Studies in Computational Intelligence 210

Andrew Lewis
Sanaz Mostaghim
Marcus Randall (Eds.)

Biologically-inspired Optimisation Methods

Parallel Algorithms, Systems and Applications

Springer

Andrew Lewis, Sanaz Mostaghim, and Marcus Randall (Eds.)

Biologically-Inspired Optimisation Methods

Studies in Computational Intelligence, Volume 210

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

Andrew Lewis, Sanaz Mostaghim,
and Marcus Randall (Eds.)

Biologically-Inspired Optimisation Methods

Parallel Algorithms, Systems and Applications

Springer

Dr. Andrew Lewis
Institute for Integrated and Intelligent Systems
Griffith University
Nathan Campus
Brisbane, Queensland, 4111
Australia
Email: a.lewis@griffith.edu.au

Assoc.Prof. Marcus Randall
Faculty of Business
Technology and Sustainable Development
Bond University
Gold Coast, Queensland, 4229
Australia
Email: mrandall@bond.edu.au

Dr.-Ing. Sanaz Mostaghim
Institut für Angewandte Informatik und
Formale Beschreibungsverfahren - AIFB
Universität Karlsruhe
76128 Karlsruhe
Germany
Email: smo@aifb.uni-karlsruhe.de

ISBN 978-3-642-01261-7 e-ISBN 978-3-642-01262-4

DOI 10.1007/978-3-642-01262-4

Studies in Computational Intelligence ISSN 1860949X

Library of Congress Control Number: Applied for

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

Preface

Throughout the evolutionary history of this planet, biological systems have been able to adapt, survive and flourish despite the turmoils and upheavals of the environment. This ability has long fascinated and inspired people to emulate and adapt natural processes for application in the artificial world of human endeavours. The realm of optimisation problems is no exception. In fact, in recent years biological systems have been the inspiration of the majority of meta-heuristic search algorithms including, but not limited to, genetic algorithms, particle swarm optimisation, ant colony optimisation and extremal optimisation.

This book presents a continuum of biologically inspired optimisation, from the theoretical to the practical. We begin with an overview of the field of biologically-inspired optimisation, progress to presentation of theoretical analyses and recent extensions to a variety of meta-heuristics and finally show application to a number of real-world problems. As such, it is anticipated the book will provide a useful resource for reseachers and practitioners involved in any aspect of optimisation problems.

The overview of the field is provided by two works co-authored by seminal thinkers in the field. Deb's "Evolution's Niche in Multi-Criterion Problem Solving", presents a very comprehensive and complete overview of almost all major issues in Evolutionary Multi-objective Optimisation (EMO). This chapter starts with the original motivation for developing EMO algorithms and provides an account of some successful problem domains on which EMO has demonstrated a clear edge over their classical counterparts.

Jaimes and Coello Coello's "Applications of Parallel Platforms and Models in Evolutionary Multi-Objective Optimization" presents an overview of state-of-the-art systems that exploit coarse and fine grained parallelism to solve multi-objective optimisation problems. Standard parallelisation models are reviewed in the context of multi-objective optimisation, and methods for the detailed assessment of their performance are discussed. In addition, a number of novel schemes for parallelisation of multi-objective evolutionary algorithms are briefly reviewed. The discussion also includes comment on how the global phenomenon of meta-computing can be used to solve these problems.

Global meta-computing brings a new set of problems to be overcome in the implementation of optimisation algorithms. In particular, the heterogeneous and dynamic nature of the computing environment require that greater consideration be given to the fault tolerance of algorithms. Lewis, Mostaghim and Scriven begin to consider these issues in "Asynchronous Multi-Objective Optimisation in Unreliable Distributed Environments", analysing the performance of multi-objective particle swarm optimisation (MOPSO) algorithms in unreliable computing environments, giving a detailed consideration of a novel approach of asynchronous updates in parallel MOPSO algorithms that significantly improves fault tolerance, and suggesting a variety of methods for adapting algorithms to "churn" of computing resources.

The consideration of recent and emerging developments of metaheuristics is continued by an exploration of dynamic optimisation problems, a class of optimisation problems that have many real-world characteristics, yet have received relatively little attention. These are difficult problems that change their structure and/or problem data while the meta-heuristic attempts to solve the problem. A comprehensive survey of genetic algorithms, particle swarm optimisation, ant colony optimisation and extremal optimisation approaches and implementations is presented in Hendtlass, Moser and Randall's "Dynamic Problems and Nature Inspired Meta-heuristics". For each of the methods discussed, consideration is given to practical issues of application to a variety of benchmark and real-world problems.

Artificial neural networks are another group of biologically inspired techniques, well suited to pattern recognition tasks. "Relaxed Labelling using Distributed Neural Networks" by Jim Austin explores a form of neural networks, known as correlation matrix memories, and their use in implementing the relaxation labelling technique for dealing with constraint satisfaction problems, in particular graph matching. The methods are built in the Advanced Uncertain Reasoning Architecture (AURA), a tool framework made available free on the Internet. An interesting application demonstrated is the matching of drug-like molecules against a large database of molecules that have potential anti-cancer properties. In order to improve the speed of search, the author also describes innovative methods for implementing the graph matcher in computer hardware

Randall, Hendtlass and Lewis, in "Extremal Optimisation of Assignment Type Problems", present a theoretical and practical exposition of the capabilities of the nature-inspired Extremal Optimisation. They extend the extremal optimisation metaphor so that it is able to handle constraints and reduce solution infeasibility in a standard way. In addition, a partially adaptive population model is also presented. Results of empirical investigations reveal that this simple meta-heuristic is very competitve with more established optimisation techniques, for a range of assignment type problems.

Enhancing another meta-heuristic, ant colony optimisation, is the subject of Angus' "Niching for Ant Colony Optimisation". Niching is a technique derived from the biological notion that different species will specialise in the exploitation of different parts of the environment. In the computational and optimisation sense, it refers to different individuals or populations of solutions exploring different parts of the search space, thus ensuring sufficient overall diversity. Two alternative forms of niching, based on crowding and fitness sharing concepts, are shown to be particularly effective for multi-modal and multi-objective problems.

The remainder of the book is concerned with the application of biologically inspired optimisation methods to problems that commonly occur in industry and the sciences. These particularly demonstrate that improved and novel solutions are capable of being generated to problems that have been traditionally exclusively the domain of human experts.

Computational optimisation found early adoption in the field of engineering design and manufacture. The design of radio antennas is an area that has historically been dominated by the considerable use of domain expertise and analytic solution methods. In recent years there has been an explosion of interest in automated design by the use of optimisation meta-heuristics, extending the ability of engineers to consider previously intractable problems. An example is the use of ant colony optimisation for the construction of compact meander line antennas for Radio Frequency IDentification (RFID) devices. Lewis, Randall, Galehdar, Thiel and Weis, in "Using Ant Colony Optimisation to Construct Meander-line RFID Antennas", present a developmental history of how the authors have solved this real-world problem - from initial application, the use of a novel local refinement technique and finally a multi-objective version that is able to optimise both antenna efficiency and resonant frequency. Results from computational experiments demonstrate the significant improvements that can be achieved.

A very practical problem in the area of establishing communication infrastructures is the radio network design problem. Mendes, Gómez-Pulido, Vega-Rodríguez, Sánchez-Pérez, Sáez and Isasi, in their chapter entitled "The Radio Network Design Optimization Problem", examine how a number of different biologically inspired meta-heuristics, including GRASP, genetic algorithms and memetic algorithms, perform on a large and difficult network design problem. Radio network design, in general, is an NP-complete problem and while a number of different approaches have been used to address it all lack a comparable measure of efficiency. Mendes *et al.* offer a reliable benchmark reference, and use it to investigate different algorithms and the reproducibility of their results.

A different form of network problem is the distribution of electricity through power grids. In particular, the issue of the imbalance between predicted electricity use and actual consumption is of great importance when reducing greenhouse gas emissions. Kamper and Eßer's chapter "Strategies for Decentralised Balancing Power" shows how a self-organising approach, based on evolutionary algorithms, can reduce this imbalance by dynamically pooling small electrical devices (such as washing machines and combined heat and power plants) together.

Conformational sampling, the prediction of the three-dimensional shapes of molecules based on their composition and connectivity, is a central problem in structural biology and drug design. There is a continuing search for general approaches to finding the most stable molecular geometries. In “An Analysis of Dynamic Mutation Operators for Conformational Sampling” Tantar, Melab and Talbi use this problem as a case study to examine the use of an *a priori* mutation operator selection and parameter tuning phase prior to execution of an evolutionary algorithm. They conclude, in part, that dynamic approaches, possibly including self-adaptive schemes, hold the most promise for tackling these extremely difficult problems.

In contrast to the preceding chapters on application of optimisation algorithms to problems in the physical sciences the book closes with a study from the field of artificial intelligence (AI). While much attention has been focussed on the use of AI for playing chess, the chapter by Quek, Chan, Tan and Tay, “Evolving Computer Chinese Chess using Guided Learning” examines evolutionary algorithms applied to playing Chinese chess. They explore how different heuristics, and indeed the knowledge of grandmasters of the game, can be used and integrated with genetic algorithms in order to produce an artificial player that can realistically challenge human opponents.

The underlying problems that the methods and techniques discussed in this book address are typically complex and demanding. In particular, the computational requirements can often be considerable and so efforts must be made to provide sufficient computing capacity to meet these needs. Currently, this generally makes parallel computing a necessity, and this is a consistent theme of the approaches considered by the contributing authors, whether explicitly, as in the overview of Jaimes and Coello Coello, or implicit in the nature of the methods adopted by several others: the population-based methods of , for example, particle swarm and ant colony optimisation algorithms, and such techniques as neural networks. Indeed, the drive for computational performance can be seen in the implementation of algorithms in hardware described by Austin.

The form in which parallel computing resources are provided can bring its own set of challenges. The search for cost-effective means of accessing large computational capacity has given rise to a trend toward grid computing and distributed, peer-to-peer computing environments. Highly dynamic, heterogeneous and prone to failure, these resources demand regard for the fault tolerance of optimisation algorithms, and efforts to address this issue such as those of Lewis, Mostaghim and Scriven will become increasingly important as the approaches become more widely employed.

The editors wish to acknowledge a number of groups and individuals that helped to make this project realisable. First of all, we wish to thank Professor Janusz Kacprzyk, Editor in Chief of the Studies in Computational Intelligence

series, for initially proposing the project and his continuing support. The series team at Springer-Verlag, in particular Dr Thomas Ditzinger and Heather King, are to be thanked for all their helpful advice and support. Along with them, we pay tribute to the work of the authors. Their outstanding ideas will resonate with the optimisation community in years to come. Finally, we thank the external reviewers for their astute comments and suggestions on each of the chapters.

February 2009
Brisbane, Australia, — Andrew Lewis
Karlsruhe, Germany, and — Sanaz Mostaghim
Gold Coast, Australia — Marcus Randall

Contents

Evolution's Niche in Multi-Criterion Problem Solving

Kalyanmoy Deb

Abstract. In a short span of about 15 years, evolutionary multi-objective optimization (EMO) has progressed on a fast track in proposing, implementing, and applying efficient methodologies based on nature-inspired computational algorithms for optimization. In this chapter, we briefly describe the original motivation for developing EMO algorithms and provide an account of some successful problem domains on which EMO has demonstrated a clear edge over their classical counterparts. More success studies exist and many more problem areas are needed to be explored. Hopefully, this chapter provides an indication and flavor of some such problem domains which may get benefited from a systematic application of an EMO procedure.

1 Introduction

With the rise and success of evolutionary multi-objective optimization (EMO) research and application, it becomes an interesting matter to discuss the problem areas in which EMO has reportedly demonstrated its edge over their classical counterparts. Such a discussion not only portrays the extent of its success, it also provides a direction and domain of problems areas along which EMO researchers and applicationists may find further grounds for success. In this paper, we attempt to discuss a few such problem areas on which EMO has demonstrated a clear edge.

Department of Business Technology,
Helsinki School of Economics,
Runeberginkatu 22-24, FIN-00101,
Helsinki, Finland
Kalyanmoy.Deb@hse.fi
and
Department of Mechanical Engineering,
Indian Institute of Technology Kanpur,
PIN 208016, India
deb@iitk.ac.in

EMO methodologies follow the principles of evolutionary optimization (EO) which has its roots starting with John Holland's conceptualization of a genetic adaptive search as a mechanism for creating new solutions in his cellular automata studies in 1962 [15], followed by seminal studies and adaptations of many of his students and followers. Despite some early suggestions for the need for solving multi-objective problem solving using an EO, the real EMO algorithms were suggested in early Nineties. But the initial algorithms and studies were effective and attractive enough for new-comers and for new ideas to make the EMO research a field of its own. In a recent survey announced during the World Congress on Computational Intelligence (WCCI) in Vancouver 2006, EMO has been judged as one of the three fastest growing fields of research and application among all computational intelligence topics. Among the 10 'Top Accessed Documents' reported in IEEE Transactions on Evolutionary Computation journal website, a 2002 EMO paper [9] is listed on the top. One single fact about multi-objective optimization has elevated EMO research and application on the top. Multi-objective problem solving ideally gives rise to a number of optimal solutions, instead of one, and an EMO procedure works with a population of solutions in every iteration, thereby potentially making it capable of maintaining a set of different solutions, if needed, in an optimization task.

The research in EMO had not been all about generating proof-of-principle algorithms for finding multiple trade-off solutions; EMO researchers continuously progressed in developing algorithms which were more and more computationally efficient. When two objectives were solved with some reasonable success, EMO researchers concentrated in solving three and more objective problems, which were found to be exceedingly more difficult to solve than two-objective problems. An increased attention in this direction needed EMO researchers to develop scalable test problems and associated performance metrics which can be used to compare and contrast different algorithms. With efficient algorithms being available both commercially and freely through downloadable codes, EMO applications became more and more popular in practice. With EMO's showing their niche in handling two, three and four-objective problems quite regularly and efficiently compared to classical methods, EMO researchers started to address an important piece of the multi-objective problem solving puzzle – an integrated optimization-cum-decision-making task which would allow one to solve multi-objective optimization problems till a single preferred solution is found. This required EMO researchers to integrate multiple criterion decision making (MCDM) principles with EMO algorithms. In a short span of about 15 years, the EMO has migrated itself from a mere collection of computer algorithms implementing some basic ideas to a fast-maturing field of research and application solving serious practical problems spanning in many fields of science and engineering, involving researchers with mathematical, scientific, engineering, and business backgrounds, and following a systematic yet a collective collaborating effort among theoreticians, practitioners, and algorithmists.

Different niches of EMO discussed in this chapter are some representative problem domains which EMO showed its promise. There exist many other areas which can be found in the EMO literature. Hopefully, this chapter will encourage motivated readers to take clues and help increase EMO's niche in a more profound way.

2 Multi-Objective Problem Solving

A multi-objective optimization problem involves a number of objective functions which are to be either minimized or maximized subject to a number of constraints and variables bounds:

$$\left.\begin{array}{rll} \text{Minimize/Maximize } & f_m(\mathbf{x}), & m = 1,2,\ldots,M; \\ \text{subject to } & g_j(\mathbf{x}) \geq 0, & j = 1,2,\ldots,J; \\ & h_k(\mathbf{x}) = 0, & k = 1,2,\ldots,K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1,2,\ldots,n. \end{array}\right\} \quad (1)$$

A solution $\mathbf{x} \in \mathbf{R}^n$ is a vector of n decision variables: $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$. The solutions satisfying constraints and variable bounds constitute a *feasible decision variable space S*.

The optimal solutions in multi-objective optimization can be defined from a mathematical concept of *partial ordering*. In the parlance of multi-objective optimization, the term *domination* is used for this purpose. The domination between two solutions is defined as follows [4, 20]:

Definition 1. A solution $\mathbf{x}^{(1)}$ is said to dominate the other solution $\mathbf{x}^{(2)}$, if both the following conditions are true:

1. The solution $\mathbf{x}^{(1)}$ is no worse than $\mathbf{x}^{(2)}$ in all objectives. Thus, the solutions are compared based on their objective function values (or location of the corresponding points ($\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$) on the objective space).
2. The solution $\mathbf{x}^{(1)}$ is strictly better than $\mathbf{x}^{(2)}$ in at least one objective.

For a given set of solutions (or corresponding points on the objective space, for example, those shown in Figure 1(a)), a pair-wise comparison can be made using the above definition and whether one point dominates the other can be established. All points which are not dominated by any other member of the set are called the non-dominated points of class one, or simply the non-dominated points. For the set of six solutions shown in the figure, they are points 3, 5, and 6. One property of any two such points is that a gain in an objective from one point to the other happens only due to a sacrifice in at least one other objective. This *trade-off* property between the non-dominated points makes the practitioners interested in finding a wide variety of them before making a final choice. These points make up a front when viewed them together on the objective space; hence the non-dominated points are often visualized to represent a *non-dominated front*. The computational effort needed to select the points of the non-dominated front from a set of N points is $O(N \log N)$ for 2 and 3 objectives, and $O(N \log^{M-2} N)$ for $M > 3$ objectives [13].

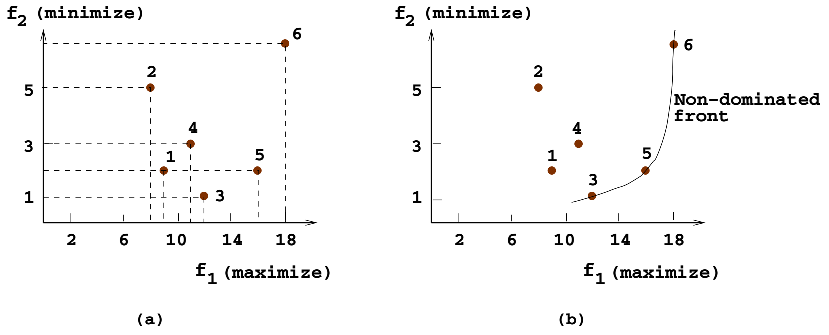


Fig. 1 (a) A set of points and (b) corresponding non-dominated front are shown.

With the above concept, now it is easier to define the *Pareto-optimal solutions* in a multi-objective optimization problem. If the given set of points for the above task contain all points in the search space (assuming a countable number), the points lying on the non-domination front, by definition, do not get dominated by any other point in the objective space, hence are Pareto-optimal points (together they constitute the Pareto-optimal front) and the corresponding pre-images (decision variable vectors) are called Pareto-optimal solutions. However, more mathematically elegant definitions of Pareto-optimality (including the ones for continuous search space problems) exist in the multi-objective optimization literature [20, 17].

2.1 *Evolutionary Principles*

In the context of multi-objective optimization, the extremist principle of finding the optimum solution cannot be applied to one objective alone, when the rest of the objectives are also important. This clearly suggests two ideal goals of multi-objective optimization:

Convergence: Find a set of solutions which lie on the Pareto-optimal front, and
Diversity: Find a set of solutions which are diverse enough to represent the entire range of the Pareto-optimal front.

EMO algorithms attempt to follow both the above principles, similar to the other a posteriori MCDM (or generating) methods [20]. Figure 2 shows schematically the principles, followed in an EMO procedure. Since EMO procedures are heuristic based, they may not guarantee in finding Pareto-optimal points, as a theoretically provable optimization method would do for tractable (for example, linear or convex) problems. But EMO procedures have essential operators to constantly improve the evolving non-dominated points (from the point of view of convergence and diversity mentioned above) similar to the way most natural and artificial evolving

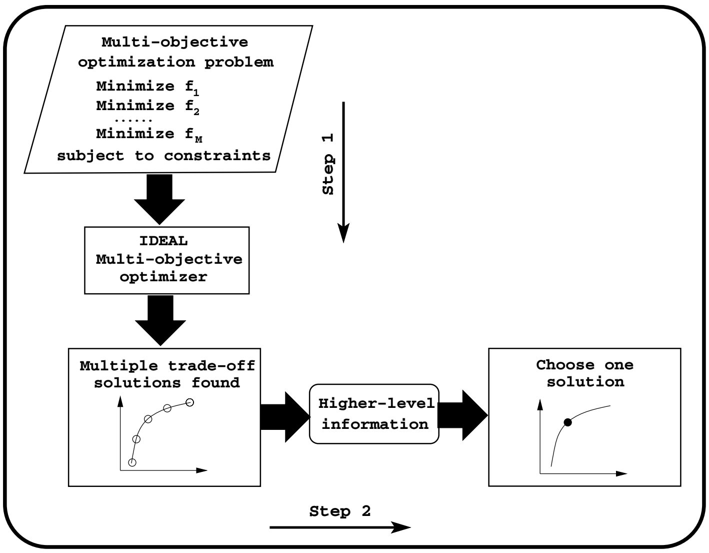


Fig. 2 Schematic of a two-step multi-objective optimization procedure.

systems continuously improve their solutions. The main difference and advantage of using an EMO compared to a posteriori MCDM procedures is that multiple trade-off solutions can be found in a single simulation run, as most a posteriori MCDM methodologies would require multiple applications.

In Step 1 of the EMO-based multi-objective optimization (the task shown vertically downwards in Figure 2), multiple trade-off, non-dominated points are found. Thereafter, in Step 2 (the task shown horizontally, towards the right), higher-level information is used to choose one of the obtained trade-off points.

2.1.1 Potential Reasons for EMO's Niche

Besides working with a population of solutions in every generation – a property which is one of the main factors contributing to EMO's niche in solving multi-objective optimization problems – there exists a few other properties which give an EMO its niche. First, an EMO uses flexible and easily updatable operators to suit a problem. For example, EMO practically allows an user to use any representation scheme for a solution. In many difficult problems, this flexibility allows an EMO to use a natural representation of the problem variables. To illustrate, continuous and discrete variables can be represented within an EMO algorithm as they are, without the need of any artificial fix-up. Second, an EMO can be started from a biased initial population, in the event of an availability of any problem knowledge. Third, an EMO allows further problem information to be included in its selection, recombination and mutation operators to help create new and improved solutions with a higher probability. Fourth, an EMO allows any termination condition to be used, including problem-specific, goal-oriented, or mathematical optimality conditions.

Specific to multi-objective optimization, a recent study [5] has shown that some specific EMO algorithms, such as the elitist non-dominated sorting genetic algorithm or NSGA-II [9], possesses a modular approach, thereby allowing a minor and conceivable change to the algorithm to suit it to solve different kinds of multi-objective problems.

In addition to these specific niches (population approach, algorithmic flexibility and modularity), the principle of EMO algorithms give them a considerable edge over their traditional counterparts. In the following, we describe some such advantages which an EMO helps provide.

3 EMO's Niches in Handling a Few Objectives

In this section, we demonstrate EMO's niche in solving two or three objectives for two different tasks: (i) while generating a representative set of Pareto-optimal front and (ii) while generating a preferred set of Pareto-optimal front.

3.1 Generating Pareto-optimal Solutions

It is discussed above that an EMO is ideally suited for generating a representative set of complete Pareto-optimal front. After some initial studies it has clearly come out that EMO algorithms can be effectively used for a few objectives (arguably, up to four) for this purpose. Here, we first discuss why an EMO may be perform better than classical scalarized methodologies and then present some comparative simulation results.

In the 'a posteriori' MCDM approaches (also known as 'generating MCDM methods'), the task of finding multiple Pareto-optimal solutions is achieved by executing many independent single-objective optimizations, each time finding a single Pareto-optimal solution. A parametric scalarizing approach (such as the weighted-sum approach, ε-constraint approach, and others [20]) can be used to convert multiple objectives into a parametric single-objective objective function. By simply varying the parameters (weight vector or ε-vector) and optimizing the scalarized function, different Pareto-optimal solutions can be found. In contrast, in an EMO, multiple Pareto-optimal solutions are attempted to be found in a single simulation by emphasizing multiple non-dominated and isolated solutions. We discuss a little later some EMO algorithms describing how such dual emphasis is provided, but now discuss qualitatively the difference between a posteriori MCDM and an EMO approach.

Consider Figure 3, in which we sketch how multiple independent parametric single-objective optimizations may find different Pareto-optimal solutions in different simulations. The Pareto-optimal front corresponds to global optimal solutions

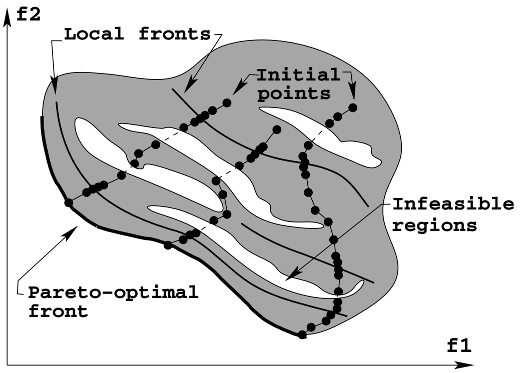


Fig. 3 Posteriori MCDM methodology employs independent single-objective optimizations.

of several scalarized objectives. However, during the course of an optimization task, algorithms must overcome a number of difficulties, such as infeasible regions, local optimal solutions, flat regions of objective functions, isolation of optimum, etc., to converge to the global optimal solution. Moreover, due to practical limitations, an optimization task must also be completed in a reasonable computational time. This requires an algorithm to strike a good balance between the extent of these tasks its search operators must do to overcome the above-mentioned difficulties reliably and quickly. When multiple simulations are to performed to find a set of Pareto-optimal solutions, the above balancing act must have to performed in every single simulation. Since simulations are performed independently, no information about the success or failure of previous simulations is used to speed up the process. In difficult multi-objective optimization problems, such memory-less a posteriori methods may demand a large overall computational overhead to find a set of Pareto-optimal solutions. Moreover, even though the convergence can be achieved in some problems, independent simulations can never guarantee finding a good distribution among obtained points.

EMO constitutes an inherent parallel search. When a population member overcomes certain difficulties and make a progress towards the Pareto-optimal front, its variable values and their combination reflect this fact. When a recombination takes place between this solution and other population members, such valuable information of variable value combinations gets shared through variable exchanges and blending, thereby making the overall task of finding multiple trade-off solutions a parallelly processed task. To illustrate this behavior, we show in Figure 4 the non-dominated solutions of a NSGA-II simulation at four different generations starting at 10th generation [25]. In the next figure (Figure 5), we show non-dominated solutions at the same generations of another NSGA-II run with identical starting population and parameter settings, except that at generation 10, one of the non-dominated

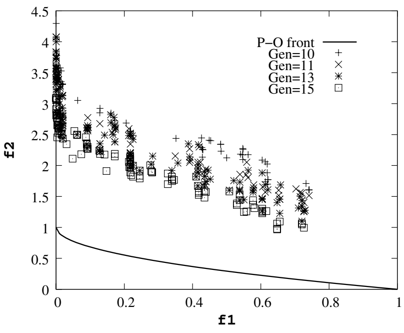


Fig. 4 Non-dominated fronts at several generations with the original NSGA-II in ZDT1.

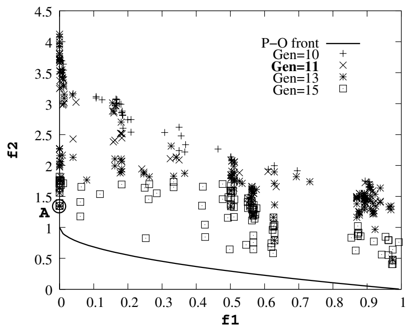


Fig. 5 Introduction of a single good point allows an improvement of the entire non-dominated front in ZDT1.

solutions is replaced by a new solution obtained by applying a local search (SQP) algorithm to the solution using the achievement scalarizing method [26]. The local searched solution is marked in the figure. It is interesting to observe how the introduction of a single good solution at generation 10 influences the progress of many other solutions in the population at subsequent generations. The good solution quickly gets recombined with other population members to bring them closer to the Pareto-optimal front. Such an inherent parallel processing of a population of points provides an EMO its success in multi-objective optimization and is absent in classical one-at-a-time generating algorithms.

3.1.1 Comparison of an EMO with the Normal Constraint (NC) Method

Three test problems are chosen in such a way so as to systematically investigate the performance of both NSGA-II (an EMO algorithm) and the NCM (a scalarized generating method) [19]. For the NSGA-II, we use a standard real-parameter SBX and polynomial mutation operator with $\eta_c = 10$ and $\eta_m = 10$, respectively [4]. For all problems solved using NSGA-II, we use a population of size 100.

First, we consider two-objective ZDT1 test problems [4, 27]:

$$\begin{aligned}
&\text{Minimize } f_1(\mathbf{x}) = x_1,\\
&\text{Minimize } f_2(\mathbf{x}) = g(\mathbf{x})\left(1 - \sqrt{\frac{x_1}{g(\mathbf{x})}}\right),\\
&\quad\text{where } g(x) = 1 + \tfrac{9}{n-1}\textstyle\sum_{i=2}^{n} x_i^2,
\end{aligned} \tag{2}$$

where the box constraints are $x_1 \in [0,1]$, and $x_i \in [-1,1]$ for $i = 2,3,\ldots,n$. Here, we choose $n = 30$. This ZDT1 problem has a convex Pareto-optimal front and ideally should not provide any difficulty to an algorithm except that the number of variables is large. Pareto-optimal solutions correspond to $0 \le x_1^* \le 1$ and $x_i^* = 0$ for $i = 2,3,\ldots,n$.

First, we apply the NC method [19], which uses an additional inequality constraint to restrict the feasible objective space. Figure 6 shows the obtained front after 20,000 function evaluations. It is clear that the NC method performs extremely well on ZDT1, both in terms of convergence and maintenance of diversity. Next, we apply NSGA-II for a total of 20,000 function evaluations. Figure 7 shows that a good distribution is achieved. Based on these simulations, it can be concluded that the ZDT1 problem is solved well by both NC method and an EMO method.

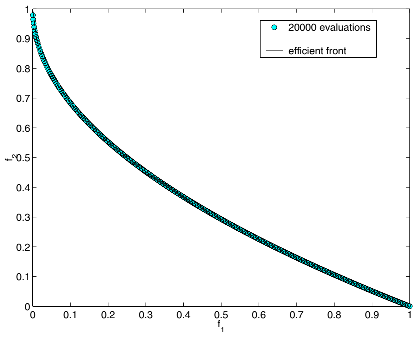


Fig. 6 Performance of NC method on ZDT1.

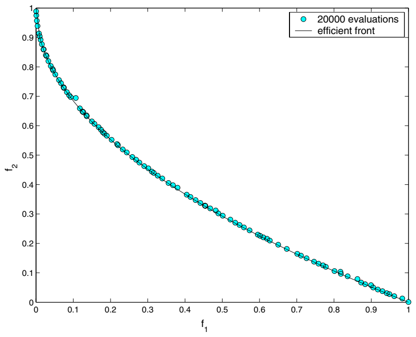


Fig. 7 Performance of NSGA-II on ZDT1.

Next, we use the 10-variable ZDT4 test problem [4]. This problem has a total of 100 distinct local efficient fronts in the objective space. The global Pareto-optimal solutions correspond to $0 \leq x_1^* \leq 1$ and $x_i^* = 0$ for $i = 2, 3, \ldots, n$. The algorithms face a difficulty in overcoming a large number of local fronts and converging to the global front: $f_2 = 1 - \sqrt{f_1}$.

Figure 8 shows the performance of the NC method after 100,000 evaluations. The multi-modality of the search space causes the NC method to not find the global efficient frontier, instead gets stuck to one of the local frontiers. However, Figure 9 shows that NSGA-II with only 20,000 evaluations is able to converge to the global efficient frontier. This problem makes a clear distinction between population-based and point-by-point-based generating methods of solving multi-objective optimization problems. In this problem, to converge to the efficient frontier, an optimization algorithm must have to overcome a number of local efficient frontiers. Since in finding every Pareto-optimal solution, such hurdles have to be overcome every time, it is too demanding for a classical point-by-point approach to expect the task to be performed well every time. On the other hand, a population-based approach recombines the decision variable vectors of good solutions to create new and hopefully better solutions. Thus, if one population member somehow reaches close to the global efficient frontier, it can *pull* the rest of population members close to the global frontier by means of interactions, thereby causing a fast and implicitly parallel search. This aspect of an EMO procedure was illustrated through an example problem in the earlier subsection.

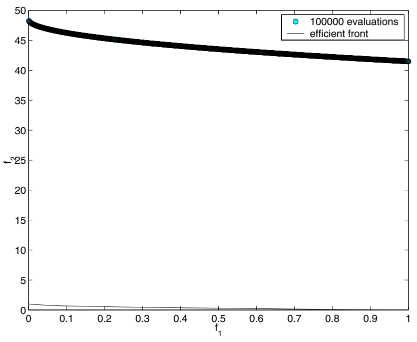


Fig. 8 Performance of NC method on ZDT4.

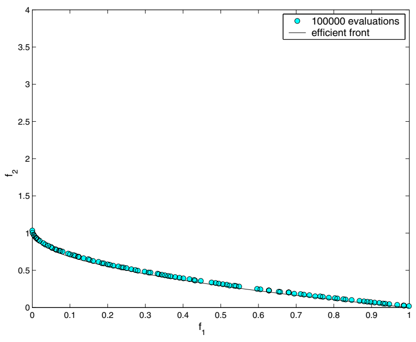


Fig. 9 Performance of NSGA-II on ZDT4.

Finally, we consider the 12-variable, three-objective DTLZ2 test problem having a spherical efficient front satisfying $f_1^2 + f_2^2 + f_3^2 = 1$ in the range $f_1, f_2 \in [0,1]$. Figure 10 shows the performance of the NC approach on this problem with a maximum of 100,000 function evaluations. Due to the increase in dimension more function evaluations are needed for each independent optimization task. With only 1,000 function evaluations allowed for each optimization (to find 100 solutions at the end), the algorithm was not able to reach the Pareto-optimal front (shown with a bounded surface) in most occasions. The solutions obtained using NSGA-II (with

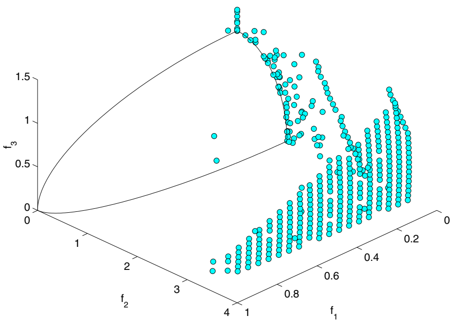


Fig. 10 Performance of NC method on DTLZ2.

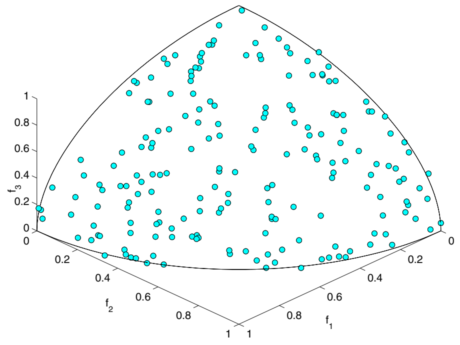


Fig. 11 Performance of NSGA-II on DTLZ2.

only 20,000 evaluations) are shown in Figure 11. Although the distribution is not quite as expected due to NSGA-II's 'quick-and-dirty' crowding distance operator, the obtained solutions spread across the entire front and are all very close to the true Pareto-optimal front. As pointed out elsewhere [24], a better niching operator than the crowding-distance operator, such as a *clustered* NSGA-II can employ a better distribution of solutions in problems having more than two objectives, as shown in Figure 12.

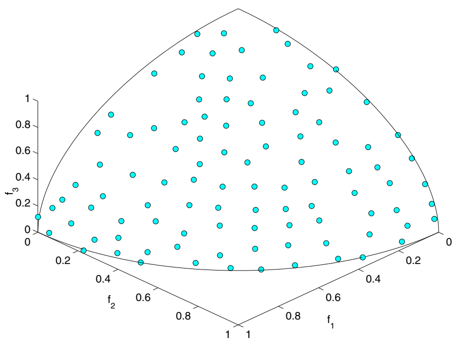


Fig. 12 Performance of clustered NSGA-II on DTLZ2.

