ranking between two adjacent Pareto front, i.e. nondominant solutions always get a better fitness value than dominant ones.

This principle is in line with the one used in the original NSGA-II, but easier to implement. The third modification is the introduction of an archive population which store all the design points in the current Pareto front.

1.3 Local Search Methods for Multiobjective Problems

Local search methods, and gradient-based method in particular, have been used in single-objective optimization problems. The most common criticism for this type of methods is that often local optimum are returned in the neighborhood of the starting point. However, hybridization of local search methods with evolutionary search methods has often lead to significant speedups in the overall optimization processes. This principle has been the basic merit for memetic algorithms and has seen wide application in single optimization problems. Recently, the same principle has been increasingly applied to multiobjective problems.[12, 13, 14]

The main issue for a hybrid local search and genetic algorithm is how to convert multiple objectives into one scalar value. The most commonly used scalarization method in local search is weighted sum, in which, multiple objective functions are turned into a single value by a weight vector. The weight vector itself can be chosen randomly[15] or can be adapted to change search direction from time to time, forming a chain of searches along different directions. Other approaches use multiple steps, applying local searches on each objective in isolation in each step.[16] Pareto ranking commonly used in population-based evolutionary approaches has also been used in local search based on comparing solutions to an archive of nondominated solutions.[17]

This work uses a constrained local searches on each objective function while treating other objective functions as constraints. The process is repeated for each objective function and result from each search step is used as starting point for the subsequent steps. The idea is consistent with the concept of Pareto dominance, and it also has the benefit of not having to specify any preferences among objective functions. The bounds on the constraints are updated in each step to reflect the current objective function values. The local search is applied to all solutions in the current Pareto set and this process can be run in parallel to improve the efficiency. This method is illustrated in Figure 1 for a two objective minimization problem. The optimization method used in local search is simulated annealing.

1.4 Performance Measures

Although it is possible and useful to compare test function results with theoretical Pareto front, more generic criteria that can be used to a wider scenarios are beneficial to evaluate whether a memetic algorithm performs better than other algorithms. As discussed by Knowles and Corne[6], the use of unary measures based on some values calculated from one approximate set of Pareto front can lead to misleading results. Based on the work by Zitzler et al.[18] on performance assessment of various multiobjective evolutionary methods, three criteria are proposed here to be used to compare the performance of different memetic heuristics. Multiple runs are carried out for each cases as is the common practice when comparing classical single objective evolutionary methods. These criteria are listed below.

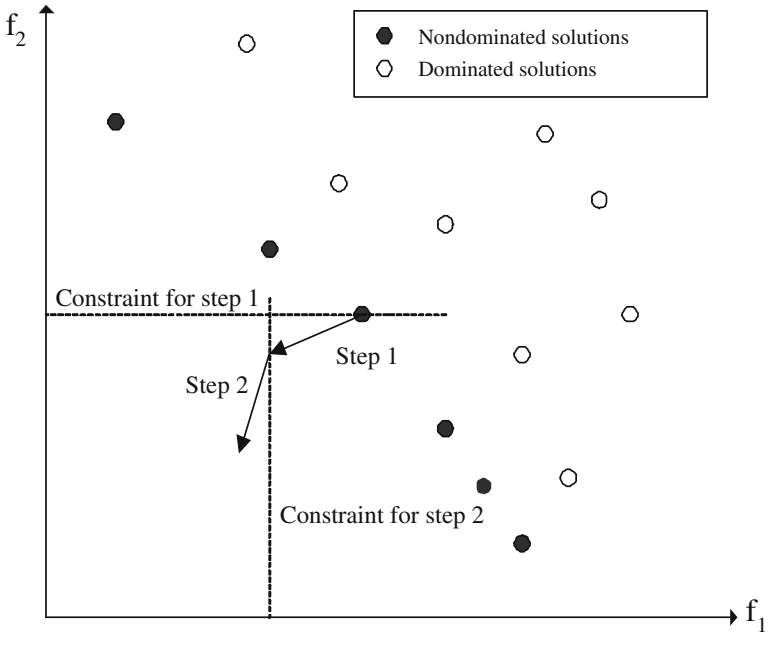


Fig. 1. Constrained local search for multiobjective optimization

- number of nondominated solutions in the final Pareto front
- number of solutions in one Pareto front that dominate at least one solution in the other Pareto front
- the distribution and the spread of Pareto solutions

Although too many Pareto solutions may present a challenge for the designer in some cases, it is generally the case that more Pareto solutions are desired. The issue on how to choose the best solutions from a large number of nondominated points remain a decision making problem for the designer, and some other techniques or domain knowledge may be required to make the best choice.

The second criterion simply compare the two Pareto front in terms of their relative dominance. This effectively measures the proximity of achieved set of solutions to the exact Pareto front.

The distribution and spread are computed in each objective space as the following quantities, which forms a vector of quality measurement

- Lebesgue measure (S metric)
- the average distance between neighboring solutions
- the variance of distances between neighboring solutions

The comparison between two Pareto fronts in terms of the distribution and the spread of Pareto solutions is carried out in two steps, first in each objective space comparison is performed in a similar fashion as used in dominance comparison in objective space, i.e. relative dominance of the quality vector. When one quality vector dominates another, it

means that the first Pareto front has better quality in terms of that particular objective function. Incompatible solutions are compared in terms of the number of better qualities, i.e. if three quality measures are better, the whole vector is deemed better than the other.

The second step is to simply compare the counts of objectives that has better quality in the results of the first step comparison. A Pareto front is deemed superior when it produces a better value in the comparison.

1.5 Metaheuristics in Mematic Algorithms

Various local search methods can combined with the evolutionary algorithms to improve computational efficiency. There are a number of options regarding when to invoke the local search and how many individuals in the population should undergo local searches. The balance between evolutionary search and local search was studied by Ishibuchi et al.[19] in order to take advantage of performance improvement without sacrificing the robustness of MOEAs in locating global Pareto front. The local search can be applied to the whole population in each iteration as was done in [17, 20], or can be used intermittently after a fixed number of generations[21] or at the end of a NSGA-II search.[22]

In the current work, instead of applying local searches on a single weighted-sum of all objective functions or on each objective function in isolation while ignoring other objectives, the local search step in the current work adopts a series of searches on each single objective, while treating all other objective functions as constraints with bounds taken to be values of corresponding objective functions. The complete process is illustrated in Figure 2.

Here, local searches are switched on in the first few generations. In addition, only a subset of randomly selected Pareto solutions undergo local searches. The number of solutions in the subset is decided as follows. All solutions in the first Pareto front will be optimized using local search methods, and the ratio of solutions that have seen

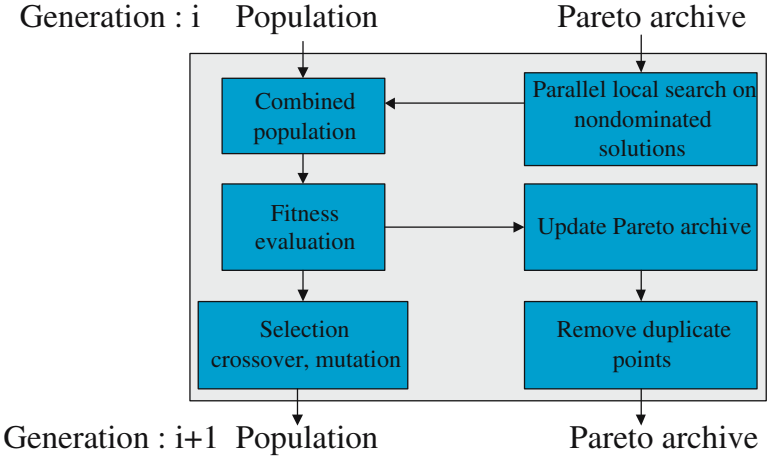


Fig. 2. Multiobjective memetic algorithm framework

improvements is calculated and used as the ratio in the selection of points in the next Pareto front. This process continues until local search is switched off or the ratio becomes zero, whichever comes first. This selective local search scheme aims to make the most effective use of local search procedures to improve the computational efficiency.

1.6 Test Function Results

This section presents some results on five commonly used test functions in multiobjective evolutionary research literature. Five of the six two-objective test functions suggested by Zitzler et al.[18] are used here to study the proposed framework. Each test function is run five times using the multiobjective algorithm modified from NSGA-II, with and without local search, respectively. Results for both the basic multiobjective genetic algorithm implemented and memetic algorithm are presented in Tables 2, 3 and 4. The population size and number of generations with the multiobjective GA is set to 100 and 50, respectively. Local searches are implemented in parallel, therefore the computational effort required for completing all the local searches is equivalent to a single local search when the number of iterations is fixed. When local search is enabled, the total number of function evaluations used is maintained roughly the same as those used in the method with no local searches, so the Pareto fronts from two approaches can be compared.