From all these simulation results, we observe that in handling simpler problems, both classical scalarized generating methods and evolutionary multi-objective optimization methodologies perform well, but if the problem possesses complexities, EMO methodologies show an edge over the classical generating methods. A previous study [24] reported more comparative results in support of this argument.

3.2 Generating Preferred Pareto-optimal Solutions

Now, we address another task of multi-objective optimization in which the focus is to find a partial (and preferred) Pareto-optimal set. Although for two or three objectives such a task may not beneficial, particularly when EMO methodologies are able to find a well-distributed set of points on such problems, such biased search strategies may be computationally beneficial than finding the complete frontier, particularly when preference information is available.

One of the popular ways to find a preferred solution in the MCDM literature is to use the reference point method [26]. Given a reference point $\bar{\mathbf{z}}$ for an M-objective optimization problem of minimizing $(f_1(\mathbf{x}),\ldots,f_M(\mathbf{x}))$ with $\mathbf{x} \in S$, the following single-objective optimization problem is solved for this purpose:

$$\begin{array}{l} \text{Minimize } \max_{i=1}^{M} \left[w_i (f_i(\mathbf{x}) - \bar{z}_i) \right], \\ \text{Subject to } \mathbf{x} \in S. \end{array} \tag{3}$$

Here, w_i is the i-th component of a chosen weight vector used for scalarizing the objectives. Figure 13 illustrates the concept. For a chosen reference point, the closest Pareto-optimal solution (in the sense of the weighted-sum of the objectives) is the target solution to the reference point method. To make the procedure interactive and useful in practice, Wierzbicki [26] suggested a procedure in which the obtained solution $\mathbf{z}'$ is used to create M new reference points, as follows:

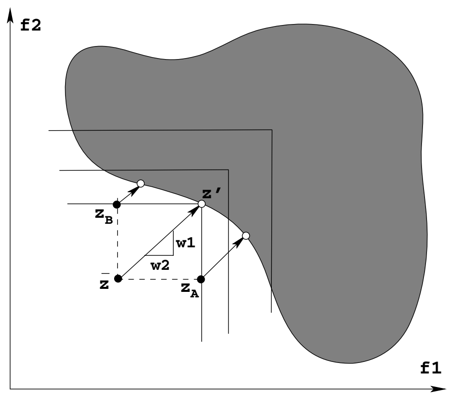


Fig. 13 Classical reference point approach.

$$\mathbf{z}^{(j)} = \bar{\mathbf{z}} + (\mathbf{z}' - \bar{\mathbf{z}}) \cdot \mathbf{e}^{(j)}, \tag{4}$$

where $\mathbf{e}^{(j)}$ is the j-th coordinate direction vector. For the two-objective problem shown in the figure, two such new reference points ($\mathbf{z}_A$ and $\mathbf{z}_B$) are also shown. New Pareto-optimal solutions are then found by forming new achievement scalarizing problems. If the decision-maker is not satisfied with any of these Pareto-optimal solutions, a new reference point is suggested and the above procedure is repeated. If viewed differently, the developer of the reference point method seem to represent a preference information through indicating the reference point in the objective space. The task is then to explore the Pareto-optimal region *close* to the supplied reference point.

An EMO methodology can be used directly for this purpose and EMO's population concept can be beneficial again in this task and instead of finding one preferred point, a number of preferred Pareto-optimal solutions can be found in a single simulation run. Deb and Sundar [11] suggested a reference point based NSGA-II approach for this purpose. EMO allows another advantage exploiting its population approach. Instead of finding a preferred set of Pareto-optimal solutions for a single reference point, multiple regions for different reference points can be captured in an EMO population simultaneously. Figure 14 shows the obtained solutions for five different reference points on test problems ZDT1. Instead of finding the complete Pareto-optimal front for the ZDT1 problem, the reference point based NSGA-II finds only a preferred set of points close to supplied reference points in a single simulation run.

The 11-variable DTLZ2 problem has a three-dimensional, non-convex, Pareto-optimal front. We use two reference points ($(0.2, 0.2, 0.6)^T$ and $(0.8, 0.6, 1.0)^T$) as shown in Figure 15. A good distribution of solutions near the two reference points is obtained. This indicates the ability of the proposed NSGA-II procedure in solving three-objective optimization problems as well.

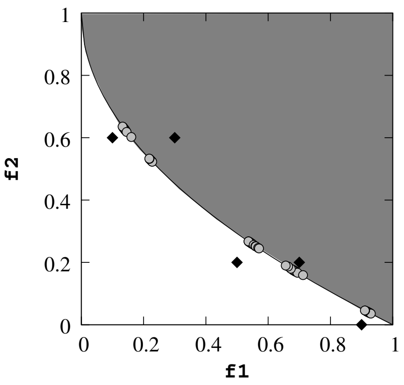


Fig. 14 Preferred solutions for five reference points on ZDT1.

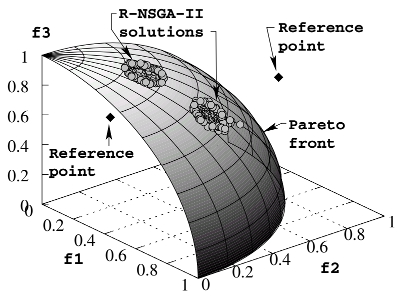


Fig. 15 Preferred solutions for two reference points on DTLZ2.

Recall that only way to find a set of preferred Pareto-optimal points using a point-by-point reference point approach would be to generate the points one after the other, as suggested by Wierzbicki [26]. For handling multiple reference points, the generating task simply has to be repeated as many times as the number of reference points. On the contrary, EMO's inherent parallel processing and population approach allows multiple preferred regions (not only points) to be found in a simulation run effectively.

4 EMO's Niches in Many Objectives

Since their inception, EMO methodologies have been mostly applied to two or three objective problems. Recent systematic studies on larger objectives have clearly indicated that EMO methodologies are not efficient in finding a representative set of solutions on the complete Pareto-optimal front for five or more objectives. At the first instance, this may sound disappointing, but there are a number of reasons for this behavior, and importantly EMO algorithms have been found to have their edge again in handling a large number of objectives for at least two different scenarios, which we shall discuss in this section.

With more conflicting objectives, the trade-off Pareto-optimal front, in general, becomes multi-dimensional and it requires exponentially more solutions to suitably represent the Pareto-optimal frontier. With more objectives, a large proportion of the initially random EMO population becomes non-dominated to each other, thereby not leaving enough room for new solutions to be created for the search to proceed towards the Pareto-optimal front. With more objectives, the Pareto-optimal front becomes difficult to visualize and analyze for an appropriate decision-making task to choose a single solution. Despite these difficulties in handling a large number of objectives, an EMO procedure can still be beneficial for the following two problem solving tasks.

4.1 Finding Preferred Solutions

EMO methodologies are found to be good candidates for finding a *preferred* set of near Pareto-optimal solutions in many-objective problems. In this section, we demonstrate their working on five or more objectives using various preference based multi-objective optimization concepts.

First, we revisit the reference point based NSGA-II approach described in Section 3.2. We first apply the procedure to the 14-variable, five-objective DTLZ2 problem. Two reference points are chosen as follows: (i) (0.5, 0.5, 0.5, 0.5, 0.5) and (ii) (0.2, 0.2, 0.2, 0.2, 0.8). Figure 16 shows the value-path plot of the five-objective solutions. It is clear that two distinct sets of solutions near the above reference points are obtained by the proposed procedure. Since the Pareto-optimal solutions in the DTLZ2 problem satisfy $\sum_{i=1}^{M} f_i^2$ equal to one, we compute this term for all obtained solutions and the values are found to lie within [1.000, 1.044] (at most 4.4% from one), thereby meaning that all solutions are very close to the true Pareto-optimal front. Solutions depicting almost identical objective values appear for the first

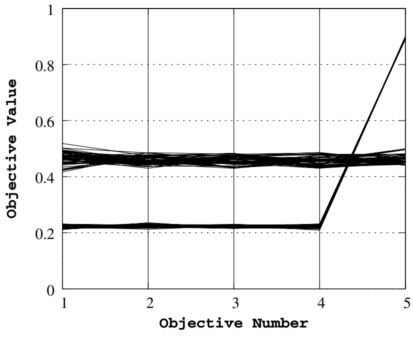


Fig. 16 Preferred solutions for two reference points with $\varepsilon = 0.01$ on five-objective DTLZ2.

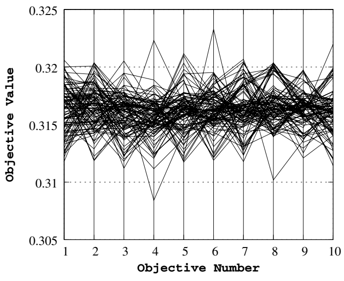


Fig. 17 Preferred solutions for one reference point with $\varepsilon = 0.01$ on 10-objective DTLZ2.

reference point which has identical objective values. For the second reference point, the fifth objective has a much larger value. The reference point based NSGA-II is able to find near Pareto-optimal solutions close to this reference point, as shown in the figure.

To demonstrate the scalability of the proposed procedure, we then solve the 19-variable, 10-objective DTLZ2 problem with one reference point: $f_i = 0.25$ for all $i = 1, 2, \ldots, 10$. The obtained distribution is shown in Figure 17. Although the objective values can vary in [0,1], the points concentrates near $f_i = 1/\sqrt{10}$ or 0.316, which turns out to be the region closest to the chosen reference point. When we compute $\sum_{i=1}^{10} f_i^2$ of all obtained solutions, they are found to be exactly equal to one, thereby meaning that all reference point based NSGA-II solutions are on the true Pareto-optimal front.

It is needless to say that if a generating classical method is used for this problem to complete the above tasks, many independent single-objective optimizations are needed. This study shows that despite EMO's generic difficulties in finding a representative set of solutions on the entire large-dimensional Pareto-optimal front, they have a niche in finding a set of preferred Pareto-optimal solutions – downplaying the dimensionality issue near the preferred Pareto-optimal region.

4.2 Light Beam Search Based EMO

The light beam search (LBS), as described in Jaszkiewicz and Slowinski [18], combines the reference point idea and tools of multi-attribute decision analysis (MADA). An aspiration and a reservation point are supplied by the DM. These two points determines the direction of the search in an iteration. Initially a non-dominated *middle* point is determined by projecting the aspiration point on to the non-dominated front by using an augmented version of Wierzbicki's scalarizing achievement function. Thereafter, a local preference model in the form of an out-ranking relation S is used to obtain neighboring solutions of the current non-dominated point, or the middle point. It is said that a out-ranks b (or aSb), if a is considered to be at least as good as b. To define out-ranking relation, DM has to specify three preference thresholds for each objective. They are *indifference threshold*, *preference threshold* and *veto threshold*. In the LBS procedure, they are assumed to provide only local information, thus they are assumed to be constants. Based on these values, the out-ranking relation finds a solution which is incomparable or indifferent to the middle point. The DM can control the search by either modifying the aspiration and/or reservation points, or by shifting the middle point to selected better point from its neighborhood or by modifying the preference threshold values.

With the above principle of the LBS procedure, Deb and Kumar [18] proposed an EMO methodology by which a set of Pareto-optimal solutions in the neighborhood of the middle point can be found using the modified out-ranking relation. In the original LBS method, the decision-maker has to specify three preference parameters for each objective, which is quite demanding on the part of the DM. In the LBS based EMO, authors have used only the veto preference parameter. Since the EMO approach deals with a population of solutions, the required points satisfying the out-ranking criterion in all the directions would be obtained in one simulation run.

Figure 18 shows the results of LBS based NSGA-II on the 14-variable, five-objective DTLZ2 problem. Ideal point $(0,0,0,0,0)$ and nadir point $(1,1,1,1,1)$ are used as aspiration and reservation points, respectively. The veto threshold of $v_j = 0.05$ for all objectives is used. Since the above settings constitute an equal importance to all objectives, the obtained solutions are concentrated near $f_j = \frac{1}{\sqrt{5}}$ or 0.447.When the quantity $\sum f_j^2$ is computed for each solution, they are found to lie between 1 and 1.016. It confirms that all the obtained solutions are very close to the true Pareto-optimal front. A good diverse set of solutions satisfying supplied veto threshold values are obtained.

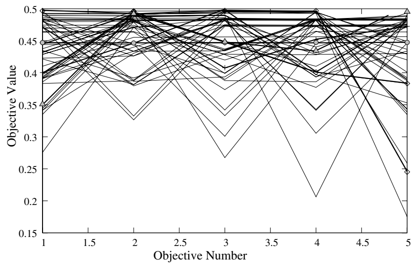


Fig. 18 LBS Based NSGA-II on five-objective DTLZ2 problem.

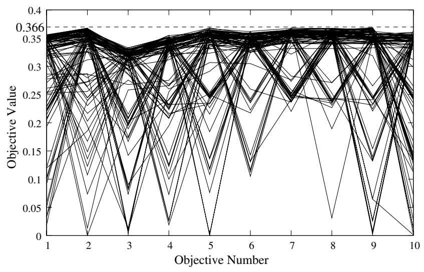


Fig. 19 LBS with NSGA-II on 10-objective DTLZ2 problem.

Next, the 19-variable, 10-objective DTLZ2 problem is solved with the same preference information as the one used above. Here also (Figure 19) the obtained solutions are concentrated near $f_j = \frac{1}{\sqrt{10}}$ or 0.316 and $\sum f_j^2$ lies between 1.006 and 1.010, thereby indicating that the obtained solutions are very close to the true Pareto-optimal front. The obtained middle point (z_j^c) is 0.316 for all $j = 1, \ldots, 10$.

4.3 *Eliminating Redundant Objectives*

In many real-world multi-objective optimization problems having a large number of objectives (say more than five), the resulting Pareto-optimal front turns out to be low-dimensional. This happens due to the correlations associated with some objectives for solutions close to the Pareto-optimal front. If the dimension of the Pareto-optimal front is two to four, EMO methodologies potentially find the complete front, despite the original problem having a large dimension. Recent studies [6, 23, 2] have suggested EMO based procedures for such large-objective problem solving tasks. Here, we reproduce one such strategy in which up to 50-objective problems are solved.

In the PCA-based NSGA-II procedure, the original problem is attempted to be solved using the standard NSGA-II procedure for a certain number of generations. The population is then analyzed using the principal component analysis (PCA) procedure to determine the most important objectives. The redundant objectives are then eliminated for further considerations. The NSGA-II procedure is continued with the non-redundant objectives and the same procedure is repeated. For details, interested readers may refer to the original study [6] and subsequent studies [23] involving non-linear PCA and kernel-based procedures.

On a 10-objective DTLZ5 problem [6, 10] having two-objective interactions on the Pareto-optimal front, the PCA-based NSGA-II procedure identifies four of the 10 objectives to be important in the first iteration: f_1, f_5, f_9 and f_{10}. Thereafter, in the second iteration f_1 was found to be redundant. In the subsequent iteration, f_5 was found redundant and no further reduction was observed with increased iterations, thereby identifying that the Pareto-optimal front of the original 10-objective DTLZ5 problem has only two-dimensional (involving objectives f_9 and f_{10}) interactions to Pareto-optimality. To further demonstrate the performance of the procedure, we present results on the 50-objective DTLZ5 problem having two-objective Pareto-optimal interactions (with 48 redundant objectives) in Table 2. Similarly Table 3 shows how the same PCA based NSGA-II approach can correctly identify the true three-objective combination on a 10-objective DTLZ5 problem in just three iterations.

Table 1 DTLZ5(2,10).

Iter. 1	f_1 f_5 f_9 f_{10}
Iter. 2	f_5 f_9 f_{10}
Iter. 3	f_9 f_{10}
Iter. 4	f_9 f_{10}

Table 2 DTLZ5(2,50).

Iter. 1	f_{20} f_{45} f_{50}
Iter. 2	f_{45} f_{50}
Iter. 3	f_{45} f_{50}

Table 3 DTLZ5(3,10).

Iter. 1	f_1 f_7 f_8 f_9 f_{10}
Iter. 2	f_8 f_9 f_{10}
Iter. 3	f_8 f_9 f_{10}

5 Other Niches of EMO

EMO's principle of finding multiple trade-off solutions has been found to be useful indirectly to other problem solving tasks. The list is long, but for brevity, we discuss two other salient problem solving tasks in which EMO methodologies have shown a definite edge.

5.1 Knowledge Discovery

EMO's ability to find a number of Pareto-optimal solutions simultaneously for two to four objectives can be exploited for a bigger cause which goes beyond simply finding the optimal solutions, often pursued in a multi-objective optimization study. Since EMO solutions are all expected to be near-optimal or optimal, these solutions can be analyzed for finding properties which are *common* to them. Such a procedure can then constitute a systematic approach in deciphering important and hidden properties which near-optimal (or high-performing) solutions must possess for a problem. In a number of practical problem-solving tasks, the so-called *innovization* procedure is shown to find important knowledge about high-performing solutions [8]. We illustrate the importance of innovization task through an electric motor design problem having five discrete variables and two conflicting objectives of maximizing power output and minimizing the size of a brushless DC motor [7]. Figure 20 shows the obtained Pareto-optimal front, every solution of which was found to have the identical values to four of the five variables: (i) a Y-type electric connection from two choices, (ii) a Y-type lamination from three choices, (iii) a 16-gauge wire from 16 different allowable gauges, and (iv) exactly 18 turns in the windings of the armature from 10 to 80 allowable turns. An analysis of the solutions revealed

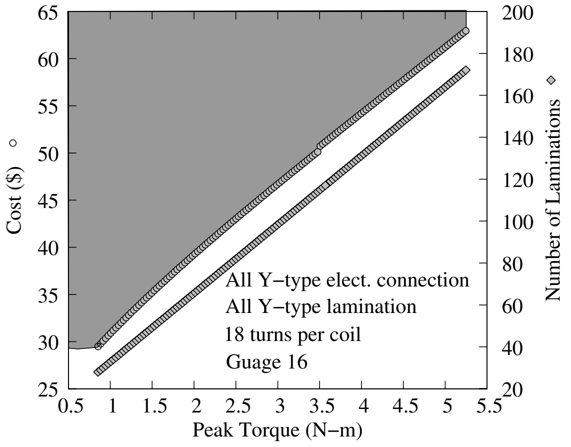


Fig. 20 Pareto-optimal front and innovative principles for the motor design problem.

that the only way an optimally designed motor differs from another motor having a different power rating is through a proportionate increase in the number of laminations, thereby implicating that a higher powered motor to appear longer through the addition of more laminations than by any other means. Not only does such a task innovate principles of designing high-performing solutions, the outcome can be highly beneficial for efficient production and operation of a plant. For example, the discovery of an identical gauge size for all optimally designed motors allows an efficient inventory to be maintained with only a single wire gauge; the fact that only 18 turns must be used in all optimally designed motors, a special purpose (and cost-effective) turning machine may be designed for a cost-efficient production system. Such useful properties are expected to exist in practical problems, as they follow certain scientific and engineering principles at the core, but finding them through a systematic scientific procedure had not been paid much attention in the past. The principle of first searching for multiple trade-off and high-performing solutions using a multi-objective optimization procedure (preferably using an EMO because of its niches compared to classical generating methods described in section 3.1) and then analyzing them to discover useful knowledge certainly remains a viable way forward. For this purpose, other post-optimality analyses of EMO solutions [22] may be useful. The EMO methodologies can thus enable innovative designs and bring out hidden principles associated with such designs which may lead to efficient operation – a matter which most successful industries are interested today.

5.2 Multiobjectivization

Interestingly, the act of finding multiple trade-off solutions using an EMO procedure has found its application outside the realm of solving multi-objective optimization problems. The concept of finding near-optimal trade-off solutions is applied to solve other kinds of optimization problems as well. For example, the EMO concept is used to solve constrained single-objective optimization problems by converting the task into a two-objective optimization task of additionally minimizing an aggregate constraint violation [3]. This eliminates the need to specify a penalty parameter while using a penalty based constraint handling procedure. If viewed this way, the usual penalty function approach used in classical optimization studies is a special weighted-sum approach to the bi-objective optimization problem of minimizing the objective function and minimizing the constraint violation, for which the weight vector is a function of the penalty parameter.

A well-known difficulty in genetic programming studies, called the 'bloating', arises due to the continual increase in size of genetic programs with each iteration. The reduction of bloating by minimizing the size of programs as an additional objective helped to find high-performing solutions with a smaller size of the code [1, 12].

Minimizing the intra-cluster distance and maximizing inter-cluster distance simultaneously in a bi-objective formulation of a clustering problem is found to yield better solutions than the usual single-objective minimization of the ratio of the intra-cluster distance to the inter-cluster distance [14].

An EMO is used to solve minimum spanning tree problem better than a single-objective EA [21]. A recent edited book [16] describes many such interesting applications in which EMO methodologies have helped to solve problems which are otherwise (or traditionally) not treated as multi-objective optimization problems.

6 Conclusions

The research and application in evolutionary multi-objective optimization (EMO) has a taken a fast ride since its inception in the early Nineties. In this chapter, we have touched upon a few interesting problem domains in which EMO has clearly shown its edge, in some occasions, over their classical counterparts and in other occasions as a natural choice. Much of their success has come from EMO's flexibility, population approach, and modularity which enable them to be applied in difficult problem domains and which enable them to capture a number of trade-off solutions.

EMO has been shown to reign in solving problems having a few objectives to a large number of objectives, in optimization tasks and in aiding decision-making tasks, and in multi-objective to other problem solving tasks. These studies on EMO challenge the imagination of researchers and practitioners in making them applicable and useful to other similar complex problem solving tasks which are currently put aside due to a lack of a suitable solution methodology.

Acknowledgment

The author is a Finland Distinguished Professor at the Helsinki School of Academics, supported by the Academy of Finland and the Foundation of Helsinki School of Economics.

References

[1] Bleuler, S., Brack, M., Zitzler, E.: Multiobjective genetic programming: Reducing bloat using spea2. In: Proceedings of the 2001 Congress on Evolutionary Computation, pp. 536–543 (2001)
[2] Brockhoff, D., Zitzler, E.: Dimensionality reduction in multiobjective optimization: The minimum objective subset problem. In: Operations Research Proceedings 2006, pp. 423–429 (2007)
[3] Coello, C.C.: Treating objectives as constraints for single objective optimization. Engineering Optimization 32(3), 275–308 (2000)
[4] Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley, Chichester (2001)
[5] Deb, K.: A robust evolutionary framework for multi-objective optimization. In: Proceedings of Genetic and Evolutionary Computation conference (GECCO 2008), pp. 633–640 (2008)
[6] Deb, K., Saxena, D.: Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In: Proceedings of the World Congress on Computational Intelligence (WCCI 2006), pp. 3352–3360 (2006)
[7] Deb, K., Sindhya, K.: Deciphering innovative principles for optimal electric brushless d.c. permanent magnet motor design. In: Proceedings of the World Congress on Computational Intelligence (WCCI 2008), pp. 2283–2290. IEEE Press, Piscatway (2008)
[8] Deb, K., Srinivasan, A.: Innovization: Innovating design principles through optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2006), pp. 1629–1636. The Association of Computing Machinery (ACM), New York (2006)
[9] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
[10] Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multi-objective optimization. In: Abraham, A., Jain, L., Goldberg, R. (eds.) Evolutionary Multiobjective Optimization, pp. 105–145. Springer-Verlag, London (2005)

[11] Deb, K., Sundar, J., Uday, N., Chaudhuri, S.: Reference point based multi-objective optimization using evolutionary algorithms. International Journal of Computational Intelligence Research (IJCIR) 2(6), 273–286 (2006)
[12] De Jong, E.D., Watson, R.A., Pollack, J.B.: Reducing bloat and promoting diversity using multi-objective methods. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001), pp. 11–18 (2001)
[13] Kung, H.T., Luccio, F., Preparata, F.P.: On finding the maxima of a set of vectors. Journal of the Association for Computing Machinery 22(4), 469–476 (1975)
[14] Handl, J., Knowles, J.: An evolutionary approach to multiobjective clustering. IEEE Transactions on Evolutionary Computation 11(1), 56–76 (2007)
[15] Holland, J.: Concerning efficient adaptive systems. In: Yovits, M., Jacobi, G., Goldstein, G. (eds.) Self-Organizing Systems, pp. 215–230. Spartan Press (1962)
[16] Knowles, J.D., Corne, D.W., Deb, K. (eds.): Multiobjective problem solving from nature. Springer Natural Computing Series. Springer, Heidelberg (2008)
[17] Jahn, J.: Vector optimization. Springer, Berlin (2004)
[18] Jaszkiewicz, A., Slowinski, R.: The light beam search approach – An overview of methodology an applications. European Journal of Operation Research 113, 300–314 (1999)
[19] Messac, A., Mattson, C.A.: Normal constraint method with guarantee of even representation of complete pareto frontier. AIAA Journal (in press)
[20] Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer, Boston (1999)
[21] Neumann, F., Wegener, I.: Minimum spanning trees made easier via multi-objective optimization. In: GECCO 2005: Proceedings of the 2005 conference on genetic and evolutionary computation, pp. 763–769. ACM, New York (2005)
[22] 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)
[23] Saxena, D.K., Deb, K.: Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: Employing correntropy and a novel maximum variance unfolding. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 772–787. Springer, Heidelberg (2007)
[24] Shukla, P., Deb, K.: Comparing classical generating methods with an evolutionary multi-objective optimization method. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 311–325. Springer, Heidelberg (2005)
[25] Sindhya, K., Deb, K., Miettinen, K.: A local search based evolutionary multi-objective optimization technique for fast and accurate convergence. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199. Springer, Heidelberg (2008)
[26] Wierzbicki, A.P.: The use of reference objectives in multiobjective optimization. In: Fandel, G., Gal, T. (eds.) Multiple Criteria Decision Making Theory and Applications, pp. 468–486. Springer, Berlin (1980)
[27] Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation Journal 8(2), 125–148 (2000)

Applications of Parallel Platforms and Models in Evolutionary Multi-Objective Optimization

Antonio López Jaimes and Carlos A. Coello Coello*

Abstract This chapter presents a review of modern parallel platforms and the way in which they can be exploited to implement parallel multi-objective evolutionary algorithms. Regarding parallel platforms, a special emphasis is given to global metacomputing which is an emerging form of parallel computing with promising applications in evolutionary (both multi- and single-objective) optimization. In addition, we present the well-known models to parallelize evolutionary algorithms (i.e., master-slave, island, diffusion and hybrid models) describing some possible strategies to incorporate these models in the context of multi-objective optimization. Since an important concern in parallel computing is performance assessment, the chapter also presents how to apply parallel performance measures in multi-objective evolutionary algorithms taking into consideration their stochastic nature. Finally, we present a selection of current parallel multi-objective evolutionary algorithms that integrate novel strategies to address multi-objective issues.

1 Introduction

Problems having two or more (normally conflicting) objectives naturally arise in a variety of disciplines. Such problems, which are called "multi-objective" have not one, but a set of solutions, which are collectively known as the "Pareto optimal set". All the solutions contained in the Pareto optimal set

CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Computación
Av. IPN No. 2508, Col. San Pedro Zacatenco
México, D.F. 07360, México
email: `tonio.jaimes@gmail.com`, `ccoello@cs.cinvestav.mx`

* The second author is also associated to UMI-LAFMIA 3175 CNRS.

are equally good, and represent the best possible trade-offs among all the objectives.

The use of evolutionary algorithms (EAs) for solving multi-objective optimization problems (MOPs) has been quite popular in the last few years. The rising popularity of multi-objective evolutionary algorithms (MOEAs) is mainly due to their flexibility and ease of use with respect to traditional mathematical programming techniques [16, 21, 51, 58, 17]. However, a wide variety of real-world MOPs are highly demanding in terms of CPU time (e.g., in aeronautical engineering [49]). This may limit the applicability of MOEAs, since they normally require a considerably large number of objective function evaluations to achieve reasonably good results. The use of parallelism is an obvious choice to solve these problems in a reasonable amount of time [62]. Besides the time reduction that they can achieve, parallel MOEAs (pMOEAs) are attractive for many other reasons: (1) they can use more memory to cope with more difficult problems, (2) they allow the use of larger population sizes, (3) they tend to improve the population's diversity, (4) they reduce the probability of finding suboptimal solutions, (5) and they can cooperate in parallel with other search techniques (including non-evolutionary techniques).

During the last three decades, parallel Evolutionary Algorithms used for global optimization (pEAs) have been widely studied [5, 15, 54, 61]. However, due to the peculiarities of multi-objective optimization there are some issues that require the use of novel approaches. From these issues, the most relevant is the fact that the evaluation of each particular solution to a MOP implies the evaluation of k ($k \geq 2$) objective functions. This implies a much higher computational cost and, therefore, motivates the need of parallelizing a MOEA. Additionally, real-world MOPs tend to have high-dimensionality (i.e., a large number of decision variables) which also normally requires a much higher computational cost in order to find a reasonably good approximation of the Pareto optimal set. Finally, the use of archiving, clustering or niching techniques (which are commonly adopted with MOEAs [38, 60, 22]) also adds to the computational overhead of a MOEA, which is one more reason to justify their parallelization.

The present chapter provides an overview of the models employed to implement parallel MOEAs and discusses some implementation issues that deserve to be considered in the light of multi-objective optimization. Additionally, we present a review of the parallel architectures currently available to implement pMOEAs. Special emphasis is given to global metacomputing which is a new form of parallel computing that allows us to share computing resources in order to build a large networked metacomputer. The chapter also presents some of the parallel performance metrics that have been adopted to assess pMOEA's efficiency as well as a short discussion regarding the conditions under which one may find superunitary speedups. We also present the concept of fixed-time speedup, which is an alternative measure of speedup that has some advantages over other forms of speedup usually adopted in the current literature.

The remainder structure of the chapter is the following. Some basic concepts related to multi-objective optimization are provided in Section 2. Section 3 describes a taxonomy of parallel architectures, emphasizing MIMD architectures (Multiple Instruction Stream, Multiple Data Stream), since they are the most common parallel platform in current use. In Section 4, we present the most commonly used pMOEA parallel models. In turn, the most popular parallel performance measures in current use are introduced in Section 5. Finally, Section 6 describes a selection of pMOEAs that incorporate novel parallel strategies. The reader can find the description of other pMOEAs in the overview presented by Talbi et al. [57].

2 Basic Concepts

We are interested in solving problems of the type:

$$\text{Find } \mathbf{x} \text{ wich optimizes } \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})] \tag{1}$$

subject to:

$$g_i(\mathbf{x}) \leq 0 \quad i = 1, 2, \ldots, n \tag{2}$$

$$h_i(\mathbf{x}) = 0 \quad i = 1, 2, \ldots, p \tag{3}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables, f_i, $i = 1, ..., k$ are the objective functions and g_i, h_j, $i = 1, ..., m$, $j = 1, ..., p$ are the constraint functions of the problem.

In multi-objective optimization problems the aim is to find good compromises (trade-offs). To understand the concept of optimality, we will introduce first a few definitions.

Definition 1. Given two vectors $\mathbf{x}, \mathbf{y} \in \mathrm{I\!R}^k$, we say that $\mathbf{x} \leq \mathbf{y}$ if $x_i \leq y_i$ for $i = 1, ..., k$, and that $\mathbf{x}$ **dominates** $\mathbf{y}$ (denoted by $\mathbf{x} \prec \mathbf{y}$) if $\mathbf{x} \leq \mathbf{y}$ and $\mathbf{x} \neq \mathbf{y}$.

Definition 2. We say that a vector of decision variables $\mathbf{x} \in \mathcal{X} \subset \mathrm{I\!R}^n$ is **nondominated** with respect to $\mathcal{X}$, if there does not exist another $\mathbf{x}' \in \mathcal{X}$ such that $\mathbf{f}(\mathbf{x}') \prec \mathbf{f}(\mathbf{x})$.

Definition 3. We say that a vector of decision variables $\mathbf{x}^* \in \mathcal{F} \subset \mathrm{I\!R}^n$ ($\mathcal{F}$ is the feasible region) is **Pareto optimal** if it is nondominated with respect to $\mathcal{F}$.

This is a notion of optimality that was originally proposed by Francis Ysidro Edgeworth [26] and later generalized by Vilfredo Pareto [52]. Although some authors call this notion *Edgeworth-Pareto optimality* (see for example [55]), the term *Pareto optimality* is the most common and widespread, and is, therefore, the one used in this chapter.

Definition 4. The **Pareto Optimal Set** $\mathcal{P}^*$ is defined by:

$$\mathcal{P}^* = \{\mathbf{x} \in \mathcal{F} | \mathbf{x} \text{ is Pareto optimal}\}$$

Definition 5. The **Pareto Front** $\mathcal{PF}^*$ is defined by:

$$\mathcal{PF}^* = \{\mathbf{f}(\mathbf{x}) \in \mathbb{R}^k | \mathbf{x} \in \mathcal{P}^*\}$$

We thus wish to determine the Pareto optimal set from the set $\mathcal{F}$ of all the decision variable vectors that satisfy (2) and (3).

3 Parallel Architectures

Many schemes to classify parallel computers [28, 10, 59, 36] have been proposed so far. However, none of them has become standard in the specialized literature. The difference among these schemes lies on the characteristics of the parallel system that are taken into account, namely: the organization of the address space, the interconnection network, or the processors' granularity. For our purposes we will use the well-known Flynn's taxonomy. We will then complement the MIMD classification following a similar approach as in Johnson [36] or Bell [10].

SISD (Single Instruction Stream, Single Data Stream). This class of computers represents the conventional single-processor von Neumann computers, where only one instruction is executed at the same time over a unique data element.

SIMD (Single Instruction Stream, Multiple Data Stream). This architecture consists of a central instruction unit that broadcasts a single instruction to a group of slave processors which apply the instructions in a synchronized fashion on different pieces of data stored in its own memory. Pipeline vector processors and processor array computers are considered as specialized SIMD architectures.

MISD (Multiple Instruction Stream, Single Data Stream). A computer of this class is able to apply different instructions to the same stream of data at a time. Although some authors consider that this architecture is mainly theoretical because there is no practical computer that fits this model [31, 25, 59], other authors have considered high-level pipeline computer instances of this model. Likewise, we can include fault-tolerant computers in the MISD class since in these systems several processors are applied upon the same data using different programs in order to mask a possible fault in one of the programs.

MIMD (Multiple Instruction Stream, Multiple Data Stream). In a MIMD architecture, a group of processors independently execute different instruction streams over different data sets. This architecture is intended to support parallel applications that require processors to operate autonomously during most of the time.

3.1 Taxonomy of MIMD Computers

Although SIMD computers were very popular in the past (e.g., Connection Machines [34], MasPar computers [12], and Cray computers [8], which were all developed during the 1980s and early 1990s), nowadays manufacturers have moved toward MIMD architectures (clusters and constellations) (see e.g., [11, 47]), and SIMD computers are only used in specialized application domains such as image processing. To illustrate this it suffices to consider that in 1993, vector computers comprised about 74% of the 500 top supercomputers[1], while MIMD computers comprised 26%. By 2008, the situation had drastically changed inasmuch as vector computers comprise only 0.4% and MIMD computers comprise the remaining 99.6%. In the light of this trend in parallel high-performance architectures, MIMD systems are further discussed in the following paragraphs.

According to the organization of the address space and the connection method between memory and processors, MIMD computers are classified into two categories: shared memory MIMD systems, commonly known as *multiprocessors*, and distributed memory systems, usually referred to as *multicomputers* (see Figure 3).