Some observations can be made from the test function results.

- MOMA generally performs better than MOEA in terms of Lebesgue measure, though only with a small margin.
- results are comparable when measured using Pareto front distribution and spread, and might exhibit different trend for different objectives.
- MOMA outperforms MOEA when the final Pareto fronts are compared.

It should be noted that the aim of using local search methods should be improve the efficiency of the search at initial stages of the processes. It is expected that when sufficient number of generations are run, the evolutionary methods tend to converge to better results and effect of local searches is diminishing. Efficient local searches and heuristics in using them are the key factors and should be an important area for further studies.

Table 2. Test function results - without local search

Test problem	Lebesgue measure	Pareto front distribution and spread	
		Average distance	Variance of Distance
zdt1	4.2983	0.0255	0.0015
		0.0654	0.0167
zdt2	0.7952	0.0590	0.0244
		0.1211	0.0245
zdt3	1.5252	0.0211	0.0028
		0.0761	0.0125
zdt4	96.4639	0.0164	0.0039
		9.4434	213.4590
zdt6	1.9663	0.0930	0.0513
		0.5611	0.3642

Table 3. Test function results - with local search

Test problem	Legesgue measure	Pareto front distribution and spread	
		Average distance	Variance of Distance
zdt1	4.3416	0.0235	0.0015
		0.0633	0.0246
zdt2	0.8685	0.0406	0.0197
		0.1178	0.0182
zdt3	1.5230	0.0201	0.0025
		0.0811	0.0131
zdt4	99.6333	0.0031	0.0001
		12.4293	400.0460
zdt6	2.0611	0.0651	0.0272
		0.4066	0.2318

Table 4. Test function results - dominance comparison

Test problem	Ratio of dominant solutions for MOEA	Ratio of dominant solutions by MOEA+LS
zdt1	0.0882	0.6122
	0	0.9211
	0.1579	0.6216
	0.7222	0.0303
	0.1220	0.5610
zdt2	0.3333	0.1667
	0	0.7500
	0	0.8333
	0.2667	0.6667
	1	0
zdt3	0.2308	0.5405
	0.7105	0.1489
	0.5854	0.1277
	0.5957	0.1000
	0.1395	0.5526
zdt4	0	1.0
	0	0.5714
	0	0.8462
	0.1538	0.7500
	0	0.7143
zdt6	0.1250	0.5000
	0.5000	0.8182
	0.5556	0.1429
	0.8333	0
	0.2667	0.4286

2 Aerodynamic Shape Optimization: A Brief Overview

Aerodynamic shape optimization is an important element of the overall multidisciplinary aircraft design process and increasingly, it finds applications in other industry such as automobile. The increasing availability of accurate flow analysis codes and high performance computing facilities has gradually placed CFD-based shape optimization methods into the reach of many designers.[23] It is now possible to calculate optimum three dimensional transonic wing shapes in a few hours, taking into account of viscous effects with the use of Reynolds averaged Navier-Stokes (RANS) equations. Typical shape optimization process starts with the definition of a parametric three dimensional CAD (Computed-Aided Design) model, which then undergoes various analysis steps leading to results of certain performance quantities. This process, when coupled with numerical optimization techniques, can lead to improved designs. This inherently multidisciplinary process often needs to be repeated several times. And this type of problems rarely contains a single objective, and the trade-offs between different aspects of the design, such as structural weight and cost, have to be made where conflicting requirements exists.

The objectives in aerodynamic design is typically lift, drag or a specified target pressure distribution. An approach which has become increasingly popular is to couple the CFD with genetic algorithms to search over a large design space for global optimal designs. One of the major issues in CFD-based shape optimization problems is the high computational cost associated with high fidelity CFD codes. Attempts have been made in several aspects to address the issue. For example, adjoint solvers have been developed to calculate gradient more efficiently. [24, 25] Morphing based technique has also been used to alleviate the need for re-meshing in shape optimization problems. [26] Surrogate modeling methods are probably the one of the most promising techniques to tackling the issue. [27, 28] Hybrid use of variable fidelity models has also been shown as an effective means reduce the turn around time. [29]

Another important issue in aerodynamic shape optimization problems is to produce shapes that are insensitive to operational uncertainties. Multi-point optimization is one of the methods used in robust airfoil design, typically using weighted sum single-objective optimization approach.[30, 31, 32] Multiobjective optimization method has also been studied.[33] Although it can be expected that domain knowledge should be incorporated into the heuristics when available, this paper focuses on the use of memetic algorithms only for robust airfoil optimization.

3 Robust Airfoil Shape Optimization Using MOMA

This section investigates the use of proposed multiobjective memetic algorithm in a robust airfoil shape optimization problem. Traditionally, airfoil shapes are optimized against a single flight condition, for example, at a specific Mach number and attitude. This method often proved to be inadequate as it might lead to airfoil shapes that would have a degraded performance at slightly different flight conditions. This drawback of single-point optimization prompted the research in multi-point approach in order to produce robust performances of an airfoil in a number of operating conditions. [34]

Here, an example application of airfoil shape optimization under multiple flight conditions is studied using multiobject memetic algorithm with constrained local search.

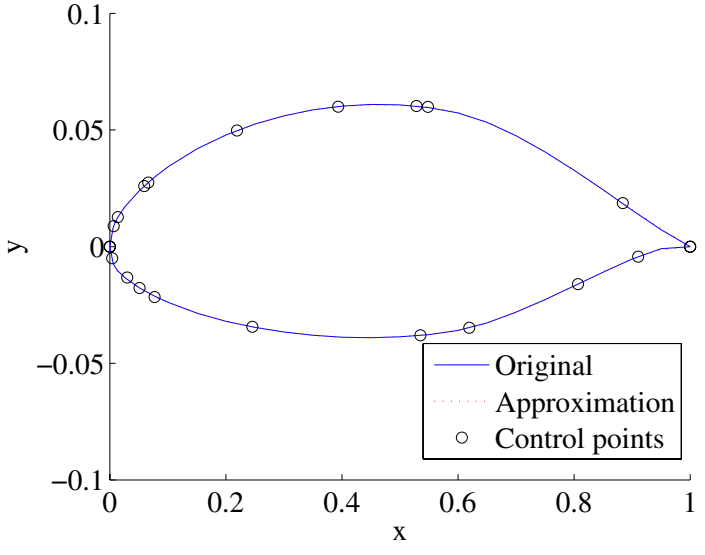


Fig. 3. B-Spline approximation of airfoil NACA66210

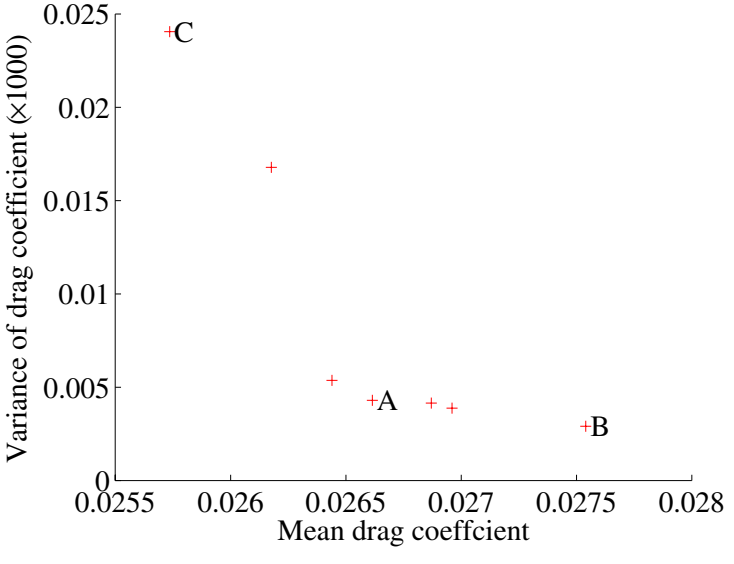


Fig. 4. Pareto front returned using MOMA method

The airfoil geometry is defined using a B-Spline interpolation and the coordinates of the control points are used as design variables. An example is shown in Figure 3, in which nine control points are used for both upper and lower surfaces. Consistent minimization of airfoil drag is sought at four flight conditions ($Ma = 0.72, 0.74, 0.76, 0.78$) subject