Multiprocessor MIMD Systems

In this class of systems all the processors read and write to a common physical address space through an interconnection network (see Figure 1). Processors communicate each other by writing information on the global memory so that the other processors can read it. A drawback of this communication scheme is that data integrity is endangered since multiple processors can concurrently request a write operation on the same memory location. In order to avoid

[1] The Top500 project maintains a worldwide classification and statistics of the most powerful computers in accordance to the LINPACK benchmark (www.top500.org).

write conflicts the programmer has to use classic synchronization methods such as semaphores, locks and barriers. In order to improve efficiency, each processor is equipped with a cache memory to speedup access to frequently used data. However, the incorporation of cache memories motivates the need of protocols to ensure the coherence between global and cache memory. The most common form of multiprocessors are known as *Symmetric Multiprocessors* (SMP), which are computers constituted by multiple processors where the memory access time to any region of the shared memory is approximately the same for each processor. On the contrary, in a *Non-Uniform Memory Access* (NUMA) system, the shared memory is partitioned in such a way that each partition is associated to a different group of processors. As a result, the access time depends on which memory location is accessed.

Multicomputer MIMD Systems

In these systems, each processor has its own local memory. In this case, an interconnection network is used to allow communication among processors by means of message passing. In this category, we can find one of the most common parallel systems in current use, the *cluster of computers*. A cluster is a system comprised of independent computers (nodes) connected by a low-latency network. The nodes in a cluster are capable of independent operation and communicate with one another via message passing. Clusters comprise two classes: *clusters of workstations* (COWs)[2] and *constellation systems*. Both classes of clusters use both commercial off-the-shelf networks and computing nodes. Nevertheless, according to Dongarra et al. [24], the difference between these systems relies on the number of nodes connected by the network and the number of processors on each node. In a COW system, there are more nodes than processors in any of its nodes, while a constellation system has more processors than nodes in the clusters. Usually, each node in a constellation is configured as a SMP with hundreds of processors. Nowadays, a cluster with a low-latency proprietary network and more than 1000 processors is considered a *massively parallel processing system* (MPP)[3].

Global metacomputing

This is an emerging form of MIMD parallel systems that has attracted a great interest in the last decade. The basic idea of *global metacomputing* is

[2] Also known as Beowulf clusters or network of workstations (NOWs).

[3] MPP is a loosely-defined term that has been used to qualify different parallel architectures through the years. For instance, the first MPP system was a SIMD computer. However the current usage is mainly intended for distributed-memory systems.

employing computers geographically distributed around the world as if they were one large parallel machine or metacomputer. Although this concept dates back to the mid 1960's [64, 41] the limitations both in storage capacity and network performance in those days made the idea impractical.

The most ambitious form of metacomputing is named *grid computing*. Grid computing is a group of technologies and infrastructure that coordinate large-scale resource sharing among individuals or organizations around the world [29, 30]. Although, currently, the most common shared resources are computer power (e.g., clusters, constellations) and data (e.g., software, databases), grid computing is not limited to computational resources. Grid infrastructure also enables organizations to share: (1) scientific instruments such as telescopes, particle accelerators and electron microscopes, (2) analysis procedures and computational results, and (3) human expertise in the form of collaborative work.

Volunteer computing is another form of metacomputing mainly focused on facilitating the sharing of computer power and data storage among individuals around the world. The idea behind volunteer computing is to provide an infrastructure that enables individuals to share the idle processing power of their computers to build a large parallel machine which is used to solve expensive computational problems [53]. Among the most successful volunteer computing projects, we can find the distributed.net project [1] that solved the RSA RC-56 decryption challenge, and the SETI@home [2] project that analyzes radio telescope data looking for signs of extraterrestrial intelligence. As of 2008, SETI@home accounted with nearly 890 000 registered volunteers supplying an average of 460Tflops[4]. In order to put this figure in its proper context, it is worth considering that the fastest supercomputer currently available operates at 1026 Tflops the second fastest operates at 478 Tflops, which is comparable with SETI@home's throughput. Although these projects are categorized as "true volunteer" computing since users are unpaid and unrelated to the project administrators, there are other variations of this scheme, including private volunteer computing. That is, organizations such as companies, universities or laboratories which turn their existing networks into virtual supercomputers that they can use for their research. For a complete taxonomy of volunteer computing systems interested readers are referred to [53]. Problems that can be modelled using the master-slave paradigm are ideal to be solved with a volunteer computing system.

4 Parallelization Models of MOEAs

The parallelization schemes that have been proposed for MOEAs are derived from the well-known models designed for single-objective optimization: the

[4] This information was taken from the web page of BOINC (http://boincstats.com), the middleware used by SETI@home and other volunteer computing projects.

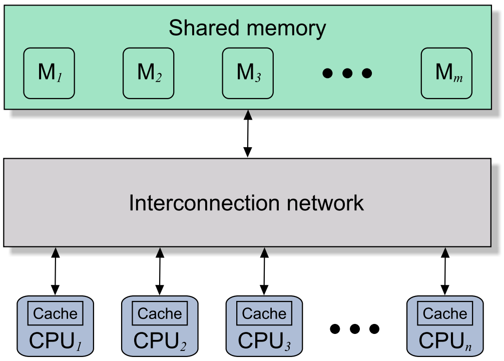


Fig. 1 Multiprocessor MIMD system (shared memory).

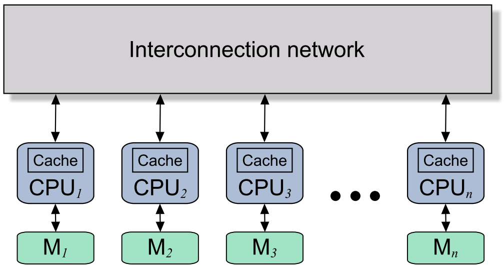


Fig. 2 Multicomputer MIMD system (distributed memory).

master-slave model, the island model, the diffusion model and the hybrid model. Nevertheless, currently there is no standard way to extend these models to the multi-objective field. Van Veldhuizen, Zydallis and Lamont [62] provide a detailed discussion of the generic versions of these models when applied to MOEAs. Next, we will briefly discuss each of them.

4.1 Master-Slave Model

The master-slave model is one of the simplest ways to parallelize a MOEA and, hence, the most popular among practitioners. Here, a master processor executes the MOEA, and the objective function evaluations are distributed among a number of slave processors. As soon as the slaves complete the evaluations they return the objective function values to the master and remain idle until the next generation.[5] In addition to the evolutionary operators (selection, recombination and mutation), the master processor executes other

[5] A scheme where the master also evaluates some individuals is possible, though.

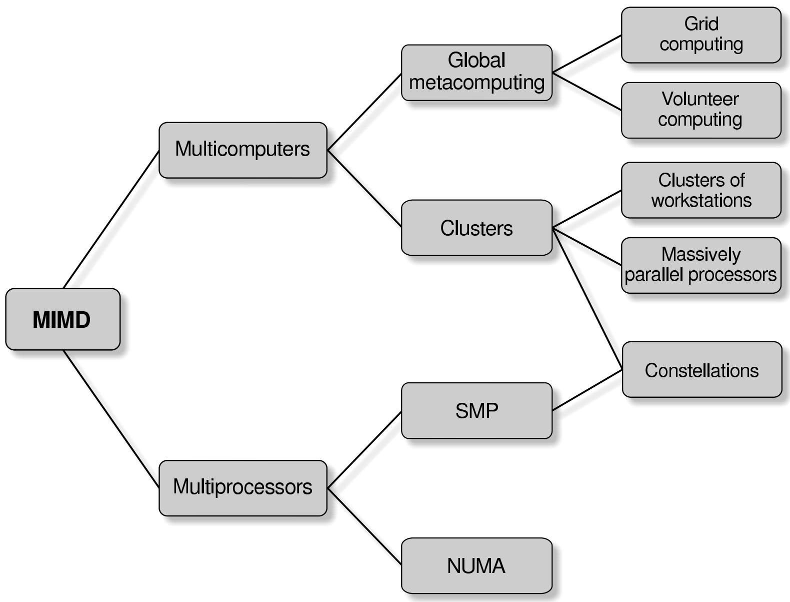


Fig. 3 MIMD taxonomy.

tasks such as Pareto ranking, and archiving. This model is depicted in Figure 4. A master-slave pMOEA explores the search space in the same way as a serial MOEA does. Therefore, it finds the same solutions found by its serial counterpart. However, the execution time is reduced (ideally p times with p processors). The master-slave model is perfect to be implemented in a volunteer computing system. However, as we indicate in the following lines, the evaluation time of the objective functions should be greater compared to the communication time.

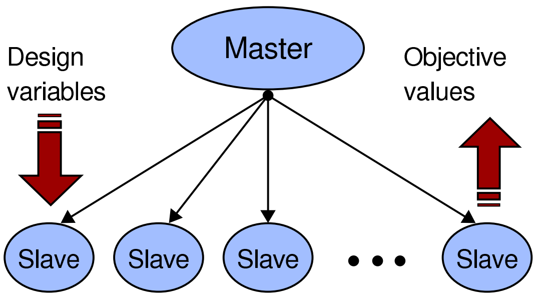


Fig. 4 Master-slave model.

In multi-objective optimization, there are three schemes to distribute the objective functions evaluations among a number of slave processors:

1. Evenly distribute the population over the slaves in such a way that each slave evaluates all the objective functions for its share of the population. Assuming that the master processor evaluates a portion of the individuals, the total computation time for each generation using P processors is given by

$$T_P = (P-1)t_{cp} + t_{ca} + \frac{nt_F}{P}, \tag{4}$$

   where t_{cp} is the time required to send one individual, t_{ca} is the time required to send back the objective values to the master, t_F is the time required to evaluate one individual, and n is the size of the population. If $t_{cp} = t_{ca}$ then we obtain the computation time presented by Cantú-Paz [15]:

$$T_P = Pt_{cp} + \frac{nt_F}{P}, \tag{5}$$

   From Equation (5) we can derive some interesting results. First, the optimal number of processors, P^*, that minimizes T_P is given by

$$P^* = \sqrt{n\gamma}, \tag{6}$$

   where $\gamma = \frac{t_F}{t_{cp}}$. Using the parallel execution time of Equation (5) and the execution time of a sequential MOEA, $T_S = nt_F$, we can ensure that a master-slave MOEA is faster than its sequential counterpart using the following condition:

$$\gamma > \frac{P^2}{n(P-1)}. \tag{7}$$

   Limited by Amdahl's law [7], the speedup of a master-slave MOEA has a maximum value. According to Cantú-Paz [15] the speedup curve is limited by

$$S_P^* = \frac{1}{2}P^*. \tag{8}$$

2. Each objective function is assigned to a different even partition of slaves (i.e., partition $\mathcal{P}_1$ evaluates f_1, $\mathcal{P}_2$ evaluates f_2 and so forth). Each partition of slaves then evenly distributes the population over their processors. Here, we are assuming that the size of each partition, $\mathcal{P}_i$, of slaves is $\frac{P}{k}$, where k is the number of objectives. If t_c is the time required to broadcast the entire population and $t_{f_{max}} = \max_{1\leq i\leq k}\{t_{f_i}\}$ is the execution time of the most expensive objective function in the set of objective functions, the execution time for each generation using this distribution scheme is given by

$$T_P = t_c + (P-1)t_{ca} + \frac{knt_{f_{max}}}{P}. \tag{9}$$

3. Each objective function is decomposed into smaller algebraic terms and then these terms are distributed to different groups of slaves. This way, each group of slaves evaluates a small part of a complex objective function. Let t_{com} be the time required to combine the decomposed objective values and $t_{f_{max}} = \max_{1\le i\le k,\, 1\le j\le p_i}\{t_{f_{i,j}}\}$ the execution time of the most expensive partition of the objective function f_i (p_i is the number of partitions of function f_i). Then, the execution time of one generation of this distribution scheme is

$$T_P = t_{com} + t_c + (P-1)t_{ca} + \frac{knt_{f_{max}}}{P}. \qquad (10)$$

4.2 Diffusion Model

Like the master-slave model, the diffusion model considers a unique population, but in this case the population is spatially distributed onto a neighborhood structure. Usually, this structure is a two-dimensional rectangular grid, and there is one individual per grid point (see Figure 5). Ideally, there is one processor per individual, and therefore this model is sometimes called *fine-grained*. This kind of pMOEA is also known as a *cellular* pMOEA because the model is similar to a cellular automaton with stochastic transition rules. The selection and mating is confined to a small neighborhood around each individual. The neighborhoods are overlapped (as depicted by the dotted lines in Figure 5) so that the good traits are spread or "diffused" throughout the whole population. The communication costs tend to be high, since the individuals who take part in the selection are distributed among several processors. As a consequence, this model is appropriate for shared-memory MIMD computers such as SMPs. However, custom hardware implementations on SIMD computers are also possible [27].

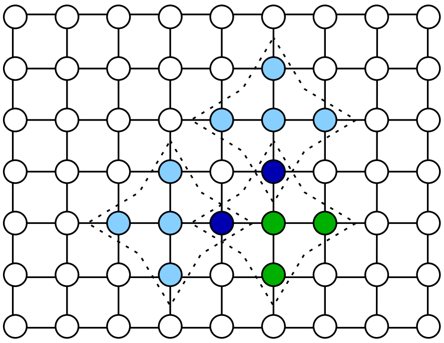


Fig. 5 Diffusion model.

4.3 Island Model

This model was inspired by the natural phenomenon in which a number of spatially isolated populations are linked together by dispersal and migration. In an island pMOEA, the population is divided into several small sub-populations, called *islands* or *demes*, which evolve independently of each other. In each of these islands a serial MOEA is executed for a number of generations called an *epoch*. At the end of each epoch, individuals migrate between neighboring islands. The neighbors are defined by the *migration topology*, which determines the migration paths along which individuals can move to other islands. A typical representation of the island model is shown in Figure 6, in which a ring topology is adopted, although other migration topologies are possible (2-D and 3-D meshes, tori, hypercubes or trees). Island pMOEAs are also known as *distributed* pMOEAs, as they are usually implemented on distributed memory MIMD computers. In particular, due to the low inter-processor communication frequency, this model is well-suited for clusters of computers or for grid computing systems.

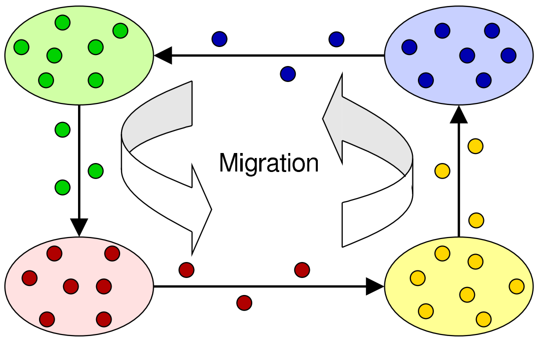


Fig. 6 Master-slave model.

This model is very popular among researchers, but it requires many parameters and design decisions. The main issues to consider with this sort of model include the migration topology, the migration frequency, the number of individuals to migrate, and the decision regarding the individuals which will migrate and those which will be replaced by the immigrants. Tables 1 and 2 present a number of possible migration and replacement schemes proposed by Coello, Lamont and Van Veldhuizen [16].

The model allows each island to have their own parameter setting. Depending on the homogeneity of the islands, we can recognize four variants of the island model:

Scheme	Description
Non-uniform schemes	
Elitist (random)	Migrate a random sample of individuals from the current nondominated front.
Elitist (niching)	Migrate individuals evenly distributed from the current nondominated front.
Elitist (front)	Migrate the entire nondominated front.
Uniform schemes	
Random	Migrate n individuals selected at random.
Elitist (random)	Migrate n individuals selected at random from the current nondominated front plus some individuals randomly selected from the ranked Pareto fronts if necessary.
Elitist (niching)	Migrate n individuals evenly distributed from the current nondominated front plus some individuals evenly distributed from the remainder ranked Pareto fronts if necessary.

Table 1 Migration schemes in the island model.

Scheme	Description
Random	Randomly replace n individuals.
None	No replacement, thereby the population increases its size.
Elitist (random)	Maintain the current nondominated front and randomly replace any dominated individual.
Elitist (ranking)	Rank the population into nondominated fronts and replace individuals from the worst ranked fronts with the immigrants.
Elitist (100% ranking)	Combine immigrants with the current population, rank the combined population and discard individuals from the worst ranked fronts.

Table 2 Replacement schemes for the island model.

1. **Island pMOEA with homogeneous nodes**. The MOEAs performed in every island have all the same parameter values (e.g., population size, mutation, crossover and migration rate).
2. **Island pMOEA with heterogeneous nodes**. Each island applies a MOEA which has a different parameter setting, uses its own evolutionary operators and solution encoding technique. Even more, each island can be constituted by a different MOEA.
3. **Island pMOEA with different objective subsets**. Each island is responsible for optimizing a different partition of the entire objective subset.
4. **Island pMOEA with different regions in the search space**. Each island may be explicitly instructed to explore a particular region of decision variable or objective function space to optimize computational resources.

There are a number of approaches [40, 63] that have successfully used the heterogeneous scheme adopting different MOEAs in each island. Whereas in León, Miranda and Segura [40] only MOEAs such as NSGA-II and SPEA2 are employed, it is possible to combine different metaheuristics as in the approach proposed by Vrugt and Robinson [63]. In this approach the authors combine four different metaheuristics, namely: a multiobjective evolutionary algorithm, a particle swarm optimization algorithm, an adaptive Metropolis search technique and a differential evolution algorithm. The idea behind a heterogeneous approach is to build a robust algorithm in which some metaheuristics can compensate the weaknesses of other metaheuristics in a particular problem. The success of a heterogeneous approach greatly depends on the strategy adopted to evaluate the performance of each metaheuristic in order to favor the best metaheuristics in future generations.

In the third variant, the straightforward strategy is to assign a different objective function to a set of islands executing, thereby, a single objective evolutionary optimization algorithm [50]. On the other hand, if the optimization problem has a large number of objectives, then it is possible to partition the objective set into groups with more than one objective. In this situation, we have to carefully decide how to group objective functions. As recent studies [14, 45] have pointed out, the conflict among objectives determines the importance of each objective in the optimization problem. In addition, there are different degrees of conflict among the objectives and it is possible that one objective that is not in conflict with a certain objective might be in great conflict with another. Therefore, we can cluster objectives according to the degree of conflict among them. That is, the objective set should be partitioned in such a way that the conflict among objectives inside each cluster is maximum and the conflict among objectives in different clusters is minimum.

In order to avoid that two or more demes exploit the same region of the search space it is convenient to instruct each deme to solve non-overlapping regions of decision variable or objective function space. In a general MOP it is very hard to devise a priori a distribution such that: (1) covers the entire search space, (2) assigns regions of equal size, and (3) aggregates a minimum complexity to constraint demes to their assigned region.

In Coello, Lamont and Van Veldhuizen [16], three strategies to distribute the search space among the demes are identified:

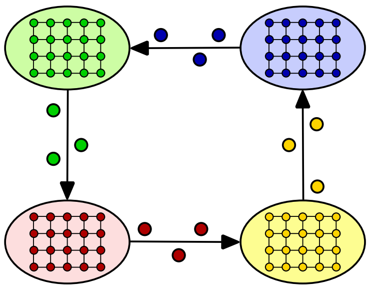
1. Constrain each deme to a particular region and force it to generate individuals until a suitable number of them is produced within its assigned region. The drawback of this strategy is that it introduces an overhead because of the extra number of objective function evaluations performed.
2. Constrain each deme to a particular region by migrating individuals to the deme covering such region. This strategy introduces a communication overhead, though.
3. Constrain each deme to a particular region of the Pareto front by introducing a bias in the search in such a way that each deme concentrates on a specific region of the Pareto front. The drawback of this approach is that the shape of the true Pareto front needs to be known a priori. Section 6 presents two pMOEAs proposed by Deb, Zope and Jain [23], and by Streichert, Ulmer and Zell [56] that follow this approach.

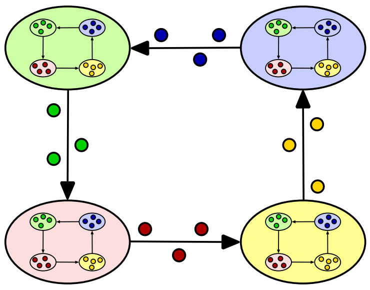
4.4 Hybrid Models

Another option to parallelize a MOEA is combining a coarse-grained parallel scheme at a high level (e.g., island model) with a fine-grained scheme at a low level (e.g., diffusion model). Cantú Paz discussed three types of hybrid schemes that use an island model at the high level:

1. Each island implements a diffusion pMOEA (see Figure 7(a)),
2. Each island implements a master-slave pMOEA (see Figure 7(c)) and
3. Each island implements an island pMOEA (see Figure 7(b)).

The first hybrid is ideal for a constellation of computers where each SMP node can be used to implement the diffusion pMOEA and the entire cluster of SMPs can be configured as the island model.


(a) Island/diffusion


(b) Island/island

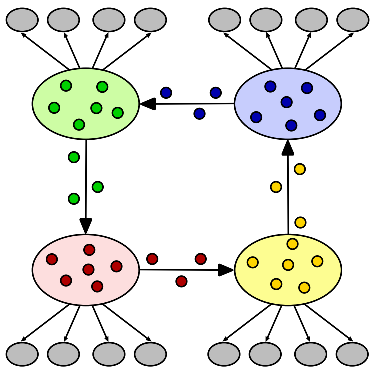

(c) Island/master-slave

Fig. 7 Hybrid models.

5 Performance Assessment of Parallel MOEAs

Parallel performance indicators provide useful information that can be used to identify the bottlenecks of the pMOEA or to predict its performance on a given parallel system. For instance, researchers can use these indicators to improve a parallel MOEA and a practitioner can decide if extending a given parallel system is cost-effective according to an analysis of the performance results.

5.1 Speedup

The most well-known measure of performance of a parallel system is the *speedup*, which is defined as the ratio of the total execution time on an uniprocessor to the total execution time in the parallel system. In other words, the speedup of a parallel algorithm executed in p processors is given by

$$S_p = \frac{T_1}{T_p},$$

where T_1 is the wall-clock time of the sequential algorithm and T_p is the wall-clock time of the parallel algorithm. Usually, T_1 should correspond to the optimal sequential algorithm (this speedup is denoted as *strong speedup* by Alba [4]). Nevertheless, as Helmbold and McDowell [33] point out, this requirement is impractical since even the best known sequential algorithm may be improved in the future and it may exhibit a different performance depending on the data of interest.

Alba [4] suggests different approaches to compute the speedup in evolutionary algorithms. First of all, given the stochastic nature of evolutionary algorithms, the execution times of the algorithms should involve the computation of average times over several runs. In fact, some authors [18] suggest to measure average execution times even for deterministic algorithms, since there are some random events in a parallel system such as the execution of system daemons or the reordenation of memory accesses, which may produce time variations from one execution to another. Additionally, it is recommended to compute the variance of the running times since a high variance may be a symptom of high contention for locks, for instance. Alba defines a type of speedup as the most appropriate for evaluating parallel evolutionary algorithms, namely *speedup with solution stop*. In this type of speedup, instead of using the number of evaluations as the stopping criterion, the "quality of the solution" is employed. In single-objective optimization, it is straightforward to determine if two solutions have the same quality, i.e., if they have the same fitness. In contrast, in multi-objective optimization, there is no consensus regarding how to determine, in general, the quality of the Pareto front generated by a MOEA. A practical approach is to use a unary performance indicator that assesses both the distribution and the convergence of the approximation of the Pareto front produced. Some indicators that might be adopted to determine the quality of the solution are: hypervolume [66], inverted generational distance [16] or the G-metric [43]. There are two variants to compute the *speedup with solution stop*: (1) compare the pMOEA against a canonical sequential MOEA with a global population, and (2) compare the parallel p-processor MOEA on a single processor and the same algorithm on p processors.

In addition to these types of speedup, we can use a similar approach to that adopted by the so-called fixed-time model [32], where a predefined time limit is used and the "work" increases as processors are added. In our case, the work is defined as the approximation to the true Pareto front measured with a suitable quality indicator. Thus, the *fixed-time speedup* is defined by

$$Sx_p = \frac{I(A_1)}{I(A_p)},$$

where I is a unary quality indicator, A_p is the approximation set obtained by the pMOEA and A_1 is the approximation set obtained by the sequential MOEA using some fixed-time window in both algorithms. As in the previous types of speedup, it is advised to use the p-processor MOEA on a single processor as the sequential algorithm. This type of speedup has some advantages over the *speedup with solution stop*. First, it is easier to incorporate the fixed-time stopping criterion to an existing code. Second, the same output datasets obtained in several runs can be used to compute the fixed-time speedup with different quality indicators. In contrast with the *speedup with solution stop*, we have to execute the algorithms again if we change the quality indicator used as our stopping criterion.

Superunitary speedup

The condition when the speedup with p processors is greater than p is referred to as *superunitary speedup* (also known as superlinear speedup). Although in the past many works on parallel evolutionary algorithms have reported superunitary speedup [42, 9, 6], in some cases this phenomenon is due to a comparison against an inefficient or ineffective sequential algorithm. Nonetheless, many authors have identified a range of conditions that give rise to superunitary speedup [32, 33, 3, 4] even if we compare against the same code on a single processor. Some of the sources of superunitary speedup are the following:

- *Increase of high-speed memories.* When the number of processors is increased, the number of high-speed memories (e.g., cache, CPU registers) is also increased. While a large number of individuals might not fit into small but high-speed memories, if the population is distributed in many processors, it is possible that the resulting subpopulations can perfectly fit in high-speed local memories instead of the low-speed RAM.
- *Reduced overhead.* The reason is that the time consumed on some operating system's kernel calls on an n processor machine is only $1/n$ of the time required on a single processor.
- *Shift algorithm's profile.* When a fixed-time model is used, superunitary speedup may result if a parallel algorithm spends more time in faster routines.

Inasmuch as these are mainly hardware sources, it is important to present details of the parallel architecture used, specially if superunitary speedup is reported.

In the specialized literature, we can find other conditions under which superunitary speedup may appear. For instance, time gains obtained from traversing the search space in parallel or improvements due to the use of smaller abstract data structures (e.g., a list for the primary population or the external archive). However, these conditions may produce superunitary speedup only when the parallel MOEA is compared to a sequential MOEA with a global population or when a subefficient sequential algorithm is employed.

5.2 Other Parallel Performance Measures

- **Efficiency** (E). This metric [39] measures the fraction of time that a processor is effectively utilized. It is defined by the ratio between the speedup and the number of processors used. This metric is defined as:

$$E_p = \frac{S_p}{p}, \tag{11}$$

  where S_p is the speedup achieved with p processors. In an ideal system, the efficiency is equal to 1. In reality, due to the communication costs, and the idle time caused by the synchronization, the efficiency oscillates between 0 and 1.
- **Serial Fraction** (F). This metric was defined by Karp and Flatt [37] to estimate the serial fraction[6] for measuring the performance of a parallel algorithm on a fixed-size problem. Mathematically, it is defined as:

[6] The ratio of the time taken by the inherently serial component of an algorithm to its execution time on one processor.

$$F_p = \frac{1/S_p - 1/p}{1 - 1/p}. \tag{12}$$

where S_p the speedup achieved with p processors. Smaller values of F are considered better. If F increases with the number of processors, then it is a symptom of growing communications costs, and, therefore, an indicator of poor scalability. Thus, if the speedup of an algorithm is small, we can still say that it is efficient if F_p remains constant when increasing p. In this case, the efficiency drops due to the limited parallelism of the algorithm. On the other hand, if the value of F decreases with p, Karp and Flatt [37] consider that the speedup tends to be superlinear.

6 Selection of Parallel MOEAs

Although several researchers have reported the use of parallel MOEAs [48, 16], not many of them have actually proposed novel schemes to parallelize a MOEA. Next, we will review the most representative work along these lines that we were able to find in the specialized literature.

The Divided Range Multi-Objective Genetic Algorithm (DRMOGA) was proposed by Hiroyasu et al. [35]. Here, the global population is sorted according to one of the objective functions (which is changed after a number generations). Then, the population is divided into equally-sized sub-populations (see Figure 8). Each of these sub-populations is allocated to a different processor in which a serial MOEA is applied. At the end of a specific number of generations, the sub-populations are gathered and the process is repeated, but this time using some other objective function as the sorting criterion. The main goal of this approach is to focus the search effort of the population on different regions of the objective space. However, in this approach we cannot guarantee that the sub-populations will remain in their assigned region. A similar approach is followed by de Toro Negro et al. [19].

Zhu & Leung proposed the Asynchronous Self-Adjustable Island Genetic Algorithm (aSAIGA) [65]. In aSAIGA, rather than migrating a set of individuals, the islands exchange information related to their current explored region. The "exploration region" (ER) is the hypercube containing most of the individuals of the archive maintained by the sequential MOEA (see Figure 9). Based on the information coming from other islands, a "self-adjusting" operation modifies the fitness of the individuals in the island to prevent two islands from exploring the same region. In a similar way to DRMOGA, this approach cannot guarantee that the sub-populations move tightly together throughout the search space, hence the information about the explored region may be meaningless.

Another island pMOEA was introduced by Deb et al. [23]. In this case, although all processors search on the entire decision variable space, the

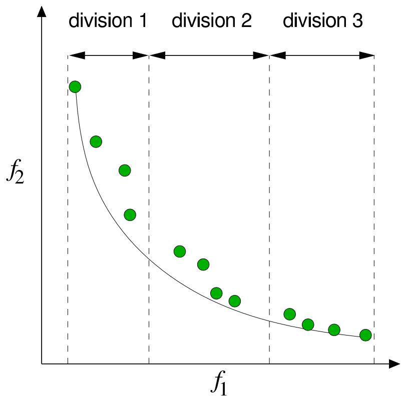
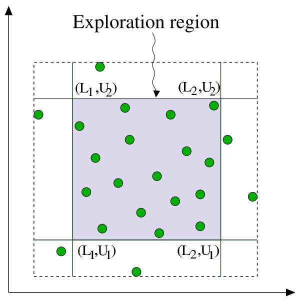


Fig. 8 Division of the population in DRMOGA.



The exploration region is defined by the hypercube $[L_1, U_1] \times \ldots \times [L_N, U_N]$. Let $\{f_i^1, f_i^2, \ldots, f_i^{n'}\}$ be the permutation with the ascending sorted values of the i-th ($i = 1, \ldots, N$) objective, where n' is the size of the archive and N the number of objectives. Then: $L_i = f_i^{\lceil \frac{1}{4}n \rceil}$, $U_i = f_i^{\lceil \frac{3}{4}n \rceil}$ for $1 \leq i \leq N$.

Fig. 9 Exploration region used in ASAIGA.

approach assigns each processor a different search region of the Pareto-optimal front. In order to steer the search towards the assigned region, the authors adopt a "guided domination" (see Figure 10) based on a concept defined by Branke, Kaußler and Schmeck [13]. This concept uses a weighted function of the objectives in order to achieve a larger dominated region for each vector. Therefore, each processor using this new concept only finds a region of the real Pareto front. The weakness of this approach is that we must have a priori knowledge of the shape of the Pareto front in order to define accurately the search directions. Furthermore, this technique can only deal with convex Pareto fronts.

Streichert et al. [56] proposed an approach that partitions the overall population using a clustering algorithm aiming to specialize the exploration of each island on different areas of the Pareto front. Periodically (after a specified number of generations) the islands are gathered, clustered and redistributed onto the available processors. The individuals are kept within their region by considering this as their feasible zone and using the constrained dominance principle defined by Deb et al. [20]. That is, any individual generated outside

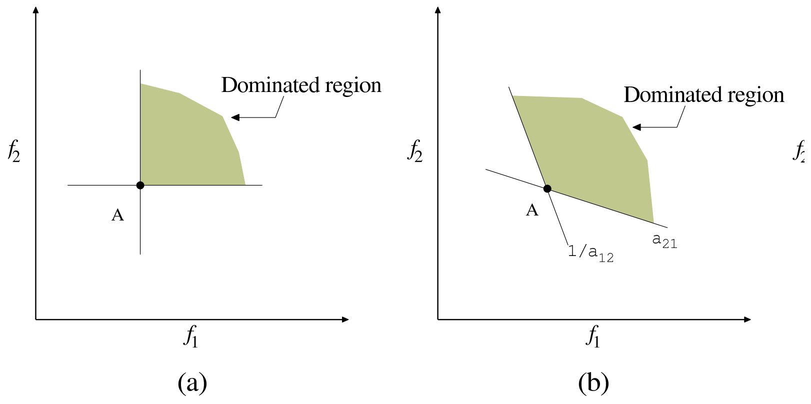


Fig. 10 Dominated region with the usual Pareto domination (a); dominated region with the concept of guided domination (b); "nondominated" region of the Pareto front (c).

its constrained region is marked as "invalid". The main drawback of the approach is that the repeated gathering of all sub-populations produces a high communication overhead, which is increased with the number of processors.

López Jaimes and Coello Coello [44] proposed an approach called Multiple Resolution Multi-Objective Genetic Algorithm (MRMOGA), which consists of a pMOEA based on the island paradigm, with heterogeneous nodes. The main idea of this approach is to encode the solutions using a different resolution in each island. Then, variable decision space is divided into hierarchical levels with well-defined overlaps. Evidently, migration is only allowed in one direction (from low resolution to high resolution islands). MRMOGA uses an external population, and the migration strategy considers such population as well (see Figure 11). The approach also uses a strategy to detect nominal convergence of the islands in order to increase their initial resolution. The rationale behind this approach is that the true Pareto front can be reached faster using this change of resolution in the islands, because the search space of the low resolution islands is proportionally smaller and, therefore, convergence is faster. The results indicated that MRMOGA outperforms a parallel version of NSGA-II, with a more significant difference as the number of processor increases.

7 Summary and Final Remarks

This chapter has presented an overview of parallel multi-objective evolutionary algorithms (pMOEAs). Firstly, we described the current parallel architectures available. Secondly, we reviewed the most common models to implement pMOEAs. Then, we presented some performance indicators that have

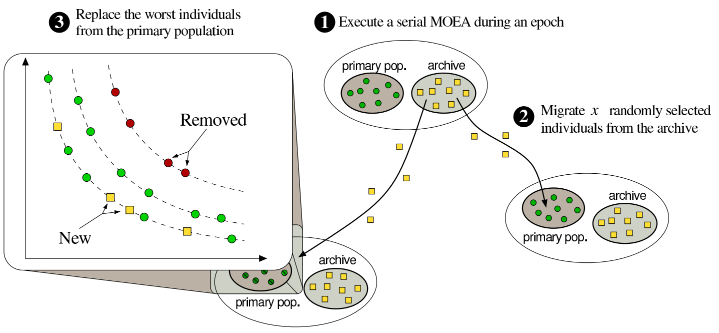

3 Replace the worst individuals from the primary population
1 Execute a serial MOEA during an epoch
2 Migrate x randomly selected individuals from the archive
Removed
New
primary pop.
archive

Fig. 11 Schematic view of MRMOGA.

been used to measure the efficiency of pMOEAs. Finally, a selection of some pMOEAs that incorporate innovative strategies was presented.

Global metacomputing (see Section 3) is one of the most promising current parallel architectures which is underused in evolutionary optimization. Grid computing is starting to be explored in multi-objective evolutionary optimization [46] and, to the best of our knowledge, no pMOEA has yet been implemented using volunteer computing in spite of the fact that both technologies represent an inexpensive alternative for supercomputing using existing local networks. Consequently, we expect that in the near future more researchers and practitioners will exploit these new forms of parallel computing.

In this chapter, we outlined some general strategies to distribute the objective set in the master-slave model, and some migration/replacement strategies. However, there is still room for new parallelization strategies.

Additionally, given the assessment methodology observed in the current literature, it is clear that there is no standard way to compare and easily ponder the performances reported of two different pMOEAs. In sequential MOEAs, we compare new algorithms against well-established MOEAs such as NSGA-II or SPEA2. Thus, we can appraise easily the value of new MOEAs reported in different works. In contrast, for pMOEAs there is no standard reference pMOEA to beat. Thereby, it is required a different methodology to assess performance and report the results produced by pMOEAs.

Acknowledgements The first author acknowledges support from CONACyT to pursue graduate studies in computer science at CINVESTAV-IPN.

References

[1] distributed.net project home Page (1997), `http://www.distributed.net`
[2] SETI@home project home Page (1999), `http://setiathome.berkeley.edu`

[3] Akl, S.G., Lindon, L.F.: Paradigms admitting superunitary behaviour in parallel computation. In: Buchberger, B., Volkert, J. (eds.) CONPAR 1994 and VAPP 1994. LNCS, vol. 854, pp. 301–312. Springer, Heidelberg (1994)

[4] Alba Torres, E.: Parallel evolutionary algorithms can achieve super-linear performance. Information Processing Letters 82(1), 7–13 (2002)

[5] Alba Torres, E., Troya Linero, J.M.: A survey of parallel distributed genetic algorithms. Complexity 4(4), 31–51 (1999)

[6] Alba Torres, E., Troya Linero, J.M.: Analyzing synchronous and asynchronous parallel distributed genetic algorithms. Future Generation Computer Systems 17(4), 451–465 (2001)

[7] Amdahl, G.M.: Validity of the single processor approach to achieving large scale computing capabilities. In: Proceedings of the AFIPS 1967, spring joint computer conference, April 18-20, 1967, pp. 483–485. ACM, New York (1967), http://doi.acm.org/10.1145/1465482.1465560

[8] August, M.C., Brost, G.M., Hsiung, C.C., Schiffleger, A.J.: Cray X-MP: The birth of a supercomputer. Computer 22(1), 45–52 (1989), http://dx.doi.org/10.1109/2.19822

[9] Belding, T.C.: The distributed genetic algorithm revisited. In: Eshelman, L. (ed.) Proceedings of the Sixth International Conference on Genetic Algorithms, pp. 114–121. Morgan Kaufmann, San Francisco (1995)

[10] Bell, G.: Ultracomputers: A teraflop before its time. Communications of the ACM 35(8), 27–47 (1992)

[11] Bell, G.: Bell's law for the birth and death of computer classes. Communications of the ACM 51(1), 86–94 (2008), http://doi.acm.org/10.1145/1327452.1327453

[12] Blank, T.: The MasPar MP-1 architecture. In: Compcon Spring 1990. Intellectual Leverage. Digest of Papers. Thirty-Fifth IEEE Computer Society International Conference, pp. 20–24 (1990)

[13] Branke, J., Kaußler, T., Schmeck, H.: Guidance in Evolutionary Multi-Objective Optimization. Advances in Engineering Software 32, 499–507 (2001)

[14] Brockhoff, D., Friedrich, T., Hebbinghaus, N., Klein, C., Neumann, F., Zitzler, E.: Do Additional Objectives Make a Problem Harder? In: Thierens, D. (ed.) 2007 Genetic and Evolutionary Computation Conference (GECCO 2007), vol. 1, pp. 765–772. ACM Press, London (2007)

[15] Cantú Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Boston (2002)

[16] Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007)

[17] Collette, Y., Siarry, P.: Multiobjective Optimization. Principles and Case Studies. Springer, Heidelberg (2003)

[18] Crowl, L.A.: How to measure, present, and compare parallel performance. IEEE Parallel Distrib. Technol. 2(1), 9–25 (1994), http://dx.doi.org/10.1109/88.281869

[19] de Toro Negro, F., Ortega, J., Ros, E., Mota, S., Paechter, B., Martin, J.M.: PSFGA: Parallel Processing and Evolutionary Computation for Multiobjective Optimisation. Parallel Computing 30(5-6), 721–739 (2004)

[20] Deb, K.: Multi-objective Evolutionary Optimization: Past, Present and Future. In: Parmee, I.C. (ed.) Proceedings of the Fourth International Conference on Adaptive Computing in Design and Manufacture (ACDM 2000), PEDC, University of Plymouth, UK, pp. 225–236. Springer, London (2000)

[21] Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Chichester (2001)

[22] Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable Multi-Objective Optimization Test Problems. In: Congress on Evolutionary Computation (CEC 2002), vol. 1, pp. 825–830. IEEE Service Center, Piscataway (2002)

[23] Deb, K., Mohan, M., Mishra, S.: Towards a Quick Computation of Well-Spread Pareto-Optimal Solutions. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 222–236. Springer, Heidelberg (2003)

[24] Dongarra, J., Sterling, T., Simon, H., Strohmaier, E.: High-performance computing: Clusters, constellations, MPPs, and future directions. Computing in Science and Engineering 7(2), 51–59 (2005), http://doi.ieeecomputersociety.org/10.1109/MCSE.2005.34

[25] Duncan, R.: A survey of parallel computer architectures. Computer 23(2), 5–16 (1990), http://dx.doi.org/10.1109/2.44904

[26] Edgeworth, F.Y.: Mathematical Physics. P. Keagan, London (1881)

[27] Eklund, S.E.: A massively parallel architecture for distributed genetic algorithms. Parallel Computing 30(5-6), 647–676 (2004), http://dx.doi.org/10.1016/j.parco.2003.12.009

[28] Flynn, M.J.: Some computer organizations and their effectiveness. IEEE Transactions on Computers 21(9), 948–960 (1972)

[29] Foster, I., Kesselman, C. (eds.): The grid: blueprint for a new computing infrastructure. Morgan Kaufmann Publishers Inc., San Francisco (1999)

[30] Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. Int. J. High Perform. Comput. Appl. 15(3), 200–222 (2001), http://dx.doi.org/10.1177/109434200101500302

[31] Giloi, W.K.: Towards a taxonomy of computer architecture based on the machine data type view. SIGARCH Comput. Archit. News 11(3), 6–15 (1983)

[32] Gustafson, J.L.: Fixed time, tiered memory, and superlinear speedup. In: Proceedings of the Fifth Distributed Memory Computing Conference, DMCC5 (1990)

[33] Helmbold, D.P., McDowell, C.E.: Modeling speedup (n) greater than n. IEEE Trans. Parallel Distrib. Syst. 1(2), 250–256 (1990), http://dx.doi.org/10.1109/71.80148

[34] Hillis, W.D.: The Connection Machine. MIT Press, Cambridge (1989)

[35] Hiroyasu, T., Miki, M., Watanabe, S.: The New Model of Parallel Genetic Algorithm in Multi-Objective Optimization Problems—Divided Range Multi-Objective Genetic Algorithm. In: 2000 Congress on Evolutionary Computation, vol. 1, pp. 333–340. IEEE Service Center, Piscataway (2000)

[36] Johnson, E.E.: Completing an MIMD multiprocessor taxonomy. SIGARCH Computure Architecture News 16(3), 44–47 (1988), http://doi.acm.org/10.1145/48675.48682

[37] Karp, A.H., Flatt, H.P.: Measuring parallel processor performance. Communications of the ACM 33(5), 539–543 (1990)

[38] Knowles, J., Corne, D.: Properties of an Adaptive Archiving Algorithm for Storing Nondominated Vectors. IEEE Transactions on Evolutionary Computation 7(2), 100–116 (2003)

[39] Kumar, V., Ananth Grama, G.K., Gupta, A.: Introduction to Parallel Computing: design and analysis of parallel algorithms. Benjamin Cummings Publishing Company, Redwood City (1994)

[40] León, C., Miranda, G., Segura, C.: Parallel hyperheuristic: a self-adaptive island-based model for multi-objective optimization. In: GECCO 2008: Proceedings of the 10th annual conference on Genetic and evolutionary computation, pp. 757–758. ACM, New York (2008), `http://doi.acm.org/10.1145/1389095.1389241`

[41] Licklider, J.C.R., Taylor, R.W.: The computer as a communication device. Science and Technology 76, 21–31 (1968)

[42] Lin, S.C., Punch III, W.F., Goodman, E.D.: Coarse-grain genetic algorithms, categorization and new approaches. In: Sixth IEEE Symposium on Parallel and Distributed Processing, pp. 28–37. IEEE Computer Society Press, Dallas (1994)

[43] Lizárraga Lizárraga, G., Hernández Aguirre, A., Botello Rionda, S.: G-metric: an m-ary quality indicator for the evaluation of non-dominated sets. In: GECCO 2008: Proceedings of the 10th annual conference on Genetic and evolutionary computation, pp. 665–672. ACM, New York (2008), `http://doi.acm.org/10.1145/1389095.1389227`

[44] López Jaimes, A., Coello Coello, C.A.: MRMOGA: A New Parallel Multi-Objective Evolutionary Algorithm Based on the Use of Multiple Resolutions. Concurrency and Computation: Practice and Experience 19(4), 397–441 (2007)

[45] López Jaimes, A., Coello Coello, C.A., Chakraborty, D.: Objective Reduction Using a Feature Selection Technique. In: 2008 Genetic and Evolutionary Computation Conference (GECCO 2008), pp. 674–680. ACM Press, Atlanta (2008)

[46] Luna, F., Nebro, A., Dorronsoro, B., Alba, E., Bouvry, P., Hogie, L.: Optimal Broadcasting in Metropolitan MANETs Using Multiobjective Scatter Search. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 255–266. Springer, Heidelberg (2006)

[47] Meuer, H.W.: The TOP500 project: Looking back over 15 years of supercomputing experience. Informatik-Spektrum 31(3), 203–222 (2008), `http://dx.doi.org/10.1007/s00287-008-0240-6`

[48] Nebro, A., Luna, F., Talbi, E.G., Alba, E.: Parallel Multiobjective Optimization. In: Alba, E. (ed.) Parallel Metaheuristics, pp. 371–394. Wiley-Interscience, New Jersey (2005)

[49] Obayashi, S., Sasaki, D.: Multiobjective Aerodynamic Design and Visualization of Supersonic Wings by Using Adaptive Range Multiobjective Genetic Algorithms. In: Coello Coello, C.A., Lamont, G.B. (eds.) Applications of Multi-Objective Evolutionary Algorithms, pp. 295–315. World Scientific, Singapore (2004)

[50] Okuda, T., Hiroyasu, T., Miki, M., Watanabe, S.: DCMOGA: Distributed cooperation model of multi-objective genetic algorithm. In: Proceedings of the PPSN/SAB Workshop on Multiobjective Problem Solving from Nature II (MPSN-II) (2002)

[51] Osyczka, A.: Evolutionary Algorithms for Single and Multicriteria Design Optimization. Physica Verlag, Germany (2002)

[52] Pareto, V.: Cours D'Economie Politique, vol. I and II. F. Rouge, Lausanne (1896)

[53] Sarmenta, L.F.G.: Volunteer computing. PhD thesis, Massachusetts Institute of Technology (2001)

[54] Sawai, H., Adachi, S.: Parallel distributed processing of a parameter-free GA by using hierarchical migration methods. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), vol. 1, pp. 579–586. Morgan Kaufmann, San Francisco (1999)

[55] Stadler, W.: Fundamentals of multicriteria optimization. In: Stadler, W. (ed.) Multicriteria Optimization in Engineering and the Sciences, pp. 1–25. Plenum Press, New York (1988)

[56] Streichert, F., Ulmer, H., Zell, A.: Parallelization of Multi-objective Evolutionary Algorithms Using Clustering Algorithms. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 92–107. Springer, Heidelberg (2005)

[57] Talbi, E.G., Mostaghim, S., Okabe, T., Ishibuchi, H., Rudolph, G., Coello Coello, C.A.: Parallel Approaches for Multi-objective Optimization. In: Branke, J., Deb, K., Miettinen, K., Slowinski, R. (eds.) Multiobjective Optimization. Interactive and Evolutionary Approaches. LNCS, vol. 5252, pp. 349–372. Springer, Heidelberg (2008)

[58] Tan, K., Khor, E., Lee, T.: Multiobjective Evolutionary Algorithms and Applications. Springer, London (2005)

[59] Tanenbaum, A.S., van Steen, M.: Distributed Systems: Principles and Paradigms. Prentice Hall, Upper Saddle River (2002)

[60] Teich, J., Zitzler, E., Bhattacharyya, S.S.: 3D Exploration of Software schedules for DSP Algorithms. In: 7th International Workshop on Hardware/Software Codesign (CODES 1999), pp. 168–172 (1999)

[61] Tomassini, M.: Parallel and distributed evolutionary algorithms: A review. In: Miettinen, K., Mäkelä, M., Neittaanmäki, P., Periaux, J. (eds.) Evolutionary Algorithms in Engineering and Computer Science, pp. 113–133. John Wiley and Sons, Chichester (1999)

[62] Van Veldhuizen, D.A., Zydallis, J.B., Lamont, G.B.: Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation 7(2), 144–173 (2003)

[63] Vrugt, J.A., Robinson, B.A.: Improved evolutionary optimization from genetically adaptive multimethod search. Proceedings of the National Academy of Sciences of the United States of America 104(3), 708–711 (2007)

[64] Vyssotsky, V.A., Corbató, F.J., Graham, R.M.: Structure of the Multics Supervisor. In: Proceedings of the AFIPS, Fall Joint Computer Conference (FJCC), Spartan Books, Las Vegas, Nevada, vol. 27, Part 1, pp. 203–212 (1965)

[65] Zhu, Z.Y., Leung, K.S.: An Enhanced Annealing Genetic Algorithm for Multi-Objective Optimization Problems. In: Langdon, W., Cantú-Paz, E., Mathias, K., Roy, R., Davis, D., Poli, R., Balakrishnan, K., Honavar, V., Rudolph, G., Wegener, J., Bull, L., Potter, M., Schultz, A., Miller, J., Burke, E., Jonoska, N. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), pp. 658–665. Morgan Kaufmann Publishers, San Francisco (2002)

[66] Zitzler, E., Teich, J., Bhattacharyya, S.S.: Evolutionary Algorithm Based Exploration of Software Schedules for Digital Signal Processors. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), vol. 2, pp. 1762–1769. Morgan Kaufmann, San Francisco (1999)

Asynchronous Multi-Objective Optimisation in Unreliable Distributed Environments

Andrew Lewis[1], Sanaz Mostaghim[2], and Ian Scriven[3]

Abstract This chapter examines the performance characteristics of both asynchronous and synchronous parallel particle swarm optimisation algorithms in heterogeneous, fault-prone environments. The chapter starts with a simple parallelisation paradigm, the Master-Slave model using Multi-Objective Particle Swarm Optimisation (MOPSO) in a heterogeneous environment. Extending the investigation to general, distributed environments, algorithm convergence is measured as a function of both iterations completed and time elapsed. Asynchronous particle updates are shown to perform comparably to synchronous updates in fault-free environments. When faults are introduced, the synchronous update method is shown to suffer significant performance drops, suggesting that at least partly asynchronous algorithms should be used in real-world environments. Finally, the issue of how to utilise newly available nodes, as well as the loss of existing nodes, is considered and two methods of generating new particles during algorithm execution are investigated.

Institute for Integrated and Intelligent Systems
Griffith University
Brisbane
Queensland
Australia
`A.Lewis@griffith.edu.au`
Institute AIFB
University of Karlsruhe
Germany
`sanaz.mostaghim@kit.edu`
School of Engineering
Griffith University
Brisbane
Queensland
Australia
`Ian.Scriven@student.griffith.edu.au`

1 Introduction

Optimisation of solutions using computational models is becoming an increasingly common practice in engineering design and scientific investigations. It has been applied to a wide variety of problems, from biomechanics [26] to avionics [44] and the demand is increasingly for the simultaneous optimisation of several, possibly competing objectives. Such real-world problems usually require complex, time-consuming computer simulations to solve potential solutions and so parallelisation of optimisation algorithms is a practical necessity. Originally confined to dedicated, parallel computing clusters or special-purpose parallel computers, the emergence of grid computing as a common approach to the provision of computational capacity [1] has dictated the development of optimisation algorithms capable of efficient and effective performance in potentially unreliable distributed environments. Such algorithms can also be applied effectively to utilise small-scale, *ad hoc* grids of networked computers, bringing the benefits of computational optimisation within the reach of small to medium enterprises.

Several Parallel Evolutionary Algorithms have been studied in the literature (e.g., [3, 6, 43, 16, 11, 10]). From these studies three main paradigms emerge: the Island model, Master-Slave model and Diffusion model. The focus of this chapter is on the Master-Slave model, a straight-forward method and perhaps the simplest, where a single processor maintains the optimisation task and uses the other processors for objective function evaluations.

We start by studying parallelisation of multi-objective optimisation (MO) algorithms using the Master-Slave model. The Master-Slave model is the simplest parallelisation paradigm when working in a homogeneous environment, and has been widely implemented in such. However, to our knowledge little has been done on the combination of multi-objective optimisation, heterogeneous resources and the Master-Slave model. Comparing multi-objective algorithm with their single objective counterparts there is a major difference. A multi-objective EA (MOEA) must evaluate not just enough trial solutions for effective search of the parameter space, but sufficient to allow determination of the relative fitness against the multiple objectives. In a Master-Slave model, the master processor has to wait for all of the evaluations from the other processors. Then it can apply a ranking method to the solutions to find the non-dominated front and continue the optimisation. In a heterogeneous environment, the waiting time might be very long. In some cases, the fast processors can deliver twice the evaluations of a slow one, in a given period of time. In this work, we study a new algorithm which utilises all of the computing resources, from the slow to the very fast (as in a normal grid).

Solving multi-objective problems on heterogeneous systems has a number of challenges:

- Dealing with heterogeneous resources, the aim is to use all of the computing resources but at the same time to be efficient in time, i.e., the master node must make use of all of the available function evaluations in every time step. Here, the master node encounters a trade-off between waiting for the slowest processor or continuing the optimisation based on the available solutions.

- The output of multi-objective optimisation problems is usually a set of so-called Pareto-optimal solutions. Most multi-objective evolutionary algorithms (MOEAs) approximate these solutions by finding a set of non-dominated solutions. In every generation, a typical MOEA method applies a ranking-based method to the *entire* population to evaluate the solutions.
- Efficient and effective exploration of the search space is another important issue, in common with most practical applications of optimisation. The Master-Slave model is typically used to solve computationally expensive problems. The main task would be to employ a reasonable exploration technique while minimising the computational cost.

One drawback of the Master-Slave model is the communication overhead between processors. Since the simulation models considered in real-world engineering design problems are typically complex and computationally expensive, the computation time will usually outweigh communication time, so this factor can safely be ignored.

Particle Swarm Optimisation (PSO) is fast being established as an efficient, parallel optimisation technique for both single and multiple objective design problems. In Multi-Objective Particle Swarm Optimisation (MOPSO), each particle (individual) follows its own best position and the position of *a global guide*. Hence, in the parallelisation scheme, it is enough to find a guide for a particular particle to improve that particle's position. This is an advantage compared to other MOEA techniques such as SPEA2 [48] or NSGAII [17]. In SPEA2, all of the individuals are compared to each other and are given a strength and in NSGAII a ranking method finds different non-dominated fronts. In these methods, the best evaluation is based on the *entire* population.

In distributed systems, different kinds of failures can happen. Some typical failures among the others are: nodes may get overloaded, fail or lose their connection to the other nodes. In this chapter, we also address optimisation using such unreliable environments. We name the new approach Parallel Asynchronous Particle Swarm Optimisation (PAPSO) and we examine the effects of asynchronous updates in PAPSO algorithms in fault-prone environments, and compare the convergence characteristics of Parallel Synchronous Particle Swarm Optimisation (PSPSO) [37] and PAPSO algorithms against both iteration count and wall-clock execution time.

Apart from investigating the failures in the system, we address the aspect of having new nodes in the available system, known as *churn*. We use the new nodes for improving the quality of the approximated front obtained by MOPSO.

This chapter is organised as follows. In the following we introduce the test functions used throughout the chapter. In the next sections (2 and 3), we study synchronous and asynchronous parallel multi-objective particle swarm optimisation. Sections 4 and 5 are about unreliable environments and parallel MOPSO methods working on such environments. In Section 6, we address the churn issue in our approach and propose two new methodologies for covering and extending the edges of the obtained approximated front. Section 7 concludes the chapter.

Table 1 Test functions

Test	Function	Constraints
ZDT1	$g(x_2,\cdots,x_n) = 1+9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_i \in [0,\ 1]$
	$h(f_1,g) = 1-\sqrt{f_1/g}$	$n = 30$
	$f_1(x_1) = x_1$	$i = 1,2,\ldots,n$
	$f_2(\mathbf{x}) = g(x_2,\cdots,x_n)\cdot h(f_1,g)$	
ZDT2	$g(x_2,\cdots,x_n) = 1+9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_i \in [0,\ 1]$
	$h(f_1,g) = 1-(f_1/g)^2$	$n = 30$
	$f_1(x_1) = x_1$	$i = 1,2,\ldots,n$
	$f_2(\mathbf{x}) = g(x_2,\cdots,x_n)\cdot h(f_1,g)$	
ZDT3	$g(x_2,\cdots,x_n) = 1+9(\sum_{i=2}^{n} x_i)/(n-1)$	$x_i \in [0,\ 1]$
	$h(f_1,g) = 1-\sqrt{f_1/g}-(f_1/g)\sin(10\pi f_1)$	$n = 30$
	$f_1(x_1) = x_1$	$i = 1,2,\ldots,n$
	$f_2(\mathbf{x}) = g(x_2,\cdots,x_n)\cdot h(f_1,g)+1$	
FF	$f_1(\mathbf{x}) = 1-exp(-\sum_i (x_i-\frac{1}{\sqrt{n}})^2)$	$n = 10$
	$f_2(\mathbf{x}) = 1-exp(-\sum_i (x_i+\frac{1}{\sqrt{n}})^2)$	$x_i \in [-4,\ 4]$
DTLZ	$f_1(\mathbf{x}) = (1+g(x_M))\cos(x_1\pi/2)\cos(x_2\pi/2)$	$x_i \in [0,\ 1]$
	$f_2(\mathbf{x}) = (1+g(x_M))\cos(x_1\pi/2)\sin(x_2\pi/2)$	$n = 8$
	$f_3(\mathbf{x}) = (1+g(x_M))\sin(x_1\pi/2)$	
	$g(x_M) = \sum_{i=3}^{8}(x_i-0.5)^2$	

1.1 Test Functions

For the experiments in this chapter, some standard test functions from the multi-objective optimisation literature are used. These test functions are two- and three-objective optimisation problems selected from [48, 18], as shown in Table 1.

2 Master-Slave Model of Parallelisation

The simplest parallelisation paradigm for optimisation is the Master-Slave model. This model is aimed at distributing the (objective function) evaluation of the individuals on several slave computing resources while a master resource executes the optimisation procedure. The master node can also be used for evaluations.

This model is of great benefit when the function evaluations are very expensive. It is very natural to use parallel computers to solve expensive functions but care must be taken that the algorithm searches efficiently. In addition, a key issue is that in many situations the computing resources are heterogeneous. In the Master-Slave model, the central computing resource (master node) has to gather information from all of the resources and perform the optimisation steps (such as ranking etc.) based on the solutions obtained. However, waiting for the slowest processor may take a long time.

The main questions when solving problems on heterogeneous computing resources therefore are:

Algorithm 1 MOPSO

```
Initialise population
repeat
    Find non-dominated solutions; store in the archive
    Update the population and the archive
    Apply turbulence factor
until Termination condition met
Return archive
```

1. how to efficiently search the space and
2. how to use all of the resources so that none of them remains idle.

3 Parallel Multi-Objective Particle Swarm

A typical MOPSO method is shown in Algorithm 1. It starts with a random population of solutions (also called particles). Every particle i has a position in the search space and a velocity, denoted by the vectors $\mathbf{x}_i$ and $\mathbf{v}_i$, respectively. The non-dominated particles are stored in an archive and the population is updated based on the positions of the global best particles P_{global} and their own personal memories P_{Best}:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t) \tag{1}$$

$$\mathbf{v}_i(t) = w\mathbf{v}_i(t-1) + R_1(P_{Best} - \mathbf{x}_i(t-1)) + R_2(P_{global} - \mathbf{x}_i(t-1)) \tag{2}$$

where w, R_1 and R_2 denote the inertia weight and control parameters. In MOPSO, selecting the best personal position and the global positions has a great impact on the quality of the solutions [5, 29]. The global best is usually selected from the archive. To avoid local optima in MOPSO, a percentage of the particles are randomly selected and moved in order to explore the search space. This is implemented using a turbulence factor. MOPSO is iteratively continued until a stopping criterion, such as a certain number of iterations or evaluations, is met. A very good survey about MOPSO can be found in [35].

Parallelising MOPSO based on the Master-Slave model for a set of homogeneous resources can be easily achieved. Algorithm 2 shows a (synchronous) Parallel MOPSO on an homogeneous set of resources. Since the computing resources are homogeneous it is safe to assume the results from slave nodes will be returned to the master node in a timely and synchronised manner, *provided the computing resources and interconnection network are dedicated to the optimisation task and free of failures*. In every iteration, the master node distributes the evaluation tasks and waits for all of the resources to finish.

However, the main issue in dealing with heterogeneous (asynchronous) resources is that in the worst case of having only one slow computing resource, almost all of the computing resources remain idle while the master node waits for the slowest one. Here, we want to make use of all of the computing resources in a way that, as soon as

Algorithm 2 Synchronous MOPSO

```
Initialise population
repeat
    Distribute evaluations of the population members on the computing resources
    Wait for all of the computing resources to finish evaluations
    Find non-dominated solutions; store in the archive
    Update the population and the archive
    Apply turbulence factor
until Termination condition met
Return archive
```

a reasonable amount of resources have completed evaluations, the master resource continues optimising without further waiting. This is not to imply the results of the slow computers are not also of interest as they also contain information about the search space. The information from them will be incorporated when it arrives at a later stage.
The first step in parallelisation on a heterogeneous set of resources is to define a good set of initial particles and send them to the slave nodes for evaluation. Usually in MOPSO a random set is used; here we propose to use a set of random particles defined by a **recursive** Gap Search (GS) Method as explained below. While use of Gap Search is peripheral to the investigation of asynchronous update in the MOPSO algorithm, it is a novel approach to satisfying the goal of efficient search of the parameter space.

3.1 Exploration Using Recursive Gap Search Method

For searching the least explored regions of a search space, Gap Search (GS) algorithm has been proposed which is very similar to the Binary Search [23]. GS is a recursive search algorithm which finds the large gaps in the search space and samples solutions in those areas where the binary search employs a successive search in most promising areas in the search space. In GS, the search is started by selecting a random point in the search space. Then the largest empty region is found in the space by computing the distances between the first point and the boundaries of the search space. Figure 1 (a) shows an example for a two dimensional space. The next point is randomly selected in that empty region (Figure 1 (b)). This is an iterative method and can be performed for any desired number of solutions. These solutions are usually stored in a list.

One disadvantage of GS is that selecting a point from a large list of solutions requires a relatively high computation time, particularly for spaces of high (> 20) dimensionality. In the cases at which this study is aimed, it is assumed the time taken for objective function evaluations will dominate.

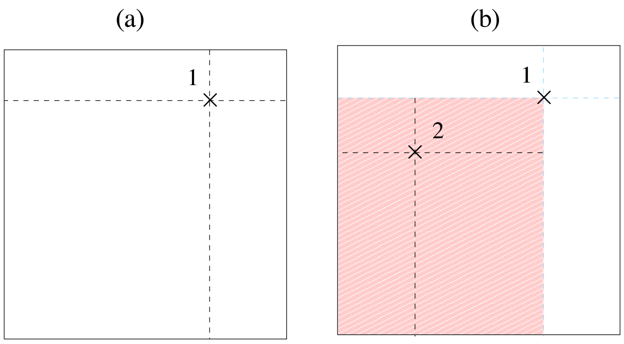


Fig. 1 An example of the Gap Search Method in a two dimensional space. In (a) one point is randomly selected in the search space. The next point is selected in the most empty region defined in (b). ©2008 IEEE

3.2 Parallel MOPSO on Heterogeneous Resources

We propose a MOPSO which starts with an initial set of solutions based on the recursive Gap Search method. These selected solutions are stored in a list and are sent to the slave processors for evaluation. The master node waits for N_s $(1 \leq N_s \leq N)$ processors where N is the total number of available processors. After receiving those evaluations, the non-dominated solutions are stored in the archive and the master node sends new evaluation jobs to the N_s processors. Note that this differs from a synchronous algorithm in that the master node is determining Pareto-dominance of solutions based on incomplete information. A synchronous MOPSO waits for *all* solution evaluations to be returned before processing them.

Before sending the parameter sets, the master node qualifies the evaluation of the processors. If the solution found by a processor is:

- dominated by one of the archive members: the master processor selects a proper global guide from the archive and updates the position of the corresponding particle based on MOPSO. Then the master processor sends the new position to the slave processor for evaluation.
- not dominated by any other archive member: the master processor gives the task of exploration to the corresponding processor. Based on the Gap Search Method, a proper random position in an unexplored region in the space is selected and is sent to the slave processor for evaluation.

The exploration task done by GS replaces the turbulence factor of the original MOPSO. The population members are stored in the Gap List over the generations. Algorithm 3 shows the parallel MOPSO on heterogeneous resources.

Algorithm 3 Asynchronous MOPSO

```
Initialise population using GS
Store the GS points in a List
Distribute evaluations of the population members on the computing resources
repeat
  Wait for at least $N_s$ of the computing resources to finish evaluations
  Find non-dominated solutions; store in the archive
  For all of the dominated solutions
    Update using MOPSO
  For all of the non-dominated solutions
    Explore using GS
    Update the GS List
  Update the archive
  Redistribute the evaluations on the resources
until Termination condition met
Return archive
```

3.3 Discussion

In this algorithm, the parameter N_s, the least number of processors to wait for, must be known in advance. In fact, this parameter depends on the heterogeneous resources and the estimated load on them. It can also change during the iterations.

In the proposed Parallel MOPSO, we do not use a turbulence factor. Instead, we apply an intentional exploration to the non-dominated solutions using the Gap Search method. The reason is that the non-dominated solutions cannot be significantly improved by the conventional MOPSO as they constitute the global best particles from which guides are drawn. Therefore, their positions are stored in the archive and we assign them a new position based on the Gap Search. On the other hand, the position of the dominated particles can be improved by the guidance of the non-dominated ones and therefore the MOPSO method can be used effectively.

Also, in this algorithm we not only store the non-dominated solutions in the archive but we also store all the population members (explored by GS) in a list. In fact, the list is very useful to obtain knowledge about the unexplored regions, in contrast to the idea of selecting any random position in parameter space using the turbulence factor. It must be mentioned that the size of the list has a great impact on the computation time. Here, we assume that the function evaluations require a relatively high computation time and that there is a limited number of solutions in the lists so that the computation time of finding the unexplored regions in the list can be neglected.

3.4 Experiments

The major goal of the experiments was to evaluate the quality of the solutions when running the system for a certain time and adding waiting time (execution

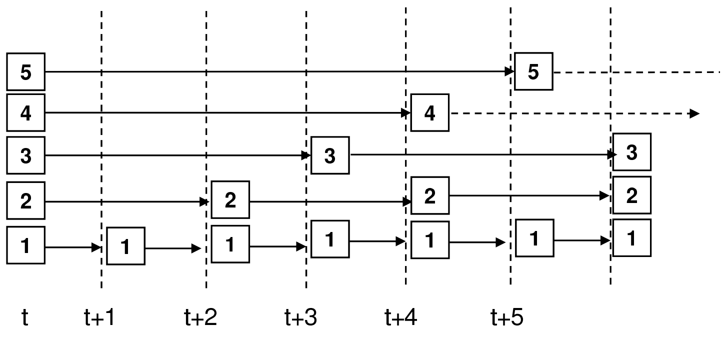


Fig. 2 Five different resources are illustrated in terms of speed. The resource type 1 is five times faster than the resource type 5. Resource types are shown in the Boxes.

heterogeneity) to the resources. The system was analysed by adding waiting time to the fast resources to wait for a certain percentage of the slow ones. The experiments were performed on a simulation environment containing 100 resources with 5 different computation speeds. The simulation was based on a real scenario in a typical grid. Figure 2 illustrates 5 different kinds of available resources. There are 3, 3, 20, 43, and 31 number of type 1 (very fast), type 2, type 3, type 4, and type 5 (very slow) resources, respectively.

The test function FF was selected from Table 1 for these experiments. The quality of the solutions was computed by the hypervolume metric [48] averaged over 20 runs.

The asynchronous MOPSO described, an extension of the algorithm from Mostaghim and Teich [29], was executed on the master node using 100 particles. The total time T_{total} was set to be 120, meaning the fast resources were able to evaluate 120 particles.

The main archive stored in the master node was set to be empty initially (for the first run). Parameter settings were selected so that the first runs of the 100 resources included only a random sampling of the search space until the fast processors finished their evaluations and updated the archive.

Figure 3 shows the quality of the archive members over time. Different plots illustrate the quality if the resources that complete wait for j other resources to finish and update the archive ($j = 0, 10, 20, 30, 40, 50$).

We observe that, if the resources do not wait, the results are worse than if they wait for at least 10 to 20 percent of the resources. This result is to be expected as, in multi-objective optimisation, the results depend on each other through the dominance relation. If the resources wait (in this case) for up to 20 percent, they achieve a better quality than if they do not wait. In fact, in every step of optimisation it is better to have enough evaluations to find more dominated and non-dominated solutions. Hence waiting for a time has the advantage of receiving more solutions. This leads to a better direction in the optimisation than not waiting. However, waiting for

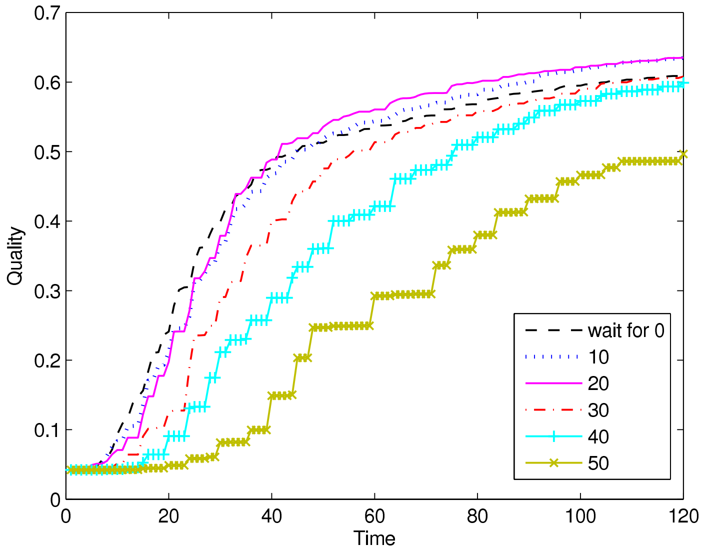


Fig. 3 The quality of the solutions obtained over time. The plots show the quality if the computing resources wait for at least 0, 10, 20, 30, 40 and 50 other computing resources to continue the optimisation. A long waiting time corresponds to the worst quality, whereas waiting time of less than 20 percent is shown to increase the quality.

a large number of other resources means that the fast processors stay idle for a long time which is, on the other hand, undesirable when allowed only a fixed time for the entire optimisation process (such as when using the grid).

4 Unreliable Distributed Environments

The experiments described in the previous section demonstrated the improvements achievable using asynchronous updates in an environment consisting of heterogeneous computing resources. Parallel MOPSO was based on the straightforward Master-Slave parallelisation paradigm. In this section we seek to extend the algorithm to use in a decentralised distributed computing environment. At the time of writing, the only studies on asynchronous particle updates in parallel particle swarm algorithms have focused mainly on parallel efficiency [26, 45], without thoroughly examining the impact of parallel updates on algorithm convergence time, and without looking at the impact of evaluation failure. Failure is an important factor to consider, as it is practically unavoidable in most environments, whether caused by inter-node communication breakdowns, node failure, or even failure of the solver itself. Indeed, in distributed computing environments it is prudent to assume it is not a matter of *if* failures of computers or communications will occur, but *when*. The work described in this section examines the effects of asynchronous updates in distributed MOPSO algorithms in fault-prone environments (Parallel Asynchronous MOPSO), and compares the convergence characteristics of Synchronous and Asynchronous MOPSO against both iteration count and wall-clock execution time.