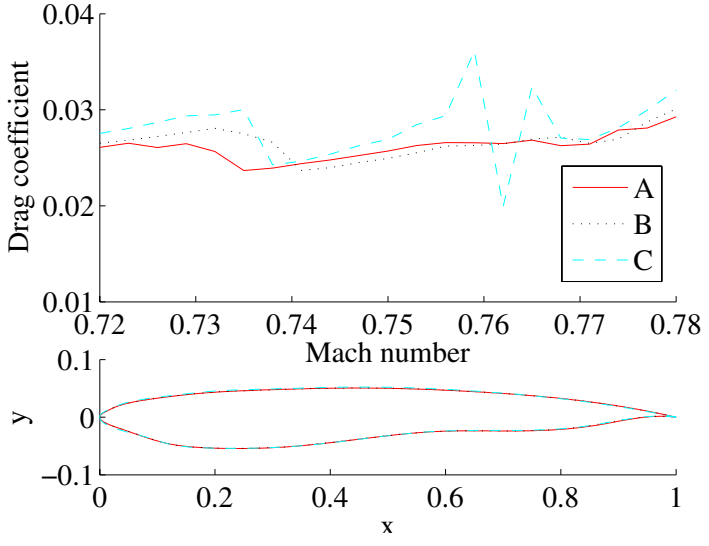


Fig. 5. Drag profile of selected airfoils

to lift and thickness contraint ($C_l \geq 0.50$ and $max(t/c) > 0.10$). The mean and variance of the drag values are used as the two objectives. The problem has 36 design variables defining the coordinates of control points.

The CFD codes used here is the viscous Garabedian and Korn (VGK) method which can model the two-dimensional, transonic, attached flow over airfoil shapes in a subsonic freestream, with allowance for viscous effects. The code is very efficient and can be run in large numbers, which is a typical requirement of evolutionary methods. The population size and number of generations is set to 200 and 200. The optimization resulted in a Pareto front of seven design points as shown in Figure 4. The Pareto front gives a reasonable coverage over the objective space. And it can be seen that a reduction in mean drag value would lead to a penalty on the robustness. Three out of the seven Pareto points are shown in Figure 5, in which, both the airfoil geometry and drag profile are shown. Among the three points, point A is clearly the most favorable design as it represents the best trade-off between low drag and robustness.

4 Conclusions

This chapter presents a multiobjetive memetic algorithm (MOMA) that hybridize a modified NSGA-II with constrained local search methods and its application in robust airfoil shape optimization problems. The algorithm is first applied to a number of commonly used test functions and results are compared between MOEA and MOMA. A selective local search scheme is also implemented to overcome the computational burden of carrying out excessive local searches. The framework is applied to a robust airfoil shape optimization problem. It remains to be studied how the constrained local search method and selective local search scheme compare with other methods.

Acknowledgment

The author was with University of Southampton where this work was carried out. The work was funded by the SIMDAT project (http://www.simdat.eu) within the European Commission Research Framework 6.

References

1. Hicks, R.M., Henne, P.A.: Wing Design by Numerical Optimization. Journal of Aircraft 15(7), 407–412 (1978)
2. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Reading (1989)
3. Srinivas, N., Deb, K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation 2(3), 221–248 (1995)
4. Zitzler, E., Thiele, L.: An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach. Technical Report 43, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (May 1998)
5. Knoak, A., Coit, D.W., Smith, A.E.: Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering and System Safety 91(9), 992–1007 (2006)
6. Knowles, J.D., Corne, D.: Memetic Algorithms for Multiobjective Optimization: Issues, Methods and Prospects. Studies in Fuzziness and Soft Computing 166, 313–354 (2004)
7. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A Fast and Elitist Multibjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
8. 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 Francisco (1993)
9. Horn, J., Nafpliotis, N.: Multiobjective Optimization using the Niched Pareto Genetic Algorithm, Technical Report IlliGAl Report 93005, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA (1993)
10. Knowles, J.D., Corne, D.W.: Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. Evolutionary Computation 8(2), 149–172 (2000)
11. Corne, D.W., Knowles, J.D.: The Pareto-envelope based selection algorithm for multiobjective optimization. In: Proceedings of the Sixth International Conference on Parallel Problem Solving from Nature (PPSN VI), pp. 838–848. Springer, Berlin (2000)
12. Ishibuchi, H., Kaige, S.: A simple but powerful multiobjective hybrid genetic algorithm, Design and application of hybrid intelligent systems, pp. 244–251 (2003)
13. Adra, S.F., Griffin, I., Fleming, P.J.: Hybrid multiobjective genetic algorithm with a new adaptive local search process, pp. 1009–1010 (2005)
14. Ishibuchi, H., Narukawa, K.: Some issues on the of implementation of local search in evolutionary multi-objective optimization, pp. 1246–1258 (2004)
15. Ishibuchi, H., Yoshida, T.: Implementation of Local Search in Hybrid Multi-Objective Genetic Algorithms: A Case Study on Flowshop Scheduling. In: Wang, L., Tan, K.C., Furuhashi, T., Kim, J.-H., Yao, X. (eds.) Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002), vol. 1, pp. 193–197. Nanyang Technical University, Orchid Country Club, Singapore (2002)
16. 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)