5 Distributed Particle Swarm Optimisation

As described earlier, the simplest and most common distributed particle swarm optimisation algorithms utilise a master-slave architecture, where a single controlling (master) processor runs only the optimisation algorithm, and utilises external (slave) processors to compute potential solutions [32]. Care must be taken, however, in translating this model to an **unreliable** distributed computing environment, as failures may impact the master node itself.

In a synchronous MOPSO, the algorithm will wait for all particles to be solved before updating particle velocities and positions using equations 1 and 2. This allows the term **iteration** to be easily defined as one step of the algorithm in which all particles are evaluated. In asynchronous MOPSO, however, the algorithm will wait for only a certain proportion of the particle solutions to be returned before updates are carried out, making an iteration harder to define. For the purposes of comparing convergence against iterations completed, a single iteration of a asynchronous algorithm will be defined as a period in which the number of evaluations performed equals the number of particles in the population.

The performance of synchronous algorithms will be seriously impacted if the computing resources are not homogeneous. If some particles take longer to compute than others, at the end of each iteration most nodes in the cluster will be idle, reducing parallel efficiency, and increasing the execution time of the algorithm. Worse still, if one or more particle solutions are not returned in a timely fashion (in which case they can be said to have failed), those particles must be evaluated again, further decreasing parallel efficiency and negatively impacting the time taken to arrive at the optimal solution. The asynchronous algorithms will not suffer as significantly from these issues. A 1% synchronous environment (i.e., fully asynchronous, the algorithm does not wait for any other evaluations to be completed) asynchronous MOPSO will theoretically obtain near 100% parallel efficiency, and the impact of failures will be lessened, as a failed particle will not delay any others. In a 50% synchronous environment, the asynchronous MOPSO will still have a somewhat higher parallel efficiency than the synchronous one, and will only be impacted by failures if half or more of the particles fail to evaluate.

However, it can be expected that the parallel asynchronous MOPSO will have some disadvantages compared to the synchronous one in regards to convergence against iterations completed. Whereas the sequential asynchronous MOPSO yielded convergence improvements over the synchronous MOPSO, the parallel synchronous MOPSO will have somewhat decreased performance. The cause of this becomes obvious when examining the amount of information (previous solutions) available when any given particle is updated. In the sequential asynchronous MOPSO, the number of previous solutions PS_{sa} for particle j at iteration i for a swarm of size N is given by Equation 3.

$$PS_{sa} = (i-1)\times N + (j-1) \tag{3}$$

For example, on the second ($i = 2$) iteration of a $N = 100$ MOPSO, particle $j = 50$ will have 149 previous solutions available to it. This is more than would be available to a synchronous MOPSO, which would only have the 100 solutions from the previous iteration available for the same example, as given by Equation 4.

$$PS_{ps} = (i-1)\times N \tag{4}$$

Calculating the amount of information available to a purely asynchronous MOPSO is slightly more complex, being based somewhat on when the particle evaluation is completed, but for a particle having around average evaluation time, it can be approximated as in Equation 5.

$$PS_{pa} \simeq (i-1)\times N - \frac{N}{2} \tag{5}$$

Unlike the sequential asynchronous and synchronous MOPSO, the Parallel asynchronous MOPSO at iteration $i = 2$ will not usually have all the information from the previous iteration available to it, as many particles will still be under evaluation. As such, for the example used previously, a particle on the second iteration will only have, on average, 50 previous solutions available to them. This clearly suggests that, in terms of iterations completed, the parallel asynchronous MOPSO should converge more slowly than the synchronous one. It is expected, however, that this performance decrease will be countered somewhat by the increased parallel efficiency of the asynchronous MOPSO, which will allow particle evaluations (and thus iterations) to be completed more quickly than in the synchronous one.

5.1 Simulation Setup and Testing Procedure

In order to examine the performance of the parallel asynchronous and synchronous MOPSO in heterogeneous, fault-prone conditions, a distributed particle swarm simulation environment was constructed using both 100 particles and 100 computational nodes. A MOPSO algorithm was used, with P_{global} and P_{best} being chosen at random from the non-dominated sets of solutions found by the entire population and by the individual particles, respectively.

Particle evaluation times were controlled so as to conform to a normal distribution, with a mean execution time of $\mu = 5$ seconds and a standard deviation $\sigma = 1$ second. Modeling the evaluation times in this way meant that failures could be detected by monitoring the time taken by each solver. If a solution was not returned with $\mu + 4\sigma$ seconds, it was assumed that the solver had failed and required that the evaluation be repeated. This is a safe assumption, as (from the properties of the normal distribution) 99.997% of evaluations would have completed in this allowed time.

The solver used was the analytical ZDT1 test function described earlier. Simulations were run for varying degrees of synchronous behaviour, ranging from 1% (fully asynchronous) through to 100% (fully synchronous), and for failure rates of 0%, 5%, 10% and 20%. It should be noted that these failure rates are *mean* failure rates, as the probability of an evaluation failure was based on a normal distribution. Results were averaged over fifteen runs to ensure reliable, repeatable experimental data was obtained. A convergence factor metric was used which measured area of the potential solution space not dominated by an approximation to the Pareto-optimal front as a ratio of the solution space not dominated by the actual Pareto-front. This convergence metric is given mathematically as

$$ConvergenceFactor = \frac{PA_{approximate}}{PA_{actual}} \tag{6}$$

where $PA_{approximate}$ is the non-dominated area under the approximate Pareto-optimal front, and PA_{actual} is non-dominated area under the real Pareto-optimal front, which can be derived mathematically for the test functions used. An approximation to the Pareto-optimal front that is equal to the actual Pareto-front has a convergence factor of one, and a solution set that has not converged will have a convergence factor less than one (but greater than zero). This convergence factor was monitored as a function of both iterations completed and time elapsed.

In addition to node failure, grid (or node) diversity will also impact the performance of the MOPSO variants. In a more diverse grid, solution times will vary more widely, increasing the parallel efficiency advantage of the asynchronous algorithm. Conversely, a grid containing mostly homogeneous nodes will lead to very similar solution times, decreasing the parallel efficiency advantage of the asynchronous method, making a more synchronous approach desirable. To test this hypothesis, simulations were performed with varying execution time distributions for both the fully synchronous and fully asynchronous implementations.

5.2 Results

The first test carried out involved measuring the convergence of the algorithms as a function of iterations completed in a fault-free environment, in order to confirm the previously stated hypothesis that the more asynchronous algorithm variations will take more iterations to converge to the same point as the less asynchronous algorithms. The average convergence factor for 1% synchronous, 30% synchronous, 70% synchronous and 100% synchronous MOPSOs over 100 iterations is shown in Figure 4.

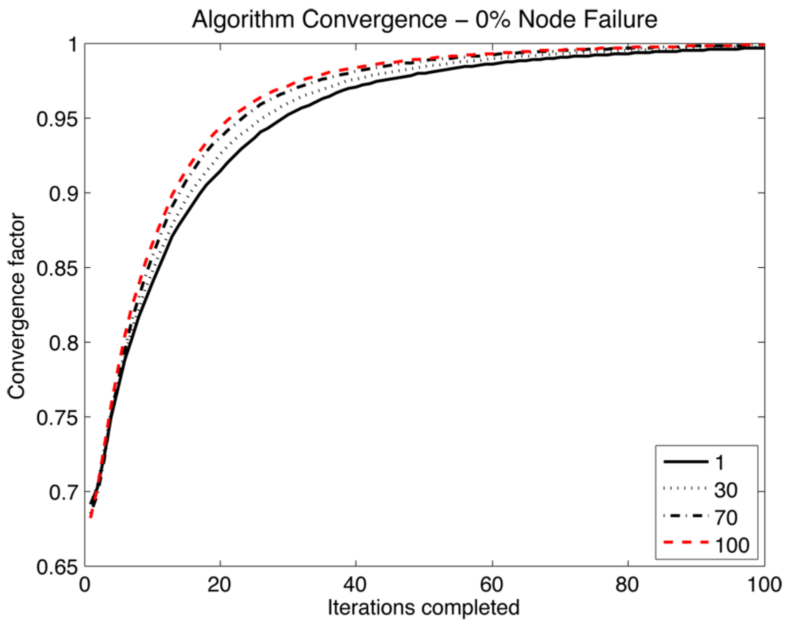
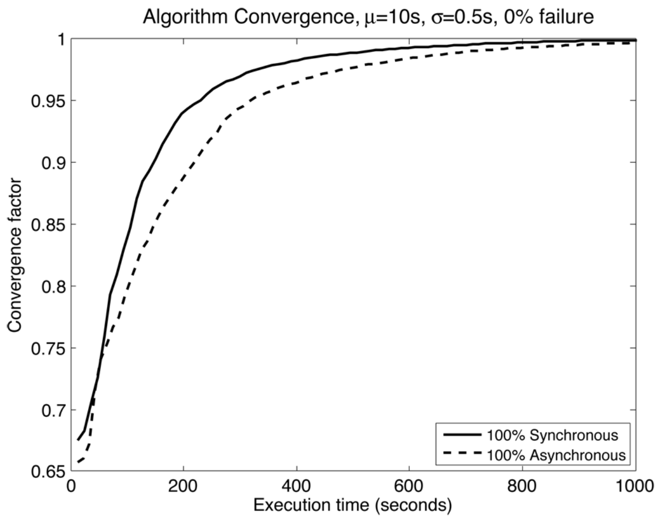


Fig. 4 Algorithm convergence for varying degrees of synchronous behaviour - 1% (fully asynchronous), 30%, 70% and 100% (fully synchronous) - as a function of iterations completed, with 0% failure rate. ©2008 IEEE

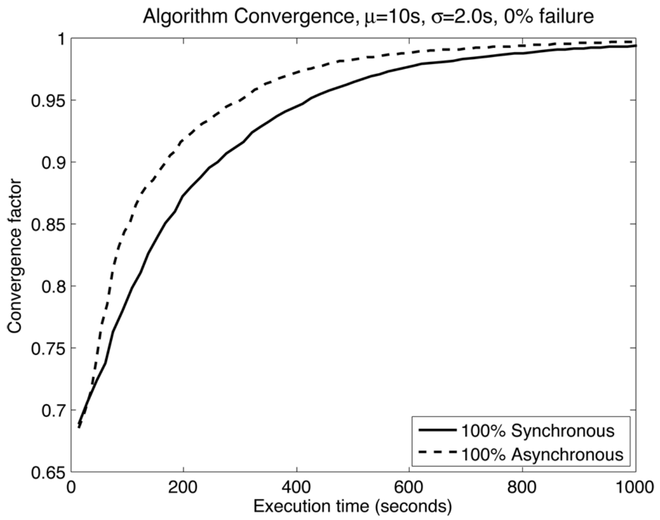
These results support the stated hypothesis, clearly showing that the fully synchronous algorithm (the parallel synchronous MOPSO) converges faster with respect to iterations completed than the asynchronous MOPSO variations, with the purely asynchronous algorithm performing the worst of all, as expected. The more synchronous the algorithm is, the faster it converges with respect to iterations completed.

Figure 5 (a) shows the algorithm convergence as a function of time for an execution time distribution having a 5% standard deviation, representing a mostly homogeneous grid environment. In such an environment it can be seen that the synchronous update mechanism outperforms the asynchronous one, due to the reduced parallel efficiency advantage gained through the use of asynchronous updates. At some solver execution time distribution, the performance of the different update mechanisms can be expected to be approximately equal. When grid diversity is increased passed this critical point, the parallel efficiency advantage provided by asynchronous particle updates outweighs the iteration-wise convergence increase of the synchronous algorithm. This is confirmed in Figure 5 (b), which shows the convergence rate of the two algorithm variations when the distribution of solver execution times has a standard deviation of 20%.

The introduction of faults has very little effect on the asynchronous (1%, 30% and 70%) algorithms, whereas the performance of the synchronous algorithm drops



(a) 5% standard deviation in solver execution time



(b) 20% standard deviation in solver execution time

Fig. 5 Algorithm convergence for both completely synchronous and asynchronous behaviour

markedly. Figure 6 shows the same algorithm performance measure for a probability of failure of 10%. Once the failure rate reaches 10%, the 90% synchronous MOPSO's performance also starts to degrade. With increasing failure rates, the synchronous MOPSO degrades further. This extra drop in performance can be attributed

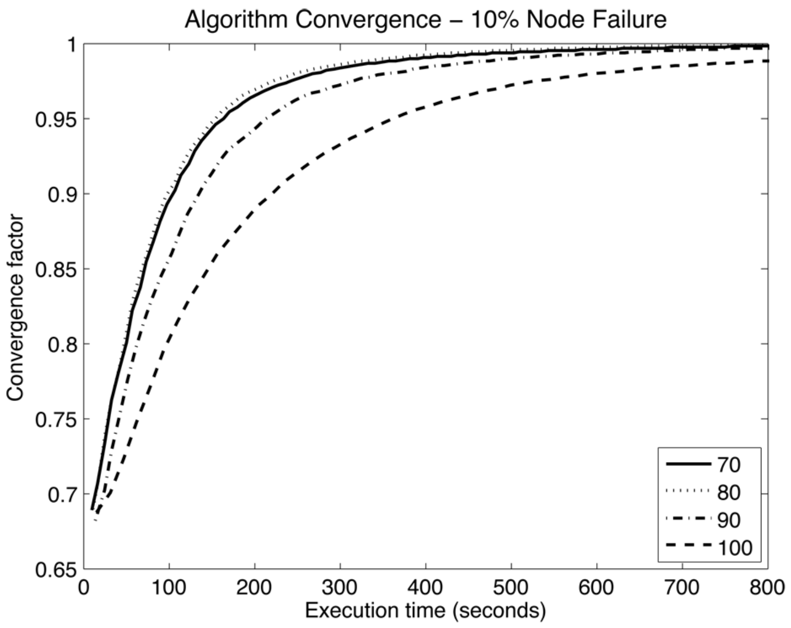


Fig. 6 Algorithm convergence for low degrees of asynchronous behaviour - 70%, 80%, 90% and 100% (fully synchronous) - as a function of algorithm execution time, with 10% failure rate. ©2008 IEEE

to particles failing multiple times in the same iteration, further increasing the time required for each iteration.

The performance losses shown in the synchronous MOPSO and the more asynchronous MOPSO algorithms would not be evident on graphs displaying algorithm convergence as a function of completed algorithm iterations, highlighting the significance of examining the execution time of such parallel algorithms in heterogeneous, fault-prone environments.

6 Addressing Churn

To this point, the analysis of the algorithms has focused on their performance in the presence of failures. In a distributed computing environment, it may be the case that additional nodes will become available for use. New nodes may appear either by addition to the resources, particularly in instances in which the computational resources comprise an *ad hoc* grid in a peer-to-peer environment [41, 39], or by a node which previously failed returning to operation. Together with failures, the turnover in resources is collectively known as *churn*. When additional resources appear, the optimisation algorithm must decide how best to use them, generating new particles for them to evaluate.

The generation of new particle positions is a new problem which required investigation. Particle generation during the initialisation stage of evolutionary optimisation algorithms has been investigated, with the general opinion tending towards random initialisation of particles when no information on the problem space is available [47]. Some studies, however, have shown that with some prior knowledge of the problem space, intelligent positioning of the initial population leads to faster convergence to the optimal solutions [34, 4, 15, 33]. When generating new particles during the optimisation process, significant information regarding both the problem space and the current swarm location will be available, and this chapter investigates a number of methods of utilising this information to improve convergence speeds in unreliable distributed environments.

6.1 Proposed Approaches

Mid-execution of a particle swarm optimisation process, the algorithm has access to the current approximate to the Pareto-optimal front (that is, the **best** solutions found by the algorithm to that point in time). Using this information, new particle positions can be generated with aim to improve the Pareto front, by either attempting to fill gaps in it, or by extending its edges.

6.1.1 Approach 1: Filling Gaps in the Pareto Front

The governing equations of the particle swarm algorithm, neglecting the inertia term, cause particles to search between two points in the search space - the local and global guide solutions. Using a similar approach when generating new particles is therefore an obvious choice. Instead of using local and global guides (the former being unavailable as we are generating a *new* particle), the two locations to search between can be taken as the two bounding solutions in the largest gap in the current Pareto-front, for example solutions $\mathbf{x}'$ and $\mathbf{x}''$ in Figure 7.

Once the bounding solutions are selected, each parameter x_k is calculated for the new particle using equation 7.

$$x_k = \frac{r_1 x'_k + r_2 x''_k}{r_1 + r_2} \tag{7}$$

The symbols r_1 and r_2 in (7) represent uniformly distributed random numbers between zero and one. Given the assumption of correlation between parameter and objective space (which is inherent in the particle swarm method), the newly generated particle should fall roughly between the two bounding solutions in the objective space.

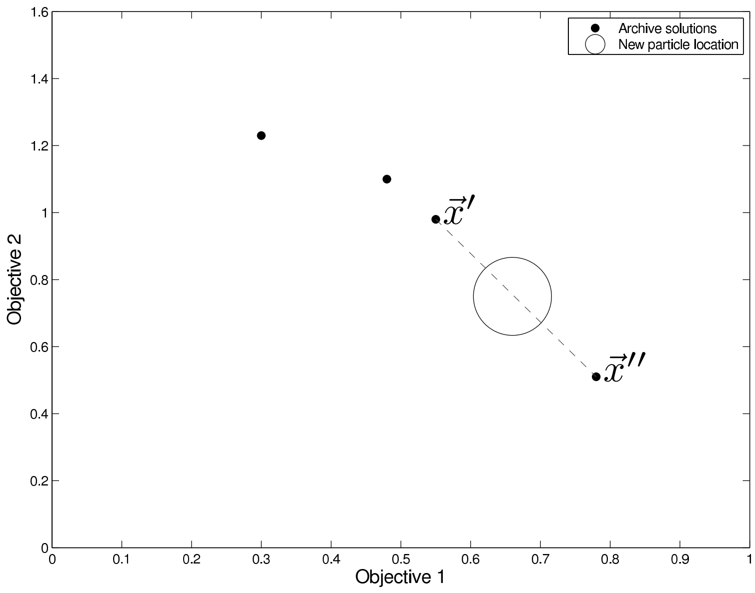


Fig. 7 Generating a new particle location by filling the largest gap in the pareto front

6.1.2 Approach 2: Extending the Edges of the Pareto Front

The gap-filling approach may cause the swarm to cluster towards the centre of the Pareto front, as it will never generate new particles outside the bounds of the current solutions on the edge of the front. In order to extend the edges of the front, a different approach must be used. The edge points are defined by sorting on an objective and choosing the extremes. Once the extremal solutions are selected, each parameter x_k is calculated for the new particle using equation 8.

$$x_k = \frac{r_1 x'_k + r_2(x'_k + (x'_k - x''_k))}{r_1 + r_2} \tag{8}$$

The symbols r_1 and r_2 in (8) again represent uniformly distributed random numbers between zero and one. The newly generated particle will this time be positioned past the current edges of the Pareto front in the solution space, again assuming correlation between the parameter and objective spaces. This approach is shown in Figure 8.

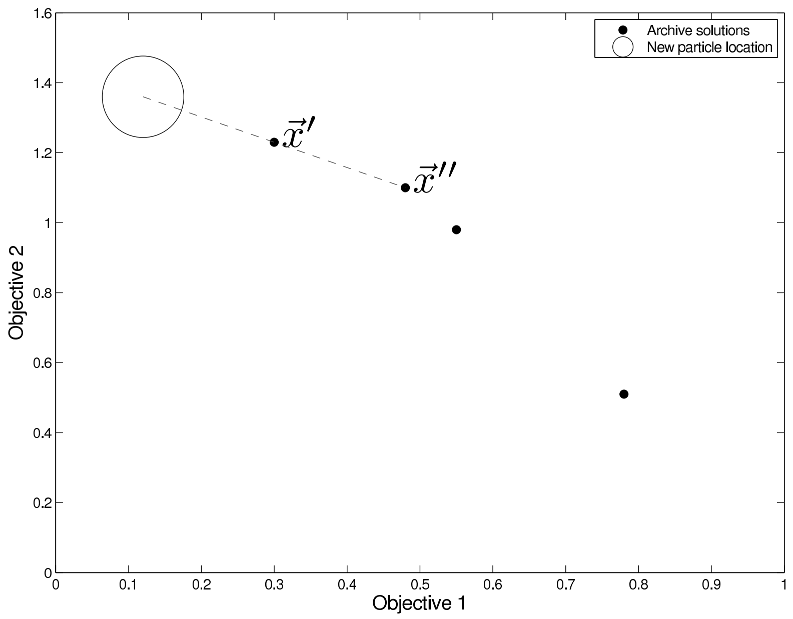


Fig. 8 Generating a new particle location by extending the edges of the Pareto front

6.1.3 Approach 3: A Hybrid Method

Both of the previously discussed approaches have both advantages and disadvantages. The gap filling method will promote good coverage of the global Pareto front, but only between the currently located extremal solutions. The edge extending approach, on the other hand, will promote exploration outside the current approximation to the Pareto front, but at the expense of convergence in the centre. It makes sense, then, to use a combination of the two approaches in order to gain the benefits of both while limiting the disadvantages.

6.2 Testing Procedure

In order to examine the performance of each approach, a testing environment was created utilising a thirty particle PSO algorithm, with one particle being removed and replaced using each of the various approaches at set intervals. Four popular analytical test functions were used as follows. Test functions ZDT1, ZDT2 and DTZL were each run for 200 iterations, with a particle replacement occurring every five iterations. The FF test function was run for 70 iterations (as it was a simpler, easier to solve problem), with particle replacement every second iteration.

Each optimisation process was executed 200 times, with convergence and coverage metric data being averaged to give reliable results. The convergence and coverage metrics used are described later in this section. As well as the three previously described approaches, a baseline test was included where no particles were added or removed, as well as a random test where new particles were randomly generated.

6.3 Algorithm Performance Metrics

Up until this point, a single algorithm performance metric has been sufficient for examining the effects of altering the implementation of a distributed particle swarm optimisation algorithm. However, in order to comprehensively examine the effect of an algorithmic modification such as creating new particles during execution, two separate performance metrics are required [17]. The two metrics used are described here.

6.3.1 Convergence Metric

The metric used to measure algorithm convergence (the closeness of a non-dominated set to the real Pareto-optimal front) in this study does so by measuring the volume/area of the objective space which is not dominated by a given approximation to the Pareto-optimal front. This is somewhat similar to the popular hypervolume metric, which evaluates the area/volume of the objective space which is dominated by a given set of non-dominated solutions [48]. This inverted hypervolume metric was used as the lower bounds for all objective in each of the test functions used are known to be zero, providing a convenient boundary for a hypervolume-like metric. In the case of minimisation problems, such as the test functions used for this chapter, a smaller value for this metric denotes a closer approximation to the global Pareto-optimal front.

6.3.2 Coverage Metric

The coverage metric used for this research is a new one, as no existing metrics were deemed suitable for the purposes of this study. Existing diversity metrics [17] measure the statistical spread or evenness of the Pareto-front, rather than quantifying how well a given non-dominated set covers the objective space. An approximate front may be very evenly spread (that is, the distances between adjacent particles are relatively similar). However, it may contain very few solutions and/or not adequately cover the entire range of possible objective values. In such cases existing diversity metrics would give a misleading evaluation of the quality of the approximate Pareto-optimal front.

A new coverage metric is proposed whereby the objective space is divided into sectors originating from the utopia point. The value of the diversity metric is then given as the ratio of these sectors which contain at least one member of the non-dominated set to the total number of sectors, or, mathematically:

$$\Psi = \frac{1}{N}\sum_{n=1}^{N} \psi_n \tag{9}$$

$$\text{where } \psi_n = \begin{cases} 1, \text{ if } \exists \mathscr{S} \in \mathscr{P}\mathscr{F}, \alpha_{n-1} \leq \tan\frac{f_1(\mathbf{x})}{f_2(\mathbf{x})} \leq \alpha_n \\ 0, \text{ otherwise} \end{cases}$$

A simple example of this metric can be seen in Figure 9. In this example, the solution space is divided into ten equal segments, with seven of these segments containing solutions in the current approximation to the Pareto-optimal front. This approximate Pareto front would therefore be classified as having a 0.7 or 70% coverage factor. Practically, more segments are used to give a higher resolution result - in this study, the same number of segments were used as there were particles in the swarm.

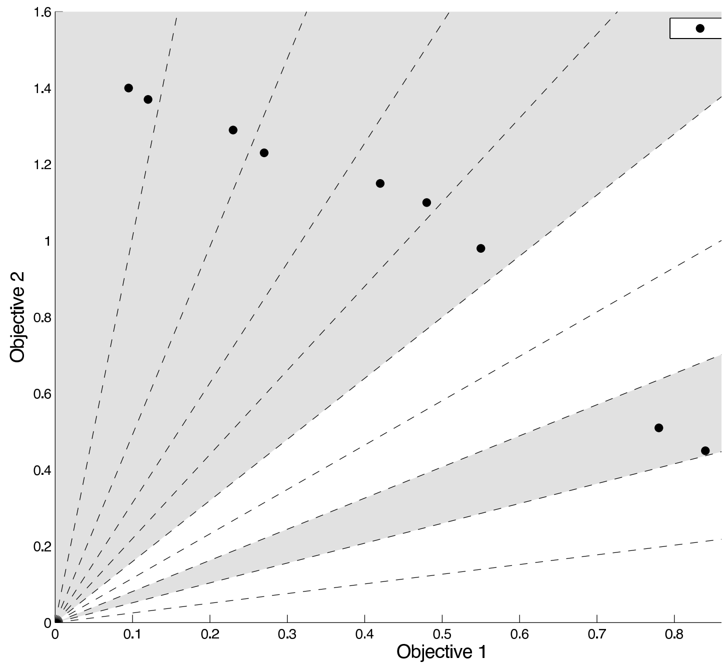


Fig. 9 Application of the proposed coverage metric to an example pareto front

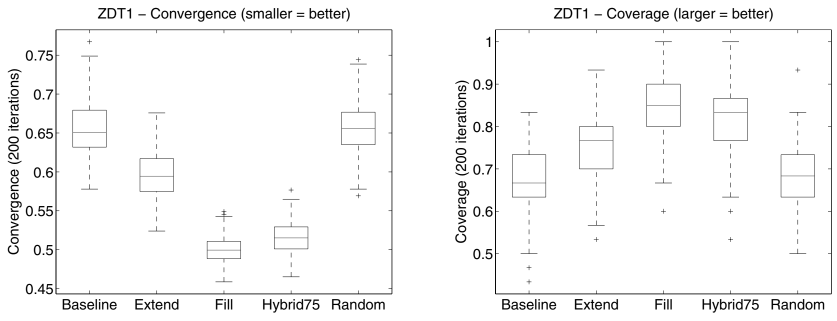


Fig. 10 Convergence (left) and coverage (right) of PSO algorithm on ZDT1 problem.

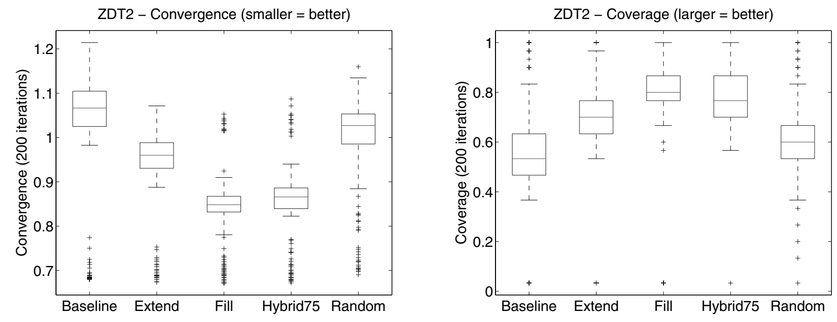


Fig. 11 Convergence (left) and coverage (right) of PSO algorithm on ZDT2 problem.

6.4 Experimental Results

Box plots of the convergence and coverage metrics for each test function and algorithmic approach can be seen in Figures 10 through 13.

It can be seen that the gap filling method significantly out-performs edge extending approach in both the ZDT1 and ZDT2 test functions, in terms of both convergence and coverage of the Pareto-optimal front. Both approaches appear to result in better optimisation results than the baseline algorithm and the random generation method.

The two approaches faired slightly differently when applied to the FF test function. The gap filling method still resulting in increased algorithm convergence when compared to extending the edges, however the difference between the two approaches here was significantly less than for the ZDT functions. Unlike the results for two ZDT functions, here the edge extending approach achieves, on average, better coverage of the Pareto front than the gap filling method for generating new particles.

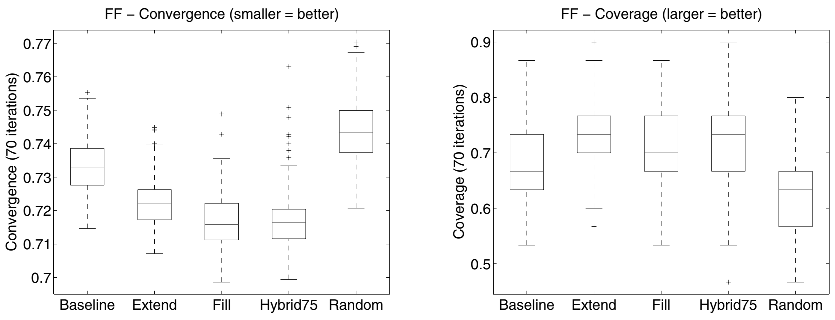


Fig. 12 Convergence (left) and coverage (right) of PSO algorithm on FF problem.

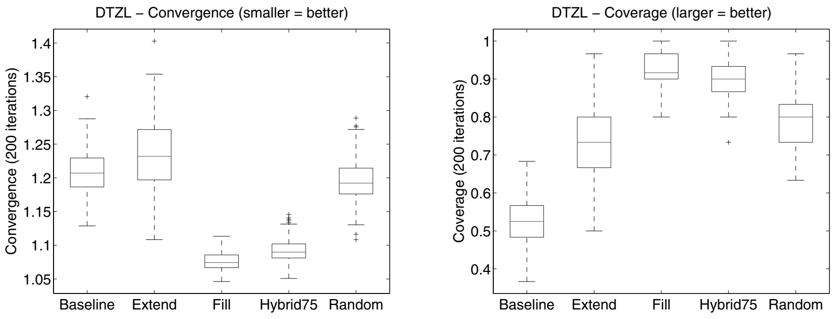


Fig. 13 Convergence (left) and coverage (right) of PSO algorithm on DTZL problem.

The results for the DTZL test function are comparable to those obtained using the ZDT1 and ZDT2 functions, with the gap filling approach being the top performer.

The hybrid approach, which generated new particles using the gap filling approach 75% of the time and used the edge extending approach the other 25%, can be seen to give the most consistent results across all four test functions. In all cases, the results produced by the hybrid approach compare quite favourable with those of the top performing method.

A question remains as to how well the methods described in this section scale with increased numbers of objectives, i.e., an objective space of higher dimensionality. When finding gaps, it is not difficult to generalise the method to higher dimensions. It suffices to find the nearest neighbour for each point in the set of non-dominated solutions by means of calculating the Euclidian distance between them in objective space, and select the pair for which this distance is a maximum. This operation has a computational complexity of $O((N/2)^2)$ but, given the assumption that the evaluation of objective functions dominates computation time, this can be ignored. Extending edges is less simple: as the dimensionality of objective space increases, so do the number of edges to be considered and it is not clear that the function of the algorithm will be best served by choosing only extremal points. The effects of higher dimensionality on algorithm performance remain a subject for future work.

With regard to the coverage metric, the metric operates essentially by taking an $N-1$ dimensional “slice” through the objective space, trivially extensible to multiple dimensions. As, however, this effectively collapses the distribution of the non-dominated set into a projection onto a single dimension, care should be exercised interpreting results for higher-dimensional problems.

7 Conclusions

In this chapter, we studied the Master-Slave model of parallelisation to solve multi-objective optimisation problems on a heterogeneous set of resources. We proposed a new hybrid Parallel MOPSO method called PMOPSO which uses a Gap Search method to search unexplored regions of the parameter space. PMOPSO was tested on a scenario containing a set of heterogeneous resources and compared with 3 other cases. The results show a good quality of solutions even when compared with a non-parallel case which takes a longer computation time. The proposed approach is particularly suitable for very expensive multi-objective problems of real-world applications using a heterogeneous set of processors.

Consideration of asynchronous update in parallel MOPSO algorithms was extended from the Master-Slave model to a general, distributed computing approach. From the results presented in this chapter it can be concluded that asynchronous particle updates, while negatively impacting algorithm convergence with respect to iterations completed, allow parallel distributed particle swarm algorithms to gracefully and efficiently handle node and solver heterogeneity as well as node, network and solver failure. Using asynchronous particle updates has been shown to increase parallel efficiency [26], which in turn decreases the computation time required for each algorithm iteration. This decrease in iteration time mitigates the decrease in algorithm convergence as a function of iterations completed which is introduced by the asynchronous update mechanism.

Simulations performed have shown that the more likely failures are, the more asynchronous the parallel particle swarm algorithm should be made. The degree of grid heterogeneity will need to be considered alongside grid volatility when setting the degree of synchronisation in particle updates. In fairly reliable environments such as fixed, dedicated high-performance computer clusters, mostly synchronous algorithms can be used, however if deploying into more volatile environments, such as distributed computer grids or non-dedicated peer-to-peer systems, the algorithms used should be made more asynchronous to counter the higher number of failures that can be expected. The degree of synchronisation could be set dynamically, allowing the performance of the algorithm to be maintained in dynamic grid environments.

Finally, we considered the issue of churn, in which compute resources may not only disappear through failure, but appear, by a variety of mechanisms, to take on tasks. Two methods, Pareto front gap filling and edge extending, and proposed a hybrid approach were evaluated for their ability to fully and effectively utilise available resources in decentralised, fault-prone distributed environments. The proposed approaches were tested using a number of popular analytical optimisation test functions, and performance evaluated with convergence and coverage metrics. Generating new particles by attempting to fill gaps in the Pareto-optimal front generally performed better than extending the front's edges. However it was shown that extending the edges of the Pareto-front can be more advantageous in certain scenarios. From this, a hybridised method was formulated to utilise both approaches while favouring gap filling, and this hybrid method was shown to perform consistently well across all problems used for testing.

Results presented here have shown that asynchronous updates in parallel particle swarm algorithms can lead to significant performance increases in diverse grid environments, where solver execution times will vary widely. Future work is desirable to investigate dynamic methods to respond to changing computing environments, in both the degree of synchronisation and manner in which additional compute resources are utilised. In addition, further analysis of the performance of PAPSO algorithms on other test problems with higher numbers of parameters and objectives is required.

Acknowledgements The authors would like to thank Jürgen Branke for his contributions to the early stages of this work. Some portions of the work described in this chapter are reprinted, with permission, from Scriven, I., Ireland, D., Lewis, A., Mostaghim, S. and Branke, J., "Asynchronous Multiple Objective Particle Swarm Optimisation in Unreliable Distributed Environments" and Mostaghim, S., Branke, J., Lewis, A. and Schmeck, H., "Parallel Multi-objective Optimization Using Master-slave Model on Heterogeneous Resources", in WCCI 2008 Proceedings (2008 IEEE World Congress on Computational Intelligence) ©2008 IEEE.

References

[1] Abramson, D., Lewis, A., Peachy, T.: Nimrod/O: A tool for automatic design optimization. In: The 4th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2000) (2000)

[2] Al-Kazemi, B., Mohan, C.: Multi-phase generalization of the particle swarm optimization algorithm. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol 2, pp. 1057–1062 (2002)

[3] Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. IEEE Transactions on Evolutionary Computation 6(5), 443–461 (2002)

[4] Angeline, P.J.: Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 601–610. Springer, Heidelberg (1998)

[5] Branke, J., Mostaghim, S.: About selecting the personal best in multi-objective particle swarm optimization. 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. 523–532. Springer, Heidelberg (2006)

[6] Branke, J., Kamper, A., Schmeck, H.: Distribution of evolutionary algorithms in heterogeneous networks. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 923–934. Springer, Heidelberg (2004)
[7] Branke, J., Schmeck, H., Deb, K., Reddy, M.: Parallelizing Multi-Objective Evolutionary Algorithms: Cone Separation. In: IEEE Congress on Evolutionary Computation, pp. 1952–1957 (2004)
[8] Branke, J., Deb, K., Miettinen, K., Slowinski, R.: Multiobjective Optimization Interactive and Evolutionary Approaches. Springer, Heidelberg (2008)
[9] Bui, L.T., Abbass, H.A., Essam, D.: Local models - an approach to distributed multiobjective optimization. Journal of Computational Optimization and Applications (2007)
[10] Cantu-Paz, E.: Designing efficient master-slave parallel genetic algorithms. IlliGAL Report 97004, University of Illinois (1997)
[11] Cantu-Paz, E.: A survey of parallel genetic algorithms. IlliGAL Report 97003, University of Illinois (1997)
[12] Cantu-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer, Dordrecht (2000)
[13] Carlisle, A., Dozier, G.: An off-the-shelf PSO. In: Proceedings of the 2001 Workshop in Particle Swarm Optimisation, pp. 1–6 (2001)
[14] Censor, Y., Zenios, S.A.: Parallel Optimization: Theory, Algorithms, and Applications. Oxford University Press, Oxford (1997)
[15] Chou, C.H., Chen, J.-N.: Genetic algorithms: initialization schemes and genes extraction. In: The Ninth IEEE International Conference on Fuzzy Systems, 2000. FUZZ IEEE 2000, vol. 2, pp. 965–968 (2000)
[16] Coello, C.A.C., Veldhuizen, D.A.V., Lamont, G.B.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer Academic Publishers, Dordrecht (2002)
[17] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)
[18] Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: Congress on Evolutionary Computation, pp. 825–830. IEEE, Los Alamitos (2002)
[19] Deb, K., Zope, P., Jain, A.: Distributed computing of pareto-optimal solutions with evolutionary algorithms. In: International Conference on Evolutionary Multi-Criterion Optimization, pp. 534–549 (2003)
[20] Fonseca, C.M., Fleming, P.J.: On the performance assessment and comparison of stochastic multiobjective optimizers. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 584–593. Springer, Heidelberg (1996)
[21] Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Inc, Reading (1989)
[22] Gupta, I., Ganesh, A.J., Kermarrec, A.M.: Efficient and adaptive epidemic-style protocols for reliable and scalable multicast. IEEE Transactions on Parallel and Distributed Systems 17(7), 593–605 (2006)
[23] Hughes, E.J.: Multi-objective binary search optimisation. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 102–117. Springer, Heidelberg (2003)
[24] Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann, San Francisco (2001)

[25] Knowles, J.: A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers. In: IEEE Intelligent Systems Design and Applications (ISDA V) (2005)

[26] Koh, B., George, A.D., Haftka, R.T., Fregly, B.J.: Parallel asynchronous particle swarm optimisation. International Journal for Numerical Methods in Engineering 67(4), 578–595 (2006)

[27] Laredo, J.L.J., Castillo, P.A., Mora, A.M., Merelo, J.J.: Evolvable agents, a fine grained approach for distributed evolutionary computing: walking towards the peer-to-peer computing frontiers. Soft Computing 12(12), 1145–1156 (2008)

[28] Laredo, J.L.J., Eiben, A.E., van Steen, M., Merelo, J.J.: On the run-time dynamics of a peer-to-peer evolutionary algorithm. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 236–245. Springer, Heidelberg (2008)

[29] Mostaghim, S., Teich, J.: Strategies for finding good local guides in multi-objective particle swarm optimization. In: IEEE Swarm Intelligence Symposium, pp. 26–33 (2003)

[30] Mostaghim, S., Teich, J.: A new approach on many objective diversity measurement. In: Dagstuhl Proceedings number 04461, Dagstuhl, Germany (2004)

[31] Mostaghim, S., Teich, J.: A new approach on many objective diversity measure. In: Proceedings of the Dagstuhl Seminar 04461 (2005)

[32] Mostaghim, S., Branke, J., Schmeck, H.: Multi-objective particle swarm optimization on computer grids. In: The Genetic and Evolutionary Computation Conference, vol. 1, pp. 869–875 (2007)

[33] Mostaghim, S., Branke, J., Lewis, A., Schmeck, H.: Parallel multi-objective optimization using a master-slave model on heterogeneous resources. In: IEEE, Congress on Evolutionary Computation (CEC) (2008)

[34] Rahnamayan, S., Tizhoosh, H.R., Salama, M.M.A.: A novel population initialization method for accelerating evolutionary algorithms. Comput. Math. Appl. 53(10), 1605–1614 (2007)

[35] Reyes-Sierra, M., Coello, C.A.C.: Multi-objective particle swarm optimizers: A survey of the state-of-the-art. International Journal of Computational Intelligence Research 2(3), 287–308 (2006)

[36] Riget, J., Vesterstrøm, J.: A diversity-guided particle swarm optimizer - the ARPSO. Technical report, University of Aarhus, Department of Computer Science (2002)

[37] Schutte, J.F., Reinbolt, J.A., Fregly, B.J., Haftka, R.T., George, A.D.: Parallel global optimisation with particle swarm algorithm. International Journal for Numerical Methods in Engineering 61, 2296–2315 (2004)

[38] Scriven, I., Lewis, A., Ireland, D., Lu, J.: Distributed multiple objective particle swarm optimisation using peer to peer networks. In: IEEE Congress on Evolutionary Computation (CEC) (2008)

[39] Scriven, I., Lewis, A., Smith, M., Friese, T.: Resource evaluation and node monitoring in service oriented ad-hoc grids. In: Proc. Sixth Australasian Symposium on Grid Computing and e-Research (AusGrid 2008), CRPIT, vol. 82, pp. 65–71 (2008)

[40] Scriven, I., Lu, J., Lewis, A.: An efficient peer-to-peer particle swarm optimiser for EMC enclosure design. In: The 13th Biennial IEEE Conference on Electromagnetic Field Computation, CEFC (2008)

[41] Smith, M., Friese, T., Freisleben, B.: Towards a service-oriented ad hoc grid. In: Proc. 3rd International Symposium on Parallel and Distributed Computing. IEEE Computer Society, Los Alamitos (2004)

[42] Talbi, E.G., Mostaghim, S., Okabe, T., Ichibushi, H., Rudolph, G., Coello, C.C.: Parallel Approaches for Multiobjective Optimization, pp. 349–372. Springer, Heidelberg (2008)
[43] Veldhuizen, D.A.V., Zydallis, J., Lamont, G.B.: Considerations in engineering parallel multiobjective evolutionary algorithms. IEEE Transactions on Evolutionary Computation 7(2), 144–173 (2003)
[44] Venter, G., Sobieszczanski, J.: Multidisciplinary optimisation of a transport aircraft wing using particle swarm optimisation. Structural and Multidisciplinary optimisation 26(1-2), 121–131 (2004)
[45] Venter, G., Sobieszczanski-Sobieski, J.: A parallel particle swarm optimisation algorithm accelerated by asynchronous evaluations. Journal of Aerospace Computing, Information, and Communication 3(3), 123–137 (2006)
[46] Wickramasinghe, W.R.M.U.K., van Steen, M., Eiben, A.E.: Peer-to-peer evolutionary algorithms with adaptive autonomous selection. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), pp. 1460–1467 (2007)
[47] Xie, X.F., Zhang, W.J., Yang, Z.L.: Adaptive particle swarm optimization on individual level. In: 2002 6th International Conference on Signal Processing, vol. 2, pp. 1215–1218 (2002)
[48] Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Shaker (1999)

Dynamic Problems and Nature Inspired Meta-heuristics

Tim Hendtlass[1], Irene Moser[1], and Marcus Randall[2]

Abstract Biological systems have often been used as the inspiration for search techniques to solve continuous and discrete combinatorial optimisation problems. One of the key aspects of biological systems is their ability to adapt to changing environmental conditions. Yet, biologically inspired optimisation techniques are mostly used to solve static problems (problems that do not change while they are being solved) rather than their dynamic counterparts. This is mainly due to the fact that the incorporation of temporal search control is a challenging task. Recently, however, a greater body of work has been completed on enhanced versions of these biologically inspired meta-heuristics, particularly genetic algorithms, ant colony optimisation, particle swarm optimisation and extremal optimisation, so as to allow them to solve dynamic optimisation problems. This survey chapter examines representative works and methodologies of these techniques on this important class of problems.

1 Introduction

Many industrial optimisation problems are solved in environments that undergo continual change. These problems are referred to as *dynamic optimisation problems* and are characterised by an initial problem definition and a series of 'events' that change it over time. An event defines some change

Centre for Information Technology Research
School of Information and Communication Technologies
Swinburne University, VIC 3001
{thendtlass, imoser}@ict.swin.edu.au
School of Information Technology
Bond University, QLD 4229
Australia
mrandall@bond.edu.au

either to the data of the problem or its structural definition *while the problem is being solved*. In comparison to static optimisation problems, dynamic optimisation problems often lack well defined objective functions, test data sets, criteria for comparing solutions and standard formulations [7, 37].

Evolutionary algorithms are those based on natural and biological systems. A very common example of these are genetic algorithms (GAs). For this class of algorithms, extensive modifications to accommodate dynamic optimisation problems have been made. A survey of these approaches up to the year 2000 is given by Branke [18]. However, for another group of evolutionary algorithms, *Ant Colony Optimisation* (ACO) [35], *Particle Swarm Optimisation* (PSO) [39] and *Extremal Optimisation* (EO) [13], suitable modifications and applications to these difficult problems are only starting to appear.

This chapter presents a survey of representative GA, ACO, PSO and EO works and methodologies for dynamic problems. Sections 2 (post 2000), 3, 4 and 5 describe how each of the identified meta-heuristics has been used to solve dynamic optimisation problems. Section 6 comments on the limitations of dynamic techniques.

2 Genetic Algorithms

Genetic Algorithms are generally regarded as the classical biologically inspired optimisation algorithms. Hence they are the predominant representatives of stochastic approaches in the work of researchers over the years, and they also dominate the field of approaches to solving dynamic problems. The concept of GAs was devised by Holland [53]. In essence, each solution to a problem is treated as a chromosome, a population of which are used in the system. Each solution component of the problem under consideration is referred to as a gene. In early GAs, these were nearly exclusively composed of sequences of bits called alleles, though real numbers and integers are now commonly used instead. At each iteration of the algorithm, a subset of the fittest chromosomes (i.e., those having the best objective function values) are permitted to form the basis of the next generation. This is often done by use of the crossover operator, in which gene segments of one individual are replaced by segments of another thus forming a child chromosome. This can be repeated until a new population is formed. Mutation of some gene values also occurs to increase the genetic diversity of the population. Over time, increasingly fitter individuals are produced, though premature convergence may limit the extent of the optimality of the chromosome solutions.

There are numerous adaptations to dynamic optimisation problems and the applications designed until the year 2000 have been reviewed in-depth by Branke [18]. The GA adaptations discussed are mainly concerned with postponing convergence to retain the ability to search the problem space sufficiently widely after a change. The initial approaches up to around the year 2000 include memorising individuals [61, 74], adjusting the mutation rate [1, 24, 25], redundant representation [43, 50, 69], randomisation [44], fitness sharing [60] and multiple populations [18, 88].

2.1 Memory

Explicit memory as a GA variation was first introduced by Ramsey and Grefenstette [74] as a knowledge base of memorised individuals, devised as an approach to detecting and exploiting similarities between problem states. When a problem enters a new state, the similarities with previous states are determined and individuals which are well adapted to these states are reintroduced into the population.

A substantial contribution on the usage of memory in connection with GAs solving dynamic problems has been made by Branke [16]. The initial paper explores a memory-based approach which divides the real-value encoded population into subpopulations. Initially, the population is subdivided into two parts, one of which is randomised after a change. This part of the population contributes individuals to the memory while the other part is responsible for the search, for which it retrieves individuals from the repository. In a second alternative, the population is split into three islands with all populations contributing to the memory while only one of the populations retrieves individuals from this memory. The individuals in the memory are counted as part of the population, which has a constant size. Individuals are retrieved from memory after each detected change. Only the best individual is added to memory from time to time (in this case, every ten generations), and several replacement strategies are explored. The two best-performing strategies propose to replace the most similar individual, alternatively to replace the most similar individual only if it is of worse fitness. The experiments on the Moving Peaks problem, using five peaks, compare the performance of the implementations with one, two and three populations combined with memory as well as a single-population GA without memory. The results show a superior performance of the implementation with two populations when the peaks change their heights but do not move, whereas the three-population variation performs best among the candidates when the peaks change their locations as well as their heights. Branke [16] concedes that memory-based approaches have limited application since they are only truly useful in the case of an exact recurrence of a previous situation but also concludes that increased diversity contributes more to the performance of a memory-based GA than to a standard implementation.

Bendtsen and Krink [8] argue that repetitive patterns are typical for real problems in everyday life, as various scheduling problems typically adhere to daily or weekly patterns. The dynamic memory they employ does not store the generation-best individual directly, but approaches it in real-valued space. The memorised individual closest to the best in the main population is moved towards this best individual by a Gaussian random number with a mean of 0.5. At every generation, the best individual from memory replaces the worst individual in the main population. Individuals in memory that do not have the best individual as a closest neighbour are moved the same distance in a random direction. The authors report that their memory-enhanced GA outperforms the implementation by Branke [16] as well as the standard GA used as a reference. As test cases, they used the moving peaks function developed by Morrison and De Jong [65] and a greenhouse problem.

Karaman, Uyar and Eryiğit [57] employ a memory which characterises the environment as it presents itself just before a change. A key figure defined as the ratio of feasible individuals over all possible individuals identifies the environment, which is then reproduced probabilistically using a frequency array of alleles (essentially a vector of the distributions of ones on the allele locations of all individuals). When the algorithm is notified of a change, it records the current individuals in the distribution array before the individuals are replaced by the closest matching environment. To establish the closest recorded environment, 100 random locations are specified at the beginning of the trial and evaluated after each change according to the new objective function. If no sufficiently close environment is available, the algorithm uses hypermutation to facilitate quick adaptation. The authors explore the algorithm ('MIA') by experimenting on a knapsack problem with 50 different environments (changes). Both a GA with hypermutation and a GA with memory are used as a benchmark. The authors conclude that the devised MIA algorithm performs better the longer the algorithms run, achieving better results after the initial 1000 generations.

Inspired by Ramsey and Grefenstette [74] as well as Karaman et al. [57] Yang [93] further explored the use of memorised individuals which provide information regarding the environment they were initially stored from. The Associative Memory GA (AMGA) algorithm updates its memory, whose size is one tenth of the main population, by storing the best individual every 5–10 generations along with a frequency array of alleles as in Karaman et al. [57]. The individual and its distribution vector replace the closest individual in memory. After a change, the fittest individual in memory is chosen and a definable number of individuals are created from the distribution vector to represent a previous environment. All new individuals are merged with the population and the worst individuals are then discarded until the original population size is restored. The experiments reveal that this algorithm works best when the number of new individuals introduced equals approximately 50% of the size of the main population. Also, AMGA works best when solving cyclic problems in which previous environments recur. It solves problems with random changes to better quality than the standard GA and the memory-based GA. The latter which merges all of the memorised individuals into the main population and keeps the best.

2.2 Memory and Randomisation

Trojanowski and Michalewicz [87] are the first authors cited for combining the memory and randomisation techniques with GAs. Replacing large parts of the population with random individuals after a change and keeping an individual memory are methods of adaptation which are explored separately and in combination. The individual memory is implemented as a memory buffer each individual updates after a generation. In the case of mutation, the previous chromosome is added to the memory buffer; in the case of crossover, the better parent's chromosome history is recorded in the buffer. When the buffer is full, the oldest individual(s) are replaced. After a change, all individuals are re-evaluated, including the chromosomes in the memory buffers. If one of the memorised individuals has a better fitness, it replaces the currently active individual. When randomisation is used in combination with memory, the memory is applied first. After this, some of the active individuals are re-randomised whereas the chromosomes in the buffers remain unaffected.

This approach was tested on a multimodal function using binary-encoded individuals with some cyclic and some random aspects. The authors observed that randomisation works best on small changes or a relatively easy objective function. They conclude that the combination algorithm finds the most optima in the multimodal function, with both the randomised and the memory-based variation outperforming the benchmark GA with standard settings.

Yang [92] used a memory whose size is one tenth of the number of individuals in the main population. After the initial random members populating the memory have been replaced with the main population's best individual every 5–10 generations, the individual closest to the replacing is substituted. If a change has taken place, the best individuals in memory are reintroduced into the main population. Yang proposes to combine this type of memory with random immigrants to be added to the main population every generation. Inspired by Bendtsen and Krink [8], Yang proposes a third variation in which the randomisation and the memory are combined. This algorithm, MIGA, makes random mutations to the best individual in memory instead of introducing random individuals. It is tested on several functions and is found to perform better than the other two variations as well as the existing random immigrants GA introduced by Cobb and Grefenstette [25].

Simões and Costa [78] have explored the use of memory as well as replacement strategies extensively. Initially, they hybridised a GA with two aspects from Immune Systems, clonal selection and hypermutation. Clonal selection is applied to the main population at every generation. The cloned individuals have a preset probability of being subjected to hypermutation, for which gene libraries, initialised randomly and kept constant throughout the generations, are used. A memory population is kept, in which the best individuals from different phases are recorded in combination with the average fitness of the main population at the time they were stored. When changes are detected, the individuals with the closest average fitness number are cloned and reintroduced into the main population to replace the worst individuals.

This algorithm, named Immune System based GA (ISGA), was developed further by Yang [94], who introduced an alignment variable for the gene segments in the library. The goal is to enforce the use of the gene segments in a predefined location of the individual. The library has two kinds of gene segments: A random group to be re-randomised at every generation and a non-random one which contains copies of gene segments and their locations from members of the main population selected by a tournament of two. For the experiments, Yang's [94] dynamic test functions are used. The performance of the devised algorithm is superseded only by a version which uses non-aligned transformation, as in the algorithm of Simões and Costa [78]. Unfortunately, the algorithm is not compared to the original ISGA [78], but to another variation devised by the same authors.

In further work [79, 80, 81, 82], Simões and Costa devised and explored the Variable-size Memory Evolutionary Algorithm (VMEA), which is based on adapting the size of the main and memory populations. However, the memory and main populations only shrink when duplicates are removed. As the populations are often at a maximum, the replacement schemes used are of considerable importance. Two aging schemes and one generational method are proposed. The first aging scheme ages the individuals in memory linearly at every generation. The age is increased more considerably when the individual is reintroduced into the main population and reset to zero when a certain age limit is reached. When a replacement takes place, the youngest individual is removed. In the second aging scheme, devised originally by Branke [16], the individual's fitness contributes to its age as well as the generations. Although the age is never reset in this model, the youngest individual is replaced. The generational scheme prescribes that the worst individual added since the last change is replaced. If no individual has been added, the closest individual by Hamming distance is replaced. The authors found that the generational scheme outperforms the age-based schemes as well as the similarity-based replacement scheme found to be superior in Branke's studies [16].

The second age-based scheme was observed to perform well on problems with frequent changes. Apart from the memory updating schemes, the VMEA algorithm also explores a genetic operator which draws its inspiration from bacterial conjugation. It resembles two-point crossover, except that the gene sequence is copied from the better individual (the donor) to the individual with worse fitness (the recipient) so that the donor remains unchanged. In one variation of the VMEA algorithm, it is used in the place of the crossover operator. In most experiments, conjugation performed better than the crossover operator.

2.3 Randomisation

Motivated by the observation that completely random ‘random immigrants’ may interfere with the solution process when changes are not very severe, Yang [95] designed a GA in which the randomisation of individuals is based on mutations of the current elite. As expected, this algorithm outperforms a GA with ‘random immigrants’ on a dynamic environment with minor (20% and less) changes. Yang [95] goes on to show that if the goal is to cater for both severe and slight dynamics, a combination of elite-based and completely random individuals is a useful choice.

To further explore the potential of randomised elite individuals, introduced by Yang [95], Yang and Tinós [96] devised an algorithm combining three degrees of randomisation. Based on the observation that elite-based individuals are most useful only in an environment undergoing only minor changes, they combine the elite-based individuals with completely random as well as dual individuals. The latter describe the mirror image of existing elite individuals and are considered the appropriate response to the most drastic changes, while completely random individuals are considered the ideal response to changes of intermediate severity. All three groups of immigrant individuals are used in combination and replace the worst individuals after a change. The less successful of the three groups have their numbers reduced until a given limit is reached. Success is measured as the performance of the best individual of each group. The tests [96] demonstrate that a traditional GA with random immigrants is indeed less successful when the changes have a low severity of less than 20% of the environment. The hybrid algorithm performs best when the severity of changes fluctuates, as it uses different types of randomness adaptively.

2.4 Self-Organised Criticality (SOC)

Tinós and Yang [85] have devised a hybrid approach which combines both the randomisation of chosen individuals and the maintenance of a subpopulation. To introduce self-organised criticality [4], their method replaces both neighbours of the worst individual as well as the worst individual itself with random individuals. The neighbours are defined as adjacent entries in the array of individuals. If the new worst individual is not within index range of the previous subpopulation, the old subpopulation is replaced. If it is at the edge of the previous range, the subpopulation will grow, a phenomenon the authors call an extinction event. The new randomised individuals are kept as a separate subpopulation to ensure that they are not eliminated for poor performance before having been given a chance to evolve separately from the other individuals.

The subpopulation is more likely to grow when the general level of fitness in the population is high and the worst individual is likely to be a random individual introduced in the previous generation. If this worst individual has a member of the main population as a neighbour, the neighbour is also replaced, leading to randomisation of a larger neighbourhood. This emulates the SOC paradigm of inducing a critical state when the general level of adaptedness is high. The authors compare this approach to a standard GA and two implementations of a GA with random immigrants, the exact implementations of which are not discussed. The benchmark problems used are dynamic unitation functions, devised by the authors as 'deceptive' functions with local optima. The authors use the adaptability metric introduced by De Jong [26], and find that their hybrid approach outperforms the benchmarks. More extensive tests on the algorithm are presented in further work [86], where the algorithm is referred to as Self-Organising Random Immigrants GA (SORIGA).

2.5 Recent Applications

Chaudhry and Luo [22] provide a survey of GA applications to operational problems published in production and operations management journals in the years 1990–2001. They found that the number of publications is generally rising over the years, with 50% of the GA implementations applied to scheduling problems. The authors do not specify if any of these articles, 178 in total over the years, describe applications to dynamic problems.

Freight train scheduling is the optimisation problem for a two-phase GA application devised by by Godwin, Gopalan and Narendran [42]. In most countries, the freight train system uses the passenger rail network with passenger trains dictating the framework for the possible schedules. The rail network is subdivided into 'blocks', which are railway sections between possible sidings, where trains can pass each other. Locomotives and freight rakes – a set of connected freight carriages without a locomotive – are located within divisions or subsections of the railway network. Relocating a locomotive between divisions means an 'empty' trip and is known as 'deadheading'. The problem has two conflicting objectives, the minimisation of deadheading and the minimisation of coupling delay, effectively the time freight rakes wait for a locomotive.

The authors propose to address the problem in two phases; the first assigns the locomotive and does a preliminary scheduling which places the freight train between the passenger trains; the second phase minimises the arrival time at the destination and deals with conflicting freight train and deadheading schedulings. Both phases use a GA with single-point and two-point crossover, mutation and elitism as well as a local search procedure which explores the immediate neighbourhood of a solution by swapping genes. The performance of the algorithm is not compared with other approaches but it is stated that, on the three hypothetical rail networks used for the experiment, the algorithm produced acceptable schedules.

The problem of inventory management is solved as a strategy problem with sliding time windows by Bosman and La Poutré [15]. The optimisation focuses on strategies to adopt when reordering and determining the size of the orders to place. These are referred to as the reorder point and order-up-to size respectively and are applicable to each of the available suppliers. Variations in lead and delivery times render the problem more difficult. As the decisions prescribed by the strategies influence the succeeding time windows, the authors chose to solve this problem on line with sliding time windows to continuously evolve suitable strategies for different situations. The dynamics are created by the fluctuating lead and delivery times.

The solver optimises the profit, which is a function of the sales revenue decreased by the holding cost and the ordering cost. The more reliable suppliers are assumed to be more expensive. If the stocks fall below a limit, emergency orders have to be placed with more expensive suppliers. The experiments use four scenarios in which the algorithm learns the optimal ordering strategy and clearly improves at doing this over time. In the fourth scenario, the authors introduce the anticipation of 'customer satisfaction' as an increase in the number of sales which depends on the inventory levels. Although an interesting consideration, it may not be a very realistic assumption.

3 Ant Colony Optimisation

ACO is an optimisation paradigm encompassing a range of meta-heuristics based on the evolutionary mechanics of natural ant colonies. Unlike many other biologically inspired optimisation algorithms, ACO, on the whole, represents a set of constructive techniques. This means that each ant at each step of the generalised algorithm adds a component value (such as the next city for the travelling salesman problem (TSP)) to its solution. This decision is based on a combination of the worth/goodness of component values and simulated chemical markers called pheromone. The use of the latter is a collective form of self adaptation. As a result, colonies of ants produce increasingly better solutions over time. Generally, pheromone is used to reinforce good solutions and their component values. However, there is also the capacity to decay pheromone strength, so that the effect of premature convergence is lessened. A comprehensive overview of ACO can be found in Dorigo and Di Caro [35].

As a maturing meta-heuristic, applications to dynamic problems have started to emerge. Apart from the benchmark problem set (including the TSP, quadratic assignment problem (QAP) and knapsack problem), industrial oriented research has been mainly in networks and telecommunications [35] and manufacturing. Scheduling, autonomous robots, continuous optimisation and more generic frameworks for ACO in dynamic environments are also discussed.

3.1 Benchmark Problems

Benchmark combinatorial optimisation problems, such as TSP, QAP and the knapsack problem, are usually presented as static problems. However, as there has been recent interest in adapting ACO to solve dynamic problems, some common problems have been changed so they have a dynamic (temporal) component. In particular, this has been done for the TSP. There are two broad ways in which a standard TSP can be transformed into a dynamic problem. These are: dynamically varying the distances between cities; and adding/dropping cities from the problem while it is being solved between separate runs of the ant colony solver.

Eyckelhof and Snoek [40] solve problems in which the distances between cities vary dynamically. They adapt the basic Ant System algorithm to avoid the twin problems of a) certain edges becoming relatively unused because of low pheromone values and b) some edges having too much pheromone and hence dominating the decision process. This is achieved by effectively limiting the lower and upper bounds on pheromone values, implicitly implementing a form of $\mathcal{MAX}-\mathcal{MIN}$ Ant System. Using two small test data sets of 25 and 100 cities, they found that the algorithm is able to quickly adapt to the changing environment and is an effective solution method for these problems.

In contrast, Angus and Hendtlass [2] solve a problem in which cities are added or dropped rather than dynamically changing distances. The addition and removal of cities occurs in separate runs of the solver. This work showed that the ant algorithm could adapt quickly to the new problem scenario. Upon the removal or addition of a city, the pheromone levels on the problem edges were normalised.

A more intricate approach is that by Guntsch, Middendorf and Schmeck [48]. Groups of cities are replaced by new cities at different frequencies. This differs from two of the authors' previous work [45] which only considered a single change to a problem instance. The approach they use for this more complex version of the problem is twofold – using pheromone modification mechanisms and a heuristic called *KeepElite* for modifying solutions to be suitable for new problem definitions. In terms of the former, three strategies were trialled which variously modified pheromone values by resetting them to the initial value or adjusting them in proportion to either the distance or existing pheromone level from the newly inserted/deleted city. It was found that a combination of resetting the pheromone and proportional adjustment worked best. This combined with the *KeepElite* heuristic had better performance than using the pheromone adjustment strategies in their pure form.

3.2 Network and Telecommunications Problems

Solving dynamic network and telecommunication problems, which include such diverse areas as routing telephone calls and the regulation of network traffic flow, have been the subject of most research.

Schoonderwoerd, Holland, Bruten and Rothcrantz [77] use a highly abstracted ant based system for solving problems involving the symmetric forward routing of telephone calls between origin and destination nodes using a switched network. The networks they study are not capable of connecting all calls at a given time so the objective is to minimise the number of lost calls. In their ant based solution, ants operate independently of the calls. Each ant travels between an origin and a random destination node based on a function of pheromone, distance and node congestion. Calls are routed according to the pheromone level on each neighbouring node. The link with the maximum pheromone is always chosen. This approach compares favourably with existing mobile agent methods of British Telecom [77]. Heusse, Snyers, Guérin and Knutz [52] extended Schoonderwoerd et al.'s [77] work to asymmetric networks that are capable of dealing with changing bandwidth and topology requirements.

Saleh and Ghani [76] create a system called the For/Backward approach which combines epochal updates (i.e., updates of routing tables only after an ant has moved forward and backward through the network) and incremental updating (in addition, ants also update the tables as they traverse the network). Applying this to a simulation of the Malaysian backbone network showed that a small packet delay and a greater bandwidth throughput could be achieved.

The problem solved in Xia, Chen and Meng [91] is one in which a composition of web services, that themselves are composed of component services, is required. Each service has a quality of service (QoS) component and different users have different preference weightings on each QoS. The objective is to maximise users' satisfaction with the supplied services. The graph that the ants walk is formed from the services and changes as new services are added, old ones removed and QoS values change. A multiple pheromone strategy embedded in a dynamic ACO, where different pheromones tracked different QoS attributes, was used. In comparison to a GA and a standard single pheromone ACO, the authors' modified algorithm provided better performance on a single simulation problem.

AntHocNet [34] is designed specifically to route packets in mobile ad-hoc networks and is an updated version of the work in Di Caro, Ducatelle and Gambardella [33] and Ducatelle, Di Caro and Gambardella [38]. Two kinds of ants known as forward and backward ants are used to construct the initial routing tables. The former are used to locate all the practicable paths within the network while the latter move backwards from the destination and establish the paths. Operationally, data packets are routed stochastically and link failures may occur. In order to manage the dynamic nature of the network, another set of ants, called proactive forward ants are used to maintain and improve the proactivity of paths. The authors note that while AntHocNet has very good performance in terms of its ability to successfully route packets, its computational overheads are more than other algorithms (such as the simpler relative routing algorithm of Perkins and Royer [73]). This is mainly due to the number of different ants.

Submarinian, Druschel and Chen [83] present two ant colony algorithms to solve a network routing problem in which network topology and link cost changes occur dynamically and unpredictably. Their first algorithm uses a simple shortest path approach and assumes symmetric path costs. Their second algorithm is a more robust version which is capable of multi-path planning and does not require symmetric path costs. Each algorithm uses slightly different ants. The first is considered standard as the ants will likely follow good non-congested routes as given by the forwarding tables. The ants of the second algorithm effectively ignore the forwarding tables and all paths have equal probability. In addition to having ants that move forward from source to destination, another set of ants are sent backward from the destination node to the source node with the aim of exploring the network as well as the routing tables of the routers that they encounter. Both algorithms perform better than standard approaches when there are higher network failure rates.

Di Caro and Dorigo [27, 28, 30, 29, 31] have designed an ant colony system (ACS) to build forwarding tables for packet routing problems (such as those that occur on the Internet). This system is known as AntNet. Like other approaches, the algorithm consists of two types of ants, the forward ants and the backward ants. Each forward ant constructs a path from a source node to its destination nodes and collects information about about the time it takes to move from node to node. The corresponding backward ant then moves from the destination to the source node, updating the routing tables of each of the nodes it visits. The authors have shown that AntNet compares favourably with existing routing algorithms. Zhang, Khun and Fromherz [98] have made substantial adaptations to the system to accommodate ad-hoc wireless sensor networks.

Di Caro and Dorigo [32] apply two versions of AntNet to the problem of adaptive learning in dynamic datagram networks. A datagram network is considered as an unreliable network as the user is not notified if the delivery of a packet fails. The first version of AntNet is denoted AntNet-CL and is the canonical form of the algorithm while AntNet-CO is specifically adapted to best-effort routing in datagram networks. Using simulations of heavy traffic networks, the results showed that AntNet-CO lead AntNet-CL in terms of performance (i.e., throughput, packet delay and resource utilisation). Both algorithms were better able to handle these conditions compared to five standard Internet routing algorithms.

Similar work to AntNet has also been undertaken by White, Pagurek and Oppacher [90] except that separate ant colonies are used to determine the routes, allocate traffic to the routes and deallocate routes. Preliminary work has also been carried out on routing with fibre optic cables [89].

3.3 Industrial Manufacturing

Some of the most common problems in industrial manufacturing are job shop scheduling and machine sequencing.

Cicirello and Smith [23] solve a dynamic problem of routing jobs on a shop floor using an algorithm based on ACO principles. In their approach, each job is assigned to an ant that makes the routing decisions through the various machines on the shop floor. Pheromone is deposited on the route that each ant/job takes. This was shown to effectively balance the workload of the factory machines.

Aydin and Öztemel [3] describe a system for solving dynamic job shop sequencing problems using intelligent agents. In this paper, an agent does not solve a complete problem, but simply reacts to the requests from a simulated environment and develops an appropriate job priority list. There are two stages; a learning stage and a production stage. In the learning stage, the simulated environment gives the agent feedback about the performance which it then uses as a part of a reinforcement learning program. Arrival times and processing durations on particular machines for particular jobs are given by an exponential distribution. However, machine breakdowns and other events (such as rush orders) are not dealt with by this approach.

3.4 Scheduling

Gutjahr and Rauner [49] present an ACO algorithm to solve a regional nurse scheduling/rostering problem for the consolidated hospital system in Vienna, Austria. This is modelled as a constraint satisfaction problem that takes nurse preferred working days, working patterns, employee satisfaction and hospital costs into consideration. The hard constraints (such as the legal number of hours a nurse can work each week) must be satisfied within the system, whereas the soft constraints do not. The dynamic components of the problem come about as a result of different day-to-day requirements and the necessity of the system to discourage delaying resource-based decisions until the last possible day. The latter is achieved by using an urgency penalty in the problem model. Using a simulated environment, it was shown that ACO could deliver schedules that incurred significantly less operational costs and higher nurse satisfaction than those that could be produced by a greedy algorithm.

3.5 Autonomous Robots

In many ways, this category is an extension to scheduling and network problems. Detection of targets in a dynamic environment is a problem considered by Meng, Kazeem and Muller [64]. A hybrid ACO/PSO algorithm is developed for the control of many small scale robots (the number of which may itself vary dynamically). In essence, robots communicate with one another using virtual pheromone, but may only do so if they are in a permissible range. Each robot holds its own pheromone matrix that it modifies over time. The swarm intelligence component of the hybrid is concerned with optimally organising the behaviour of the robots at a global level. These concepts working in combination allow agents to make decisions about the direction in which they move. The factors taken into consideration include each target's utility value (e.g., attractiveness and distance separation), and the agent's inertia, experience, and interactions with other agents. Simulation results showed that the hybrid ACO/PSO outperformed, and was more consistent than, a standard ACO implementation.

The work of Pekala and Schuster [72] differs from the above as a heterogeneous swarm of robots is controlled by the ACO algorithm. The aim of the research is to 'develop a method or algorithm that can control a heterogeneous swarm in an optimal manner to accomplish any particular task without human intervention' (p. 354). Each agent/ant represents a robot and attempts to complete a set of jobs which then makes up a task. The different types of robots in the swarm can complete only certain job types. One of the distinguishing factors of the algorithm is its ability for robots to learn the order in which tasks should be completed using a quality matrix metric. The dynamic element of their Modified Dynamic ACO (MDA) allows the addition and deletion of jobs. The simulation environment used to test MDA models the scenario in which food must be brought to a kitchen and then stacked. In this application, robots may scout for food, transport it to the kitchen, or stack the food items. The simulation showed that the system's ability to learn appropriate job orders made it highly effective for this problem. In addition, partially disabled robots were still able to make an effective contribution to solving the problem.