17. Knowles, J., Corne, D.: M-PAES: A Memetic Algorithm for Multiobjective Optimization. In: 2000 Congress on Evolutionary Computation, vol. 1, pp. 325–332. IEEE Service Center, Piscataway (2000)
18. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8(2), 173–195 (2000)
19. 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)
20. Ishibuchi, H., Murata, T.: Multi-Objective Genetic Local Search Algorithm. In: Fukuda, T., Furuhashi, T. (eds.) Proceedings of the 1996 International Conference on Evolutionary Computation, Nagoya, Japan, pp. 119–124. IEEE, Los Alamitos (1996)
21. Talbi, E.-G., Rahoual, M., Mabed, M.H., Dhaenens, C.: A hybrid evolutionary approach for multicriteria optimization problems: Application to the flow shop. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, p. 416. Springer, Heidelberg (2001)
22. Deb, K., Goyal, T.: Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergence, KanGAL report 200004, Indian Institute of Technology, Kanpur, India (2000)
23. Jameson, A., Kim, S., Shankaran, S., Leoviriyakit, K.: Aerodynamic Shape Optimization: Exploring the Limits of Design, KSAS 1st International Sessions in 2003 Fall Conference, GyeongJu Korea, November 14-15 (2003)
24. Kim, S., Alonso, J.J., Jameson, A.: Multi-element high-lift configuration design optimization using viscous continuous adjoint method. Journal of Aircraft 41(5), 1082–1097 (2004)
25. Carpentieri, G., Koren, B., van Tooren, M.J.L.: Adjoint-based aerodynamic shape optimization on unstructured meshes. Journal of Computational Physics 224(1), 267–287 (2007)
26. Rousseau, Y., Men'shov, I., Nakamura, Y.: Morphing-based shape optimization in computational fluid dynamics. Transactions of the Japan Society for Aeronautical and Space Sciences 50(167), 41–47 (2007)
27. Booker, A.J., Dennis, J.E., Frank, P.D., Moore, D.W., Serafini, D.B.: Managing surrogate objectives to optimize a helicopter rotor design - further experiments. In: The Seventh AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization (1998)
28. Jouhaud, J.C., Sagaut, P., Montagnac, M., Laurenceau, J.: A surrogate-model based multidisciplinary shape optimization method with application to a 2D subsonic airfoil. Computers & Fluids 36(3), 520–529 (2007)
29. Gano, S.E., Renaud, J.E., Sanders, B.: Hybrid variable fidelity optimization by using a kriging-based scaling function. Aiaa Journal 43(11), 2422–2430 (2005)
30. Cliff, S.E., Reuther, J.J., Saunders, D.A., Hicks, R.M.: Single-point and multipoint aerodynamic shape optimization of high-speed civil transport. Journal of Aircraft 38(6), 997–1005 (2001)
31. Epstein, B., Peigin, S.: Efficient Approach for Multipoint Aerodynamic Wing Design of Business Jet Aircraft. AIAA Journal 45(11), 2612–2621 (2007)
32. Peigin, S., Epstein, B.: Multipoint aerodynamic design of wing-body configurations for minimum drag. Journal of Aircraft 44(3), 971–980 (2007)
33. Nemec, M., Zingg, D.W., Pulliam, T.H.: Multipoint and multi-objective aerodynamic shape optimization. AIAA Journal 42(6), 1057–1065 (2004)
34. Li, W., Huyse, L., Padula, S.: Robust airfoil optimization to achieve drag reduction over a range of Mach numbers. Structural and Multidisciplinary Optimization 24(1), 38–50 (2002)

Author Index