3.6 Continuous Optimisation

Beyond discrete dynamic optimisation problems, ACO has also been applied to continuous problems. These have generally been solved by genetic algorithms and particle swarm optimisation. However, Dreo and Siarry [36] propose the hybrid algorithm *Dynamic Hybrid Continuous Interacting Ant Colony (DHCIAC).* In addition to pheromone, ants also pass messages about search progress directly to any of the other ants in the colony. The authors implemented two versions of DHCIAC that are differentiated by the local search algorithm. $\text{DHCIAC}_{\text{find}}$ uses the Nelder-Mead simplex algorithm while $\text{DHCIAC}_{\text{track}}$ uses a modified simplex algorithm known as dynamic simplex. These effectively implement diversification and intensification strategies respectively. It was found that $\text{DHCIAC}_{\text{track}}$ was better for problems that had a faster rate of change. Overall, DHCIAC generally appears to be useful for slow changing problems with many local optima – though comparative results with other algorithms were not provided.

3.7 General Approaches

There have only been limited attempts to modify standard ACO algorithms to process dynamic problems more seamlessly.

Population ACO (P-ACO) [46, 47] is an ACO strategy that is capable of processing dynamic optimisation problems. It achieves this by using a different pheromone updating strategy. Only a set of elite solutions are used as part of the pheromone updating rules. At each iteration, one solution leaves the population and a new one (from the current iteration) enters. The candidate solution to be removed can be selected on age, quality, a probability function or a combination of these factors. The authors argue that this arrangement lends itself to dynamic optimisation, as extensive adjustments, due to the problem change, need not be made to the pheromone matrix. Instead, a solution modified to suit the new problem is used to compute the new pheromone information. This modification process works for full solutions only and is tailored for particular problems. Experimental work on the dynamic TSP and QAP showed that P-ACO could adapt to small changes in the problem better than restarting the entire algorithm.

A set of generic modifications have been proposed to allow ACO to solve dynamic optimisation problems [75]. This framework defines categories of events that change the problem definition (i.e., data and/or structure) while ACO solves it. Rather than discarding solutions and restarting the construction process, if an event occurs, the process of deconstruction begins. Deconstruction removes the components from a solution that make it infeasible to the changed problem. Results for a dynamic version of the multidimensional knapsack problem showed that the modified ACO could quickly adapt to the problem changes. Further work on the dynamic aircraft landing problem [41] (using a real-time simulator) indicates that the approach is capable of producing good schedules in a timely fashion.

4 Particle Swarm Optimisation

The classical particle swarm optimisation algorithm [58] was inspired by the swarming behaviour of birds and fish. While the metaphor is a dynamic one, there have been few applications to dynamic problems. PSO tries to mimic the natural behaviour and applies it to a number of particles moving through problem space. This movement occurs under the influence of a number of factors, some of which can be considered personal in that no other particle is involved while others are social and involve more than one particle.

When the velocity of each particle is updated, the particle keeps a part of its velocity at the last iteration. The fraction that is kept, referred to as the momentum (or alternately as the inertial weight) of the particle, prevents any drastic velocity changes and may permit the particle to pass through a local optimum. This is clearly a personal factor. A further possible personal influence is a tendency to return to the best position yet found by this particle (*pbest*).

Apart from the personal there are also social influences. The particle is attracted towards the best position currently experienced by other particles that forms its local neighbourhood (*lbest*) and/or towards the best position found by any particle in the swarm so far (*gbest*). The neighbourhood of a particle can be defined by those other particles with some distance D of this particle. Alternatively, since all particle must have an identifying index, a neighbourhood can be defined as all particles whose indices are within a user specified number of this particles index. Since the particles in a neighbourhood move closer together these two approaches tend to merge after a period of time.

Each particle updates its velocity simultaneously, normally using some combination of personal and social influences. Momentum is almost always used and generally two others, at least one of which must be social. Using *pbest* and *lbest* (a less greedy form) encourages the parallel exploration of multiple local optima while using *gbest* and *lbest* (a more greedy form) encourages the whole swarm to converge on the best optimum encountered. Using *pbest* and *gbest* is also a viable option as experience shows that the number of particles that constitute a local neighbourhood is not critical and may be reduced to one. It is the interplay of the chosen influences which produces an efficient search mechanism.

4.1 Adapting PSO for Dynamic Problems

There are a number of non-biological adaptations that need to be made to the classical swarm algorithm so that it suits dynamic problems. These can be summarised as: preventing the complete convergence of the swarm, keeping personal and social reference points up to date and maintaining or generating explorer particles far from any current point of convergence. Approaches that achieve at least one of these aims will be considered.

4.1.1 Preventing Total Convergence

Social influences between particles, attractions to *gbest* and *lbest*, will tend to result in total convergence. To change this it is necessary to introduce some counter influence. One method [10] is to give at least some particles a charge so that, by analogy with electrostatics, two particles experience a repulsive force as they approach and the swarm would then not be able to fully converge. The particles would in time reach some (possibly dynamic) equilibrium between the convergence and divergence effects, but this does not mean that they are actively exploring. A second method [19] is to divide the swarm into sub-swarms so that not all particles converge on the same point. A particle and its closest neighbours may form a sub-swarm if the variance in the fitness of the particles is less than some threshold. Any particle that is not a member of a sub-swarm belongs to the main swarm. These sub-swarms may merge, acquire extra particles from the main swarm or collapse back into the main swarm. Whilst developed for multi-modal functions this niching behaviour could also be used, in principle, to limit total swarm convergence. However the algorithm depends on a uniform distribution of particles in the search space, a condition that may be able to be met after initialisation but which is not met after convergence into the sub-swarms has taken place. An alternative approach to niching is described in Bird and Li [9].

4.1.2 Refreshing the Best Positions

If an attraction to *pbest* is being used these best positions may be updated by allowing particles to replace their previous best position with the current position periodically [20]. Choosing a suitable period without detailed knowledge of the problem being optimised can be problematic. If an attraction to *gbest* is being used, the fitness at this position may be periodically re-evaluated [54]. As the fitness at that point deteriorates, the probability that it will be replaced by another position as a result of the current fitness at that position increases. Again a suitable re-calculation frequency has to be chosen.

4.1.3 Forcing Explorer Particles

The simplest approach just requires that a number of particles be periodically moved to randomly chosen points and have their fitness re-evaluated [21]. Another approach organises particles in a tree with each particle being influenced by the particle above it (social) and itself (best position and momentum). A particle swaps with the one above it if it out performs it. This gives a dynamic neighbourhood that does require extensive calculation. This has been adapted to dynamic problems by Janson and Middendorf [55, 56]. After the value of the best-known position (*gbest*) changes (it is re-evaluated every cycle) a few sub-swarms are reinitialised while the rest are reset (have their old personal best information erased and replaced with the current position). The sub-swarms then search for the new optimum. Blackwell and Branke [11] introduce a more elaborate approach using quantum particles. Using an analogy to quantum mechanics, a particle on measurement is placed randomly within a given radius of its net current point of attraction. A uniform distribution is used and a function chosen so that a finite probability exists of a movement to a distance far from the point of attraction.

4.1.4 Meeting All Three Requirements

The approaches described above could, if used in combination, be used to track dynamic problems. However, one further approach (WoSP) [51] has the ability to meet all three requirements simultaneously by altering the normal PSO requirement to inhibit total convergence to one that reinforces the tendency to totally converge.

The approach was originally developed to sequentially explore an arbitrarily large number of optima. An extra short-range force of attraction was added to the basic swarm equation. As a result of the discrete way in which fitness evaluations and updates to the velocity of the particles is done, an aliasing effect causes pairs of particles to approach, pass each other and then continue on at very high velocities. The probability that this will happen increases with decreasing distance between the particles. Particles are most likely to be at close range when the swarm converges. There is no longer a need to stop the particles fully converging. As the velocity with which the particles leave the swarm is variable, exploration can continue both locally and at a distance. The total swarm is automatically broken into a number of sub-swarms called waves, each with its own *gbest* value. Particles that leave a converging swarm as a result of this aliasing effect leave the wave they were in and join the highest numbered wave (creating a new one if no higher numbered wave exists). A form of evolution takes place with lower performing waves being compulsorily recruited into better performing higher numbered waves. Waves that run out of particles (owing to promotion or recruitment) die out. In this way there is a continual automatic updating of best position information available to the successive waves.

The main difference between the algorithm for static and dynamic problems is that in the former each particle keeps a tabu list of places that it has already explored and was repelled from any place on its tabu list. In this way re-exploration of any point in problem space is largely (but not totally) eliminated. For dynamic problems this tabu list can be completely removed on the grounds that any particular point in problem space may be a good optimum at several disjointed times. Alternatively extending the idea from Janson and Middendorf [56], each of these previously explored optima could be periodically re-examined and only those points whose fitness had significantly changed are removed from the tabu lists. It is not clear at this stage how the evolutionary pressure that is an important part of WoSP would respond to dynamic problems. Clearly if a number of waves were generated and completed their existence (in the time it took for the dynamic problem to change significantly), evolution would have time to make a contribution. Should the problem change much faster than this, it is likely that the contribution evolution would be able to make would be, at best, limited.

Another integrated approach, the Unified Particle Swarm Optimising scheme (UPSO) was proposed for static environments by Parsopoulos and Vrahatis [70]. It is based on a PSO with constriction factor which uses both a global and a local best. The velocities for these two search directions are unified into a single direction balanced by a factor $0 \leq u \leq 1$. The combined velocity is formed using a weighting u for the global best and $(1 - u)$ for the local best. The scheme also proposes the contribution of a Gaussian random displacement to either the local or the global best particle as they form the combined velocity. The authors observe a natural balance between exploration and exploitation in this algorithm, suggesting it may resist premature convergence. Parsopoulos and Vrahatis [71] therefore conducted some experiments on UPSO in a dynamic environment. Five well-known function optimisation problems, the Sphere, Rosenbrock, Rastrigin, Griewank and Schaffers functions are adapted to exhibit a global optimum with Gaussian random fluctuations. The results were measured as the mean of the best positions over the iterations. The authors find that their approach, when used with $u = 0.2$ or $u = 0.5$ performs significantly better than algorithms which use the influence of either the global or the local best particles exclusively (i.e., when u is set to 1.0 or 0.0 respectively).

4.2 Some Industrial Applications

As is evident in the previous discussion, a relatively large amount has described how PSO may be adapted so that it is generally suited to dynamic optimisation problems. On the whole, far fewer works have yet been devoted to applying PSO beyond benchmark test functions. Some novel applications are described below.

Using PSO to optimise the performance characteristics of wireless local area networks has been undertaken by Kókai, Christ and Frühauf [59]. Their work showed that an adaptive PSO implementation, based on the framework of Hu and Eberhart [54], could outperform a genetic algorithm and neighbourhood search in a simulation environment. The simulator modelled participants in the network shifting positions, failing transmitters and competition amongst transmitters. The system was able to adapt to these changes in the specified time interval of 1 ms so that communications would not break off. An additional feature of this work was that the PSO was directly implemented in Field Programmable Gate Array (FPGA) hardware.

In a different area of communications, Zhang, Li, Zhou, Shen, Zhang, Zhang, Wu, Yuan, Chen, Zhang, Yao and Yang [97] develop a two stage process for adaptive compensation of polarisation mode dispersion used in communication along fibre optic cables. The degree of polarisation in these links changes dynamically because of factors such as environmental temperature. This in turn affects the search space of the problem. Once a global optimisation phase of the problem is complete, the extent of the change is analysed and may invoke the tracking algorithm. This algorithm uses PSO to search around the previously found global best solution so as to track changing optima. From this the authors found that the response time for adapting to disturbances in the system was in the order of a few milliseconds – a good result for this application.

Muller, Monson and Seppi [68] turn their attention to an economic problem of pricing items in a dynamic and noisy environment, such as that which occurs in e-commerce. The aim is to maximise product prices in the midst of fluctuating demand. The authors propose a form of PSO, called P-Best Decay PSO (PBDPSO). Rather than resetting particles and losing valuable information about the search space, a decay function is used so that particles are attracted to new areas of space, despite the noise in the system. In a dynamic pricing simulation, this PSO compares very well to a the Kalman Filter, an approach traditionally used for this problem.

5 Extremal Optimisation

The paradigm of self-organised criticality [4] explains a wide range of natural dynamical systems and has been previously mentioned in relation to genetic algorithms. EO [12, 13, 14] uses elements of SOC [4] by replacing the worst individual (in this case a solution component) in a population by a random one. Over a number of iterations it is expected that the overall solution quality will increase. The original version mutated the worst component only. The absence of noise made the solver very deterministic and vulnerable to local minima. To counteract this, τ-EO was introduced. It assigns each solution component a rank k based on its current fitness within the solution and mutates it according to a probability distribution of $k^{-\tau}$.

Only a few attempts to apply EO to dynamic problems have been made. These use EO's flexible solving mechanism to adapt to underlying fluctuations automatically. EO is guided only by the component fitnesses, which are associated with the objective function. Typically EO algorithms will incorporate subtle changes automatically, as long as they are reflected in the amended objective function.

5.1 The Satisfiability Problem

Menai [63] investigates the use of EO on Incremental Satisfiability (ISAT) problems. Two different types of dynamics are added to static satisfiability problems from the SATLIB library: ISAT1 is obtained by starting from one clause and adding the others one by one, ISAT2 divides the same standard problems into two equal parts and solves the first half before adding the second half of the clauses. Adding the dynamic aspect to the problem seems to increase the challenge for the EO solver. A comparison between the results of EO solving a problem with more dynamics (ISAT1) and its application to a problem with a single change (ISAT2) corroborates this assumption.

The performance of EO is compared with a variation of the WalkSAT algorithm described in detail in McAllister, Sellman and Krautz [62]. Unfortunately, the benchmark algorithm (called R-Novelty) is only applied to the static case of the 29 problems used. This is somewhat surprising as the algorithm is very similar to EO and can be adapted to dynamic problems as easily as EO. Compared to R-Novelty, EO has a better average success rate when solving ISAT2. Solving ISAT2, EO uses approximately as many iterations as R-Novelty takes to solve the static problem.

5.2 A Defense Application

The Swedish Defense Research Agency investigated target recognition in which increasing numbers of moving sensors produce reports that may or may not contain information on the targets. Fast optimisation techniques are necessary to cluster the incoming reports according to the objects, before the relevant information on target objects can be processed by a tracker module. Svenson [84] compares EO and τ-EO on this task, which requires the ability to include bursts of incoming reports whenever they arrive.

The experiment on clustering the incoming reports using the EO algorithm stops short of adding reports dynamically. It only observes that EO performs better with $\tau = 1.5$ than when setting $\tau = \infty$, and that the pairwise comparison of a smaller subset of records to establish the current fitness of the report within a cluster leads to a better result than the evaluation of a larger subset of pairs of reports.

5.3 Dynamic Composition Problem

In order to test the EO solver's capabilities at solving dynamic problems, Moser [66] especially designed the composition problem as a dynamic knapsack-like combinatorial problem, where items of different types have to be added to a solution until a predefined limit is reached. The item's cost attribute relates to the solution's capacity limit, whereas the value or benefit attribute contributes to the solution's quality which is maximised.

The EO solver optimises the quality by improving an initial random solution. A uniformly random component is picked for removal from the solution and replaced with a component not currently in the solution which is chosen according to the power-law distribution.

The problem is made dynamic by introducing various changes, such as changing the values of the cost and benefit attributes of some components or by removing some of the available items from the problem space. Other problem instances prescribe the change of the percentage of a type of item that has to be represented.

The experiments confirm that the iterative aspect of the EO solver has certain disadvantages – large jumps in the search space are virtually impossible as the number of possible deteriorating moves is limited. However, the EO solver is able to adapt to a change very quickly, producing good quality solutions within few iterations. Sometimes a change of the underlying problem helps the solver produce a better solution, as it enables a substantial change in the composition of the solution, which could not have been achieved by the iterative algorithm itself. Given the simplicity of the algorithm, it scales well with the number of components. It is able to handle solutions with over a hundred components having a thousand components available.

5.4 'Moving Peaks' Dynamic Function Optimisation

Dynamic function optimisation was explored by Moser [66] using the moving peaks benchmark function [17]. The benchmark consists of several scenarios in which randomly placed peaks move in a multi-dimensional landscape. The peaks change locations as well as height and width around a given average. For the scenarios, exact specifications exist and therefore, comparable results from different approaches are available.

Several solution representations were attempted, leading to different ways of traversing the search space. Due to the nature of the moving peaks landscape, more than one EO solution was employed to keep track of the peaks moving in the landscape. Nonetheless, the results received from the best-performing EO implementation provided no match for some of the PSO solvers. Further experiments showed that the canonical EO implementation is not well suited to the moving peaks landscape. It does not make use of the exploitable features of the function. For example, it does not climb to the top of the hill it found and thus scores suboptimally on the standard performance measure, the offline error.

5.5 Aircraft Landing

The single-runway dynamic aircraft landing problem (ALP) consists of a sliding time window with aircraft reporting to the air traffic control for landing. According to the formulation by Beasley [6], the aircraft report target landing times. The problem quality is measured according to the penalties for aircraft landing before or after the target times. Optimising the schedule of a sequence of aircraft is straightforward once the order of aircraft is known. Realistically, there are not many aircraft in the active time window at any given time. The problem instances provided as benchmarks by Beasley [5] have window lengths between 2 and 18 aircraft.

The EO adaptation devised by Moser [67] treats the ALP as a permutation problem of the aircraft in the current time window. The EO solver produces candidates for a new solution using pairwise swaps of aircraft. All candidates have their schedules optimised deterministically before their costs are evaluated. The new solution is then chosen among the candidates. The new solution is compared to the current best-known solution. Aircraft are removed from the active time window when they are within a given distance from their scheduled landing times.

The EO solver produced ideal schedules for each individual time window in all cases in which the optimal schedule was known. Speeding up the arrival of aircraft to shorten the intervals to one fifth did not change the results.

6 Tracking Frequency Limitations

The previous sections have demonstrated that all four algorithms have the capacity to solve dynamic optimisation problems. This last part of this chapter considers what determines the maximum rate at which changes can occur before the algorithm performance essentially is reduced to that of random search.

For population-based algorithms that use a history of previous performance to guide future search, such as ACO, GAs and PSO, the obvious limitation comes from how fast they can learn to disregard the now irrelevant part of their history. For population-based ACO this either implies a short history list or a method of detecting and removing all historic paths that are no longer valid. For non-population based ACO this will require careful handling of the evaporation to deposition ratios in the time immediately after a change has been detected. For PSO either the *gbest* value needs to be periodically re-evaluated and a reset of it and all *lbest* (or *pbest*) positions made as soon as *gbest* alters or, alternatively a sequence of sets of *gbest* and *lbest* (or *pbest*) values have to be used (as in WoSP [51]) so that changes can be responded to without having to be explicitly detected.

For the single individual algorithm EO there is no explicit historical information used to guide future exploration[1]. Instead the algorithm will detect a change when the current solution suddenly becomes invalid. The current stored best solution then has to be cancelled and a special repair procedure will then have to be performed on the current individual. This will take a number of changes and after this enough time must be allowed so that a good solution can be built up.

It is possible to describe the factors that will determine the time it will take any of these algorithms to respond to a change and again have a good valid solution. However, the stochastic nature of the algorithms (amongst other factors) makes it impossible to quantify what this delay will be, thus allowing the maximum rate of problem change to be specified. Therefore, some limiting rate must exist, albeit it problem and algorithm dependent. Should any of these algorithms be used on a problem changing faster that the relevant limiting rate, the effect will be, that the algorithm makes too infrequent a sampling of problem space. By analogy to the aliasing effect seen when a digital data stream is sampled below the Nyquist-Shannon frequency when high frequencies appear as lower frequencies, it can be predicted that the algorithms may well present solutions to problems that, in fact, have never been posed. Worse than getting wrong results, there may be no clear indication that this has most undesirable effect has occurred. This suggests that either practical experimentation involving varying the frequency of known problems, or more ideally an analytical analysis, should be undertaken so that an estimate of the limiting rates of change the algorithms can handle can be established.

Until this is done, since the limiting change rate for the four algorithms discussed in this chapter are almost certain not to be the same, one pragmatic solution may be to run two or more of the algorithms in parallel, only accepting the solutions when they are at least similar. However, in the long run this is hardly practicable and no substitute for a fuller understanding of how these algorithms work and thus of their limiting features. For all the progress made so far in applying these algorithms to dynamic problems much yet remains to be done.

[1] As a result of the fitness build up and collapse cycle inherent in the behaviour of EO, it may be that the currently best known solution was found some time ago and could thus be considered historic. However, since the development is unaware of this best value and always modifies the current individual there is no historic information used as there is with ACO, GAs and PSO.

References

[1] Angeline, P.: Tracking extrema in dynamic environments. In: Angeline, P.J., McDonnell, J.R., Reynolds, R.G., Eberhart, R. (eds.) EP 1997. LNCS, vol. 1213, pp. 335–345. Springer, Heidelberg (1997)

[2] Angus, D., Hendtlass, T.: Ant Colony Optimisation Applied to a Dynamically Changing Problem. In: Hendtlass, T., Ali, M. (eds.) IEA/AIE 2002. LNCS (LNAI), vol. 2358, pp. 618–627. Springer, Heidelberg (2002)

[3] Aydin, M., Öztemel, E.: Dynamic job shop scheduling using reinforcement learning agents. Robotics and Autonomous Systems 33, 169–178 (2000)

[4] Bak, P., Tang, C., Wiesenfeld, K.: Self-organized criticality: An explanation of 1/f noise. Physical Review Letters 59, 381–384 (1987)

[5] Beasley, J.: OR-Library: Distributing test problems by electronic mail. Journal of the Operational Research Society 41(11), 1069–1072 (1990)

[6] Beasley, J., Krishnamoorthy, M., Sharaiha, Y., Abramson, D.: Scheduling aircraft landings – the static case. Transportation Science 34, 180–197 (2000)

[7] Beasley, J., Krishnamoorthy, M., Sharaiha, Y., Abramson, D.: The displacement problem and dynamically scheduling aircraft landings. Journal of the Operational Research Society 55, 54–64 (2004)

[8] Bendtsen, C., Krink, T.: Dynamic memory model for non-stationary optimisation. In: Proceedings of the Congress on Evolutionary Computation, pp. 992–997 (2002)

[9] Bird, S., Li, X.: Adaptively choosing niching parameters in a PSO. In: Proceedings of the 8th Conference on Genetic and Evolutionary Computation, pp. 3–10 (2006)

[10] Blackwell, T., Bentley, P.: Dynamic search with charged swarms. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 19–26 (2002)

[11] Blackwell, T., Branke, J.: Multi-swarms, exclusion, and anti-convergence in dynamic environments. IEEE Transactions on Evolutionary Computation 10, 459–472 (2006)

[12] Boettcher, S., Percus, A.: Extremal optimization: Methods derived from co-evolution. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 825–832. Morgan Kaufmann, San Francisco (1999)

[13] Boettcher, S., Percus, A.: Nature's way of optimizing. Artificial Intelligence 119, 275–286 (2000)

[14] Boettcher, S., Percus, A.: Optimization with extremal dynamics. Physical Review Letters 86, 5211–5214 (2001)

[15] Bosman, P., La Poutré, H.: Inventory management and the impact of anticipation in evolutionary stochastic online dynamic optimization. In: Proceedings of the Congress on Evolutionary Computation, pp. 268–275. IEEE Computer Society Press, Los Alamitos (2007)

[16] Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the Congress on Evolutionary Computation, pp. 6–9. IEEE Computer Society Press, Los Alamitos (1999)

[17] Branke, J.: The moving peaks benchmark (1999), http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/

[18] Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, Norwell (2001)

[19] Brits, R., Englebrecht, A., van der Bergh, F.: A niching particle swarm optimiser. In: Proceedings of the Asia Pacific Conference on Simulated Evolution and Learning, pp. 692–696 (2002)
[20] Carlisle, A., Dozier, G.: Adapting particle swarm optimization to dynamic environments. In: Proceedings of the International Conference on Artificial Intelligence, pp. 429–434 (2000)
[21] Carlisle, A., Dozier, G.: Tracking changing extrema with adaptive particle swarm optimizer. In: Proceedings of the World Automation Congress, pp. 265–270 (2002)
[22] Chaudhry, S., Luo, W.: Application of genetic algorithms in production and operations management: A review. International Journal of Production Research 43(19), 4083–4101 (2005)
[23] Cicirello, V., Smith, S.: Ant colony control for autonomous decentralized shop floor routing. In: Proceedings of the 5^{th} International Symposium on Autonomous Decentralized Systems, pp. 383–390. IEEE Computer Society Press, Los Alamitos (2001)
[24] Cobb, H.: An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical Report AIC-90-001, Naval Research Laboratory, Washington, USA (1990)
[25] Cobb, H., Grefenstette, J.: Genetic algorithms for tracking changing environments. In: Proceedings of the 5^{th} International Conference on Genetic Algorithms, pp. 523–530. Morgan Kaufmann, San Francisco (1993)
[26] De Jong, K.: An analysis of the behaviour of a class of genetic adaptive systems. PhD dissertation, University of Michigan (1975)
[27] Di Caro, G., Dorigo, M.: AntNet: A mobile agents approach to adaptive routing. Tech. Rep. IRIDIA/97-12, Université Libre de Bruxelles, Belgium (1997)
[28] Di Caro, G., Dorigo, M.: An adaptive multi-agent routing algorithm inspired by ants behavior. In: Proceedings of 5^{th} Annual Australasian Conference on Parallel Real Time Systems, pp. 261–272 (1998)
[29] Di Caro, G., Dorigo, M.: Ant colonies for adaptive routing in packet-switched communications networks. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 673–682. Springer, Heidelberg (1998)
[30] Di Caro, G., Dorigo, M.: AntNet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research 9, 317–365 (1998)
[31] Di Caro, G., Dorigo, M.: Mobile agents for adaptive routing. In: Proceedings of the 31^{st} Annual Hawaii International Conference on System Sciences, pp. 74–83. IEEE Computer Society, Los Alamitos (1998)
[32] Di Caro, G., Dorigo, M.: Two ant colony algorithms for best-effort routing in datagram networks. In: Proceedings of the 10^{th} IASTED International Conference on Parallel and Distributed Computing and Systems, pp. 541–546. IASTED/ACTA Press (1998)
[33] Di Caro, G., Ducatalle, F., Gambardella, L.: AntHocNet: An ant-based hybrid routing algorithm for mobile ad hoc networks. 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. 461–470. Springer, Heidelberg (2004)

[34] Di Caro, G., Ducatalle, F., Gambardella, L.: AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. European Transaction on Telecommunications - Special Issue on Self-organisation in Mobile Networking 16, 443–455 (2005)
[35] Dorigo, M., Di Caro, G.: The ant colony optimization meta-heuristic. In: New Ideas in Optimization, pp. 11–32. McGraw-Hill, London (1999)
[36] Dreo, J., Siarry, P.: An ant colony algorithm aimed at dynamic continuous optimization. Applied Mathematics and Computation 181, 457–467 (2006)
[37] Dror, M., Powell, W.: Stochastic and dynamic models in transportation. Operations Research 41, 11–14 (1993)
[38] Ducatelle, F., Di Caro, G., Gambardella, L.: Ant agents for hybrid multipath routing in mobile ad hoc networks. In: Proceedings of Wireless On-demand Network Systems and Services, pp. 44–53 (2005)
[39] Eberhart, R., Kennedy, J.: A new optimizer using particles swarm theory. In: Proceedings of the 6^{th} International Symposium on Micro Machine and Human Science, pp. 39–43 (1995)
[40] Eyckelhof, C., Snoek, M.: Ant systems for a dynamic TSP: Ants caught in a traffic jam. In: Dorigo, M., Di Caro, G.A., Sampels, M. (eds.) Ant Algorithms 2002. LNCS, vol. 2463, pp. 88–99. Springer, Heidelberg (2002)
[41] Gauthier, S.: Solving the dynamic aircraft landing problem using ant colony optimisation. Masters Thesis, School of Information Technology, Bond University (2006)
[42] Godwin, T., Gopalan, R., Narendran, T.: Locomotive assignment and freight train scheduling using genetic algorithms. International Transactions in Operational Research 13(4), 299–332 (2006)
[43] Goldberg, D., Smith, R.: Nonstationary function optimization using genetic algorithm with dominance and diploidy. In: Proceedings of the 2^{nd} International Conference on Genetic Algorithms on Genetic algorithms and their application, pp. 59–68. Lawrence Erlbaum Associates, Inc., Mahwah (1987)
[44] Grefenstette, J.: Evolvability in dynamic fitness landscapes: A genetic algorithm approach. In: Proceedings of the Congress on Evolutionary Computation, vol. 3, pp. 2031–2038. IEEE Press, Los Alamitos (1999)
[45] Guntsch, M., Middendorf, M.: Pheromone modification strategies for ant algorithms applied to dynamic TSP. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 213–222. Springer, Heidelberg (2001)
[46] Guntsch, M., Middendorf, M.: Applying population based ACO to dynamic optimization problems. In: Dorigo, M., Di Caro, G.A., Sampels, M. (eds.) Ant Algorithms 2002. LNCS, vol. 2463, pp. 111–122. Springer, Heidelberg (2002)
[47] Guntsch, M., Middendorf, M.: A population based approach for ACO. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R. (eds.) EvoIASP 2002, EvoWorkshops 2002, EvoSTIM 2002, EvoCOP 2002, and EvoPlan 2002. LNCS, vol. 2279, pp. 72–81. Springer, Heidelberg (2002)
[48] Guntsch, M., Middendorf, M., Schmeck, H.: An ant colony optimization approach to dynamic TSP. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 860–867. Morgan Kaufmann, San Francisco (2001)

[49] Gutjahr, W., Rauner, M.: An ACO algorithm for a dynamic regional nurse-scheduling problem in Austria. Computers and Operations Research 34, 642–666 (2007)

[50] Hadad, B., Eick, C.: Supporting polyploidy in genetic algorithms using dominance vectors. In: Angeline, P.J., McDonnell, J.R., Reynolds, R.G., Eberhart, R. (eds.) EP 1997. LNCS, vol. 1213, pp. 223–234. Springer, Heidelberg (1997)

[51] Hendtlass, T.: WoSP: A multi-optima particle swarm algorithm. In: Proceedings of the Congress of Evolutionary Computing, pp. 727–734. IEEE Press, Los Alamitos (2005)

[52] Heusse, M., Snyers, D., Guérin, S., Knutz, P.: Adaptive agent-driven routing and load balancing in communication networks. Adaptive Complex Systems 2, 1–15 (1998)

[53] Holland, J.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor (1975)

[54] Hu, X., Eberhart, R.: Adaptive particle swarm optimisation: Detection and response to dynamic systems. In: Proceedings of the Congress on Evolutionary Computing, pp. 1666–1670. IEEE Press, Los Alamitos (2002)

[55] Janson, S., Middendorf, M.: A hierarchical particle swarm optimizer. In: Proceedings of the Congress on Evolutionary Computing, pp. 1666–1670. IEEE Press, Los Alamitos (2003)

[56] Janson, S., Middendorf, M.: A hierarchical particle swarm optimizer for dynamic optimization problems. In: Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 513–524. Springer, Heidelberg (2004)

[57] Karaman, A., Uyar, S., Eryigit, G.: The memory indexing evolutionary algorithm for dynamic environments. 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. LNCS, vol. 3449, pp. 563–573. Springer, Heidelberg (2005)

[58] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the IEEE Conference on Neural Networks, pp. 1942–1947 (1995)

[59] Kókai, G., Christ, T., Frühauf, H.: Using hardware-based particle swarm method for dynamic optimization of adaptive array antennas. In: Proceedings of Adaptive Hardware and Systems, pp. 51–58 (2006)

[60] Liles, W., De Jong, K.: The usefulness of tag bits in changing environments. In: Proceedings of the Congress on Evolutinary Computation, vol 3, pp. 2054–2060 (1999)

[61] Louis, S., Johnson, J.: Solving similar problems using genetic algorithms and case-based memory. In: Bäck, T. (ed.) Proceedings of the 7^{th} International Conference on Genetic Algorithms, pp. 283–290 (1997)

[62] McAllester, D., Selman, B., Kautz, H.: Evidence for invariants in local search. In: Proceedings of the International Joint Conference on Artificial Intelligence, pp. 321–326 (1997)

[63] Menai, M.: An Evolutionary Local Search Method for Incremental Satisfiability. In: Buchberger, B., Campbell, J. (eds.) AISC 2004. LNCS, vol. 3249, pp. 143–156. Springer, Heidelberg (2004)

[64] Meng, Y., Kazeem, Q., Muller, J.: A hybrid ACO/PCO control algorithm for distributed swarm robots. In: Proceedings of the IEEE Swarm Intelligence Symposium, pp. 273–280 (2007)
[65] Morrison, R., De Jong, K.: A test problem generator for non-stationary environments. In: Proceedings of the Congress on Evolutionary Computation, pp. 2047–2053 (1999)
[66] Moser, I.: Applying extremal optimisation to dynamic optimisation problems. PhD in information technology, Swinburne University of Technology. Faculty of Information and Communication Technologies (2008)
[67] Moser, I., Hendtlass, T.: Solving dynamic single-runway aircraft landing problems with extremal optimisation. In: Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling, pp. 206–211 (2007)
[68] Mullen, P., Monson, C., Seppi, K.: Particle swarm optimization in dynamic pricing. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1232–1239 (2006)
[69] Ng, K., Wong, K.: A new diploid scheme and dominance change mechanism for non-stationary function optimization. In: Proceedings of the 6^{th} International Conference on Genetic Algorithms, pp. 159–166. Morgan Kaufmann, San Francisco (1995)
[70] Parsopoulos, K., Vrahatis, M.: UPSO: A unified particle swarm optimization scheme. In: Proceedings of the International Conference on Computational Methods in Sciences and Engineering, pp. 868–873 (2004)
[71] Parsopoulos, K., Vrahatis, M.: Unified particle swarm optimization in dynamic environments. 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. LNCS, vol. 3449, pp. 590–599. Springer, Heidelberg (2005)
[72] Pekala, M., Schuster, E.: Dynamic optimization of a heterogeneous swarm of robots. In: Proceedings of the 10^{th} IASTED International Conference on Intelligent Systems and Control, pp. 354–359 (2007)
[73] Perkins, C., Royer, E.: Ad-hoc on-demand distance vector routing. In: Proceedings of the 2^{nd} IEEE Workshop on Mobile Computing Systems and Applications, pp. 90–100 (1999)
[74] Ramsey, C., Grefenstette, J.: Case-based initialization of genetic algorithms. In: Forrest, S. (ed.) Proceedings of the 5^{th} International Conference on Genetic Algorithms, pp. 84–91 (1993)
[75] Randall, M.: A dynamic optimisation approach for ant colony optimisation using the multidimensional knapsack problem. In: Recent Advances in Artificial Life, Advances in Natural Computation, vol. 3, pp. 215–226. World Scientific, Singapore (2005)
[76] Saleh, M., Ghani, A.: Adaptive routing in packet-switched networks using agents updating methods. Malaysian Journal of Computer Science 16, 1–10 (2003)
[77] Schoonderwoerd, R., Holland, O., Bruten, J., Rothkrantz, L.: Ant-based load balancing in telecommunications networks. Adaptive Behavior 2, 169–207 (1996)
[78] Simões, A., Costa, E.: An immune-system-based genetic algorithm to deal with dyamic environments: Diversity and memory. In: Proceedings of the 6^{th} International Conference on Artificial Neural Networks, pp. 168–174 (2003)

[79] Simões, A., Costa, E.: Improving memory-based evolutionary algorithms in changing environments. Technical Report TR2007/004, CISUC (2007)
[80] Simões, A., Costa, E.: Improving memory's usage in evolutionary algorithms for changing environments. In: Proceedings of the Congress on Evolutionary Computation, pp. 276–283. IEEE Press, Los Alamitos (2007)
[81] Simões, A., Costa, E.: Variable-size memory evolutionary algorithm to deal with dynamic environments. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 617–626. Springer, Heidelberg (2007)
[82] Simões, A., Costa, E.: VMEA: Studies of the impact of different replacing strategies in the algorithm's performance and in the population's diversity when dealing with dynamic environments. Technical Report TR2007/001, CISUC (2007)
[83] Subramanian, D., Druschel, P., Chen, J.: Ants and reinforcement learning: A case study in routing in dynamic networks. In: Proceedings of the 15^{th} International Joint Conference on Artificial Intelligence, pp. 832–838 (1997)
[84] Svenson, P.: Extremal optimization for sensor report pre-processing. In: Proceedings of Signal Processing, Sensor Fusion, and Target Recognition XIII, pp. 162–171 (2004)
[85] Tinós, R., Yang, S.: Genetic algorithms with self-organized criticality for dynamic optimisation problems. The IEEE Congress on Evolutionary Computation 3, 2816–2823 (2005)
[86] Tinós, R., Yang, S.: A self-organizing random immigrants genetic algorithm for dynamic optimization problems. Genetic Programming and Evolvable Machines 8(3), 255–286 (2007)
[87] Trojanowski, K., Michalewicz, Z.: Searching for optima in non-stationary environments. In: Proceedings of the Congress on Evolutionary Computation, vol. 3, pp. 1843–1850. IEEE Press, Los Alamitos (1999)
[88] Ursem, R.: Multinational GAs: Multimodal optimization techniques in dynamic environments. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 19–26 (2000)
[89] Varela, G., Sinclair, M.: Ant colony optimisation for virtual-wavelength-path routing and wavelength allocation. In: Proceedings of the Congress on Evolutionary Computation (1999)
[90] White, T., Pagurek, B., Oppacher, F.: Connection management by ants: An application of mobile agents in network management. In: Proceedings of Combinatorial Optimization (1998)
[91] Xia, Y., Chen, J., Meng, X.: On the dynamic ant colony algorithm optimization based on multi-pheromones. In: Proceedings of the 7^{th} IEEE/ACIS International Conference on Computer and Information Science, pp. 630–635 (2008)
[92] Yang, S.: Memory-based immigrants for genetic algorithms in dynamic environments. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1115–1122. ACM, New York (2005)
[93] Yang, S.: Associative memory scheme for genetic algorithms in dynamic environments. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 788–799. Springer, Heidelberg (2006)

[94] Yang, S.: A comparative study of immune system based genetic algorithms in dynamic environments. In: Proceedings of the 8^{th} Conference on Genetic and Evolutionary Computation, pp. 1377–1384. ACM, New York (2006)

[95] Yang, S.: Genetic algorithms with elitism-based immigrants for changing optimization problems. In: Giacobini, M. (ed.) EvoWorkshops 2007. LNCS, vol. 4448, pp. 627–636. Springer, Heidelberg (2007)

[96] Yang, S., Tinós, R.: A hybrid immigrants scheme for genetic algorithms in dynamic environments. International Journal of Automation and Computing 4, 243–254 (2007)

[97] Zhang, X., Li, X., Li, Y., Zhou, Y., Zhang, J., Zhang, N., Wu, B., Yuan, T., Chen, L., Zhang, H., Yao, M., Yang, B.: Two-stage adaptive PMD compensation in 40 Gb/s OTDM optical communication system using PSO algorithm. Optical and Quantum Electronics 36, 1089–1104 (2004)

[98] Zhang, Y., Kuhn, L., Fromherz, M.: Improvements on ant routing for sensor networks. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 154–165. Springer, Heidelberg (2004)

Relaxation Labelling Using Distributed Neural Networks

Jim Austin

Abstract This chapter presents the Relaxation by Elimination methods (RBE) for searching large collections of graph data that has been implemented on a distributed platform and is in daily use for searching a database of molecules. The core of the approach uses an 'inverted' relaxation labelling method that finds a good match of the input data with stored examples. The method is shown to scale linearly with the number of graphs, and to scale linearly under given circumstances to the number of nodes in the graph. Key to the idea is that the system cuts the search time by removing a set of sub-optimal matches leaving those that could match. The system uses arrays of biologically plausible neural networks, Correlation Matrix Memories (CMMs) to store the constraints between the nodes of the graphs being searched. This is coupled to a novel search method. The system is highly parallel. Recently we have developed a parallel Grid enabled computer system (Cortex II) which utilises Digital Signal Processors (DSPs) and Field Programmable Gate Arrays (FPGAs) and have implemented the method on this system. A service for matching small molecules to a molecule database, in which the molecules are represented as attributed graphs, is currently running online. The methods have also been applied to searching trademark databases allowing people to find trademarks that are geometrically similar. The chapter describes the method in detail and its implementation and application. It also brings together work that has appeared separately and presents a new mathematical formulation of the mapping of RBE onto correlation matrix methods.

1 Introduction

Graph based data composed of nodes and edges are widely used data structures in many applications [8]. For example, an image can be represented as a graph where the nodes in the graph represent features and the relations between the nodes represent the distances between the features. Text may be represented in a graph where the nodes are concepts and edges are the relations between the concepts. These find application in representing 2D and 3D images of molecules, maps, faces as well as being used in text analysis and in many other problems. Although representing data in a graph is relatively simple, the problem of searching for

Advanced Computer Architectures Group
Department of Computer Science
University of York
York YO10 5DD
UK

graphs that match an example graph is known to be a computationally difficult problem [8]. Even more problematic is searching on incomplete graphs against large numbers (100,000's) of graphs stored in databases. The task is to find the optimal match of the query graph with those in the database.

The method described in this chapter has been developed within the Advanced Uncertain Reasoning Architecture (AURA) project that has been running at the University of York for over 10 years [5]. The method may be implemented on a large number of parallel machines using grid technology, and has been targeted at application specific hardware developed at York. The AURA graph matcher software embodies the graph matching method and has been applied to search trademark image databases, 3D face databases and 3D molecular databases. The methods have been commercialised by Cybula Ltd. Demonstrations of the method, that search large molecular databases, are available online at www.cybula.com. Although the method described here is posed as a search problem it is a general, fast, heuristic optimisation method.

The core of the approach originated in relaxation labelling methods, developed in computer vision to solve image matching problems, which have their roots in the Waltz constraint optimisation methods used in understanding 2D line images of 3D objects [22]. The basic task is to find which of a set of constraints should be 'relaxed' to allow a problem to be solved. In the graph matching problem the constraints are supplied by the query graph, and the problem, is to find which graph the query graph, will match to in the database. To do this it is necessary to find the largest match between two graphs, by relaxing the smallest number of constraints.

In developing the algorithm described in this chapter the original relaxation methods were modified to (1) operate faster on digital hardware and (2) reduce the computation involved in finding a solution to the constraint satisfaction problem.

Our solution aims to find the graphs that might match by quickly finding those which have a large mismatch between the constraints. Doing this allows those that have few mismatches to be looked at in more detail. This process has been called relaxation by elimination (RBE), as we aim to solve the problem by finding all the graphs that possibly could not match and eliminating them, leaving those that are left as being the candidate matches.

The approach was motivated by considering how sets of biologically plausible neurons could be used to match images. The method is implemented as a Correlation Matrix Memory (CMM) [4] which consists of simple sum and threshold neurons and may be implemented on parallel hardware. The CMM is used to store the constraints between the nodes of the graphs being searched. This is coupled to the RBE method to determine which graph optimally matches the input data. The system is highly parallel.

We have developed a parallel grid enabled computer system called Cortex II [3] which utilises Digital Signal Processors (DSPs) and soft programmable hardware (Field Programmable Gate Arrays, FPGAs) and implemented the method on this system.

To summarise, this chapter adds to our previous publications by giving a complete implementation of a graph matching application, using a neural network inspired approach, implemented using correlation matrix memories on specialised hardware, across a grid to solve a molecular matching problem. In essence, it brings together all our previous work.

The chapter starts by describing the task and the challenges involved. It then describes its relationship to other exhaustive and heuristic methods. The development of the AURA graph match method is then described, by first showing how a simple binary neural network is constructed. The operation of the method on trademark and molecules is then described. The implementation of the methods on dedicated systems (Cortex and PRESENCE hardware) is then considered, along with distribution of the problem over a number of machines in a grid. Finally, the online implementation is described. The style of this chapter is designed to make the subject matter accessible to a wide audience. For a more rigorous treatment of the material the reader is encouraged to consult the references.

2 The Task

This section describes the task and the challenges involved in solving it. The problem investigated in this chapter is the practical application of graph matching.

A graph is made up of a set of nodes (or vertices) and edges (or arcs). Graphs can be directed, where there is a relationship between the nodes that applies in given directions, or they are undirected, where the node relationships exist in both directions. The edges may or may not be directed in our approach, although here we only consider problems using undirected graphs. The nodes in the graph may or may not hold attributes; there may be none, one or many attributes at each node. The more attributes on each node or vertex, the more complex the matching task becomes. In general, the problem here is attributed relational graph matching, where the graphs are possibly incomplete. A graph *isomorphism* is where the nodes and arcs in one graph are mapped to the nodes and arcs in the other graph (a *bijective* mapping). A *subgraph isomorphism* is where a part of one graph is mapped onto some of the nodes and arcs of the other graph. Finally, a *maximum common subgraph* is the maximum set of nodes and arcs that can be mapped between two graphs. There may be sub-optimal mappings, but the aim is usually to find the maximum match between the two graphs [8].

In the applications given here, the problem is a search engine task which contains the problem of graph matching. A database of objects is stored and the task is to find which of these best matches an input object. In this case the input must first be coded as a graph. An ancillary challenge is how to translate the raw input data into graph form. This can substantially affect the quality and speed of the search. As an example, we have implemented search engines for trademarks [1] and molecules [17]. In basic terms the task can be framed as selecting a set of image features that form the nodes of a graph. The distance between these features is represented as the edges between the graphs. In this case the graph is undirected as the distance relation between two nodes does not have a directional component. Other relations could be size, where nodes represent sub objects of the image and the relations represent if one object is larger than another. This is clearly a directed relationship. The nodes could be unattributed. In this case the presence of a node indicates the presence of a feature. When the node is not present, this indicates that the feature is not present. If the nodes indicate some measure of the feature, i.e. colour, angle, type etc., then the nodes will represent this, possibly multiple, information.

There are many papers published on the theoretical properties of graph matching (See Bunke [8] for a review), however, there are few real and large applications that have been deployed. This is possibly because the computation time on large datasets has been a hindrance. Although it is possible to use heuristic methods, many of these return too many incorrect matches to be of any value.

The challenge in matching directed graphs is that typically the time to compute a match increases exponentially with the number of nodes in each graph. The most basic approach to matching one graph to another is the brute force heuristic search. The aim is to return a score which represents how well one graph matches another. This is simply the count of the nodes and arcs that match between the two graphs. This score can then be used to select the graph that best matches the input graph.

Graph matching is known to be NP-complete [8], that is, it can take an exponentially long time to find the optimal match between two graphs, even though it is also possible to find such a match in less time, i.e., the worst case match time is exponential in the number of nodes and arcs, while the best case is linear time.

Although the time to compute a graph to graph match using a brute force (exhaustive) approach is exponential in the number of nodes in the graph, the time to match a query, Q, to n stored graphs is linear in n.

3 The AURA Graph Matcher

This section describes the AURA graph matcher system. The basic method was developed in Turner and Austin [19] and subsequently extended [18]. The AURA approach is a set of methods, based on neural networks, for searching large incomplete and complex datasets. In general, the AURA methods allow searching of three classes of data: graphs as outlined here; text as described in Lomas [15]; and signals (or numbers) [11]. Together, the AURA approach forms a family of methods that can be used on a wide variety of problems. More recently, we have been applying the methods to commercial problems through the team's spin off company, Cybula Ltd (www.cybula.com). Initially the methods were developed to understand image data [17]. From this an associative memory was developed based on a simple neural network (a CMM). The memory was developed to allow both rapid recall of associated items as well as practical implementation in scalable computer hardware.

The AURA graph matcher is a combination of relaxation labelling methods [12] and CMM neural network methods. The following description starts by outlining the concept of relaxation, and then describes how it is implemented in CMM based neural networks. It then describes how the methods have been implemented on parallel hardware and used in a number of practical applications. The approach has been described in Turner and Austin [20], where it is shown how a probabilistic interpretation of relaxation labelling can be implemented using a binary method with only a small loss in accuracy. Here we bring together the methods and applications for the first time in a single chapter. The mathematical treatment of the mapping onto CMMs is also new.

3.1 Relaxation by Elimination

Relaxation by elimination is a method that allows a constraint space to be searched in a reasonably effective way. The process of matching one graph to another is a straight forward task, if the graphs are identical. Unfortunately in many applications one graph will be significantly different from another. To make matters worse, the constraints on the nodes and the arcs may only be similar. Thus the algorithm that matches one graph to another must not only find out which nodes and arcs match between the two graphs, but also find the ones that are most similar. One way to approach this problem is to use the idea of matching through the relaxation of constraints. In general, this approach attempts to find the minimum set of constraints needed to be withdrawn (relaxed) to allow the two graphs to match. These constraints can be unary i.e. single constraints such as found at a node (a feature or colour for example) and binary between two nodes such as is represented by an arc (a distance between nodes for example). In RBE the process of relaxation is achieved by elimination of constraints that are unsupported. In conventional relaxation [12] the relaxation process is achieved through a normalisation process that uses a probabilistic framework to filter the information. The RBE approach has been particularly designed for rapid operation though implementation on binary neural networks.

First we present an overview of the method. Consider the following simple example. A graph to be matched, *Q*, is shown in Fig. 1 and in Fig. 2 two examples in the database of graphs that are to be matched against *Q*. We have three nodes to match, labelled 1, 2 and 3. Each node has a single node constraint, *n*, and this may represent a local feature such as a colour or angle of an edge. The arcs are undirected and also have a single constraint, *c*, which could represent the distance between nodes for example.

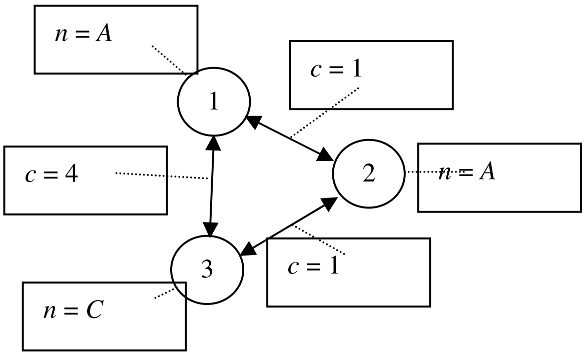


Fig. 1. A simple example of a graph, *Q*, with single attributes on the nodes (circles) and arcs (arrows) that are undirected. The nodes are numbered 1, 2 and 3. The arc constraints are given by, *c*, and the node constraints by *n*.

Fig. 2 shows a similar pair of graphs in a database: D_1, which is the same as Q and D_2, which differs significantly from Q.

The match process aims to find a mapping between Q and each of the graphs D and thus find the match that minimises the number of constraints that are violated. This represents typical data from an image analysis system, where the node information might be the type of vertex found at the node and the relations between these nodes are the distances between the edges. Note that the graphs contain some ambiguity in the node information meaning the relational information must be used to achieve a correct match.

The process of matching takes the following four stages:

1. Initialisation by matching single nodes between graphs Q and D_x.
2. Computation of the relational support for the nodes.
3. The process of elimination of weakly supported nodes.
4. The termination check.
5. Repeat from 2 until termination.

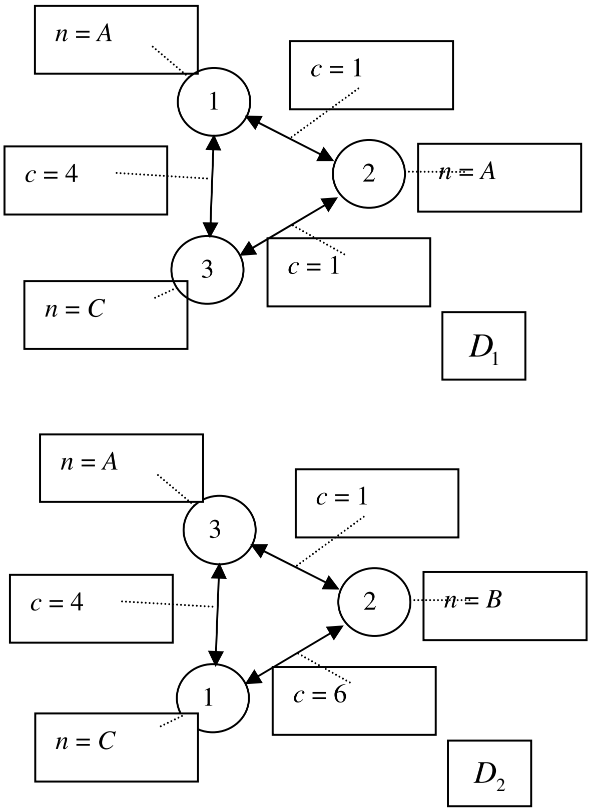


Fig. 2. Two graphs to be matched to the example in Fig.1. An identical graph, D_1, and a non identical graph D_2. The nodes are numbered 1, 2 and 3. The arc constraints are given by, c, and the node constraints by n.

3.1.1 Initialisation of Matching

The first stage takes each node in Q and attempts to match this against the graphs in the database taking into account only the node constraints. The result of this is a table of initial support shown in Table 1.

Table 1: Initial matches using only node constraints that match.

Node Number in Q					
1		2		3	
Graph	Node	Graph	Node	Graph	Node
D_1	1	D_1	2	D_1	3
D_1	2	D_1	1	D_2	2
D_2	3	D_2	3		

Table 1 shows that nodes 1 and 2 in Q match two nodes in graph D_1 and one node in graph D_2. Node 3 matches one node in both D_1 and D_2.

The method here has exactly matched each node constraint. This process could use a distance measure to identify close matches. It is important that the match process is inclusive to ensure that no plausible candidates are excluded at this point. To avoid unnecessary computation, clearly impossible matches are removed at this point.

The next stage aims to remove the ambiguity in which nodes in the query match those in the database by taking into account the arc constraints between the nodes.

3.1.2 Calculation of Relational Support

The next stage uses the arc constraint information to calculate the support for each node and remove inconsistent matches. The process is summarised in pseudocode as:

```
1. For each node, $X$, in graph $Q$
2. For each graph $D$ in the database
3. For each node $Y$ in graph $D$ that matches the node constraint on $X$
4. For each arc, $A$, from node $X$
5. Count, $C$, the arcs from node $Y$ with the same arc constraint as $A$
```

To do this, the table of initialised matches (Table 1) is consulted along with the relational information from each node in Q and D_x. The counts (support) are given for each node X as shown in Table 2.

Note that the correct match is the first row (highlighted) in the table. The ‘sup’ field is the count C, of support for each node. This is a very simple sum operation and, as will be shown later, has been designed to be implementable on binary neural networks. This stage of processing is the same as used by Waltz [22] on an early implementation of this approach.

Table 2: The relational support for each node.

Node Number in Q								
1			2			3		
Model	Node	Sup	Model	Node	Sup	Model	Node	Sup
D_1	1	2	D_1	2	2	D_1	3	2
D_1	2	1	D_1	1	1	D_2	1	1
D_2	3	2	D_2	3	1			

Note that node 1 in Q matches equally well node 1 in graph D_1 and node 3 in graph D_2. Thus there is an ambiguity. This will be dealt with by the elimination phase described next.

3.1.3 Elimination of Unsupported Nodes

As noted above, the next stage is to eliminate the weakest supported nodes from Table 2 and attempt to remove any ambiguity. In this example, nodes with support of 1 or less are removed, as shown in Table 3. The value at which nodes are removed is reasonably critical. Too low a value and ambiguities will remain in the match, too high a value and nodes will be removed too quickly, perhaps removing good matches.

As shown in Table 3, nodes 2 and 3 have one match with graph D_1. Thus they are now complete. However, node 1 still has two possible matches. The next stage removes this ambiguity.

Table 3: Result after elimination.

Node Number in Q								
1			2			3		
Model	Node	Sup	Model	Node	Sup	Model	Node	Sup
D_1	1	2	D_1	2	2	D_1	3	2
D_2	3	2						

3.1.4 The Next Iteration

To remove the ambiguity in node 1 in Table 3, the support values are removed and the constraint propagation process is used again. This is the process of elimination that the RBE method refers to. Note by removing nodes, matching constraints are also removed (i.e. the node constraint of the node being removed and the arc constraints for the arcs from that node). Table 4 shows the nodes with the support values removed.

Table 4: The information for the second iteration.

Node Number in Q					
1		2		3	
Graph	Node	Graph	Node	Graph	Node
D_1	1	D_1	1	D_1	3
D_2	2				

Relational support is calculated again, as done to calculate Table 2. The result is shown in Table 5.

Table 5: The node support after the second iteration.

Node Number in Q								
1			2			3		
Model	Node	Sup	Model	Node	Sup	Model	Node	Sup
D_1	1	2	D_1	2	2	D_1	3	2
D_2	3	0						

Because there is only one node from graph D_2 in Table 5, this node cannot get any support. Thus, it will now fall below the elimination threshold in the next step.

3.1.5 The Final Elimination Stage

The elimination stage is re-applied. Any node with a support of 1 or below will be removed. This produces just one node matching each Q node.

Table 6: The final match of each node.

Node Number in Q					
1		2		3	
Graph	Node	Graph	Node	Graph	Node
D_1	1	D_1	2	D_1	3

3.1.6 Termination of the Search

The process of elimination and calculation of support continues until no more nodes can be removed by elimination.

The following covers some of the issues when using the method in practical applications.

3.2 Selecting the Best Match

After the method terminates, identification of which molecules is the best match is done. It's possible that many nodes could exist in the result list with a mapping to a number of nodes in the same model. These represent possible symmetries in the match. For example, it is possible that the result in Table 2 was the final state of the search if a user decided to stop at that point. In this case there are two ways that Q matches D_i, one more optimal than the other. In some applications it is useful to know sub-optimal matches. Unfortunately the process of extracting the consistent mappings is a search in itself using the information in the table. However, the number of candidates is far smaller than in the original search on the database, thus this operation can be achieved relatively quickly.

3.3 Calculating the Similarity Score

It will be clear that the final result is a list of candidate matches for each node in Q against nodes in the database. In many applications it is desirable to have a score that indicates how good the match is.

There are a number of methods that can be used to calculate the match strength. We have found useful ones to be the Tanimoto [18], Bunke [9] and Simpson [16] coefficients. These similarity scores are defined based on the number of matches between the nodes in the database D_i and input Q, represented as $D_i \cap Q$, normalised by different factors depending on the method:

- The Tanimoto coefficient is defined as $\dfrac{D_i \cap Q}{QD_i - D_i \cap Q}$
- The Bunke coefficient is defined as $\dfrac{D_i \cap Q}{\max Q, D_i}$
- The Simpson coefficient is defined as $\dfrac{D_i \cap Q}{\min Q, D_i}$

All these metrics provide a value between 0 and 1 with a higher value indicating greater similarity between D_i and Q.

Other metrics are possible, and there has been considerable discussion in the literature on which of these or other metrics provide the best coefficient. However, it is difficult to capture the subtlety of a match in a single coefficient.

In addition, the final match may be subject to a *clique detection* algorithm so that small cliques are removed. This lies outside the scope of the RBE method, and its use may or not be appropriate depending on the data and the type of graph comparison the user wishes to perform. The standard clique detection algorithms are not assured to have a finite completion time, and so such algorithms need to be used with care. In practice a limit on the time taken in clique detection is imposed to ensure that the time taken is bounded.

3.4 Summary of RBE

The process in the previous section has shown how a method for graph matching can be implemented. The following sections show how the method has been implemented using neural networks. Although neural implementation is not essential for the technique to work, it illustrates a direct and effective method for implementing the algorithm that allows efficient parallel operation.

4 Interesting Properties of RBE

The RBE approach has the interesting and important property that it is 'local' in its computation, allowing effective parallel processing. To be able to parallelise a method effectively, it helps if the methods can be partitioned into separate, non-interacting segments. The approach described here has the property that each node calculation only requires information on the constraints applied by other nodes. The nodes only use local information during the threshold phase. Alternative approaches require a maximum to be calculated across all the nodes, which means they need to interact.

As pointed out already, the RBE method was developed specifically to be supported by the AURA methods, which are inherently parallel. The next section describes this approach and how RBE is implemented on the AURA methods.

5 Binary Correlation Matrix Memories

The AURA methods are based on the correlation matrix memory [17]. This is basically a single layer neural network that uses Hebbian learning [13], one of the most basic neural network learning methods. Fig. 3 illustrates a CMM as a binary matrix and a single layer network.

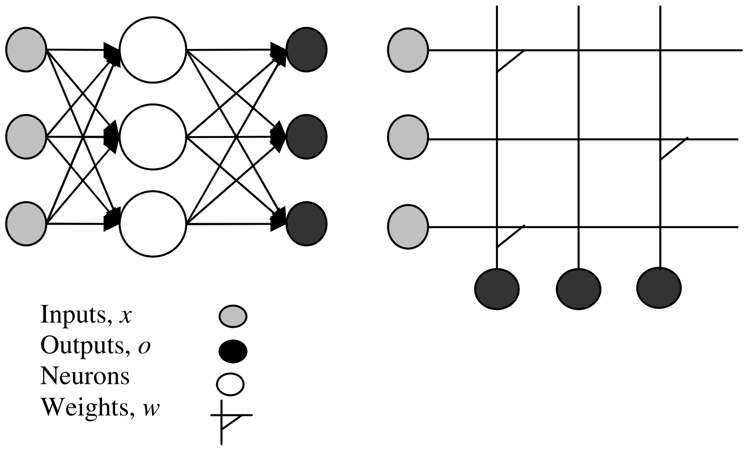


Fig. 3. Diagram of a CMM. On the left is a conventional diagram of a CMM, as a single layer network with three inputs and three neurons. The diagram on the right shows the same network as a matrix. The advantage of the right hand representation is that the weights can be shown explicitly.

A CMM can be seen as an associative memory that, given a pattern vector ***x***, recalls the associated pattern, ***o***. The memory can be trained on large numbers of patterns to associate. At some point, due to capacity limitations, the memory will fail to recover the associated pattern correctly. The analysis of this has been covered extensively elsewhere [17].

In the following, a bold capital letter, i.e. $\mathbf{M}$, represents a matrix and a lower case letter with an arrow e.g. $\vec{v}$ represents a vector. Other letters represents a scalar values.

In this section, a CMM is defined as a binary matrix $\mathbf{M}$. To train patterns into a CMM, binary vector operations are used. To train (create) the matrix you define input pattern, $\vec{x}$, output pattern, $\vec{o}$, as vectors. Here is a simple example with an input vector of three elements and an output vector of three elements:

$$\vec{o} = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$$
$$\vec{x} = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$$

In our case these patterns are binary. The outer product, $\otimes$, is taken of these two vectors to create the CMM,

$$\mathbf{M} = \vec{x} \otimes \vec{o}$$

For this example, this produces:

$$\mathbf{M} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

When more pairs of patterns are associated, the resultant matrices are OR'ed with each other,

$$\mathbf{M} = \bigcap_{\forall i} \vec{o}_i \vec{x}_i$$

where $\bigcap$ represents a logical OR, the superposition, of all values of the matrices with each other. For two matrices $\mathbf{M}^1$ and $\mathbf{M}^2$ this is given as:

For all i and j

$$\mathbf{M}_{ij} = \mathbf{M^1}_{ij} + \mathbf{M^2}_{ij}$$

where *i* is the row index into the matrix **M** and *j* is the column index and + is the logical OR operation between two matrices $\mathbf{M}^1$ and $\mathbf{M}^2$.

If we had two patterns to train:

$$\vec{o}_0 = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$$
$$\vec{x}_0 = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix}$$

and,

$$\vec{o}_1 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}$$
$$\vec{x}_1 = \begin{pmatrix} 0 & 1 & 1 \end{pmatrix}$$

The result of training will be:

$$\mathbf{M} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

To recall data (an association) from the network, the CMM uses the basic sum-and-threshold operation found in single layer neural networks:

$$O_j = \sum_{\forall i} w_i * x_i$$

where O_j is the output of the j^{th} neuron. w_i and x_i is the i^{th} weight connecting the i^{th} input to the neuron.

In matrix terms, this is defined as the product between the input vector, $\vec{x}$, and the matrix $\mathbf{M}$:

$$\vec{r} = \vec{x}\mathbf{M}$$

In typical neural networks, the product, *, is defined as the sum of the products between $\vec{x}$ and $\mathbf{M}$:

$$\vec{r}_j = \sum_{\forall i} x_i * \mathbf{M}_{j,i}$$

In binary CMMs, the product is replaced by the sum of the logical AND's (given as '$\wedge$' here) between the terms of $\vec{x}$ and **M**:

$$\vec{r}_j = \sum_{\forall i} \vec{x}_i \wedge \mathbf{M}_{j,i}$$

To show this operating, consider the matrix created above. We present the original pattern to the matrix:

$$\begin{pmatrix} 0 & 2 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

The result, **r**, must be thresholded to recover the original pattern, *o*. In this example, we select the highest element from the result, to produce:

$$\begin{pmatrix} 0 & 1 & 0 \end{pmatrix}$$

This is the pattern previously trained into the CMM. It is also useful to have a simple iconic representation of a matrix as shown in Fig. 4.

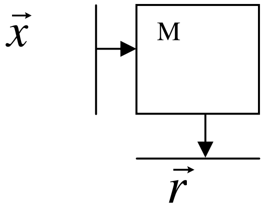


Fig. 4. The image shows an icon of a CMM recall resulting in the unthresholded value, *r*.

In the AURA methods, it is often necessary to rewrite a matrix $\mathbf{M}$ as a vector. This is defined by the operator $\mathbf{L}()$ as in $\vec{v} = \mathbf{L}(\mathbf{M})$. To do this, the columns of the matrix are extracted in order, transposed and concatenated with the others to produce a single row vector. The inverse, $\mathbf{L}'()$, can also be applied to a vector to recover the matrix, $\mathbf{M} = \mathbf{L}'(\vec{v})$.

The weights in the CMMs we use are binary 0 and 1 weighted, as opposed to continuous weights used in most other CMM implementations. This allows us to implement CMMs very effectively on computer hardware at high speed, as will be discussed later.

One of the advantages of using a binary CMM is its speed. In many neural networks learning is a slow, iterative process used to obtain the best classification of the input data. Here, the aim is to store examples of the input data in the weights, so a simple Hebbian learning rule as shown above is used. Much of our work has been inspired by using methods that are biologically plausible. This does not mean that they are used by the nervous system, but could plausibly be used. The idea is that we then use methods that biology might use to solve problems that are hard to solve using conventional methods. It will be evident that the CMM associates an input pattern and an output pattern with each other, so that input of one pattern can elicit the recall of the other. Because of this CMMs are often called associative memories.

CMMs have been implemented in the AURA software library developed at the University of York [5]. To make the implementation particularly efficient the library uses sparse vector methods to efficiently store very large CMMs. Evaluation of these large CMMs is particularly fast, especially where the input pattern is sparse. The AURA library has been mapped onto dedicated hardware using FPGAs for more effective parallelisation of the methods.

6 Binary Correlation Matrix Memories

One of the benefits of RBE is that it can use the CMMs to store the constraints in the graph and also for the mapping of each node in the graph *Q* to the database nodes. Our approach has used CMMs to implement a particularly fast version of the graph matcher system. The following describes how the method is mapped onto CMMs, and is described using vector and matrix operations.

The following labels are defined and used in the following:

The input graph	Q
The database	D
An instance of a graph in the database	g
A node in the database graph	i
A node in the query graph	r
A node constraint	n
The binary vector of constraints at a node	$\vec{n}$

An arc in the database graph	j
An arc in the query graph	s
An arc constraint	c
The binary vector of constraints at an arc	$\vec{c}$

The matrix of node and arc constraints for all nodes, *i*, and all arcs, *j*, in the database graph *g* in the input *Q*	$C^{Q}_{r,s}$
The vector of node and arc constraints for all nodes, *r* and all arcs, *s*, in the input *Q*	$\vec{v}^{Q}_{r,s}$
The matrix of node and arc constraints for all nodes, *i*, and all arcs, *j*, in the database graph *g* in the database *D*	$C^{D}_{i,j,g}$
The node and arc constraints for all arcs in a node in the database *D*	$\vec{v}^{D}_{i,j,g}$
A vector containing all the arc and node constraints for a database graph *g*.	$v^{D}_{i,g}$

The elimination threshold	T
The support of each mapping of a node in a graph in D to a node in Q	$\vec{R}_{\mathrm{r}}$

The following sets out the steps in the process:

Step1: Node and arc constraints
As described already, each node in the graph, *Q*, has a local constraint, called a node constraint, *n*. Also each arc from that node has a constraint called an arc constraint, *c*.

Step 2: Binary representation of node and arc constraints in *Q*
As binary methods are used, the value of a node constraint is represented as one bit set in a vector, $\vec{n}_r$, where *r* is the node in the query graph *Q***.** This has length $|\vec{n}|$ equal to the number of values that a constraint at a node may take.

The same applies to the arc constraint information which is represented as a vector $\vec{c}_s$ for arc *s* at node *r*. This vector has a length $|\vec{c}|$ equal to the number of possible values for the constraints in the system. For example, if an arc constraint is a distance between the nodes and the value of the distance varies from 1 to 10, then the constraint vector $\vec{c}$ would have 10 elements. A 1 in element 4 would represent the distance 4 between the two nodes.

Step 3: Matrix of node and arc constraints
The node constraint vector, $\vec{n}_r$, and the arc constraint vector, $\vec{c}_s$, are combined into a correlation matrix, $C^Q_{r,s}$, for the input graph *Q*:

$$C^Q_{r,s} = \vec{n}^Q{}_r \otimes \vec{c}^Q{}_s$$

There is one for each node and arc in the graph where $\otimes$ is the outer product given in Section 5.

Step 4: Conversion of the constraint matrix to a vector
For reasons that will be shown below, all $C^Q_{r,s}$ are re-written as vectors, $\vec{v}^Q_{r,s}$, using:

$$\vec{v}^Q_{r,s} = \mathbf{L}(C^Q_{r,s})$$

where **L**( ) is defined in Section 5. The constraint vector, $\vec{v}^{Q}_{r,s}$, represents the constraints for node *r* and arc *s* in the input *Q* each vectors has the length $|\vec{n}| \times |\vec{c}|$.

Step 5: Encoding the database information
Now we have the constraint information for the input graph, *Q*, the information in the database graphs is also encoded in the same way by combining the database information into a CMM.

Step 6: The database nodes and arcs
All graphs, *g*, in the database, *D*, have nodes numbered *i*, and arcs numbered *j*. For each graph we create a constraint vector, as in the query graph.

Step 7: Binary representation of node and arc constraints in *D*
Again, each node and arc in a given database graph has its constraints encoded into a vector. The arc constraints are given as $\vec{c}^{D}_{j,g}$ for the arc *j* and graph *g* at any node. The node constraints are given as $\vec{n}^{D}_{i,g}$ for node *i* in graph *g*.

Step 8: The matrix for the node and arc constraints of one database graph
As for the query graph, the arc and node constraints are combined into a CMM matrix $C^{D}_{i,j,g}$ that holds the constraint information, and is formed as:

$$C^{D}_{i,j,g} = \vec{n}^{D}_{i,g} \otimes \vec{c}^{D}_{j,g}$$

Thus $C^{D}_{i,j,g}$ holds all constraints for node *i* and arc *j* from that node in a given graph *g* in D.

Step 9: Conversion of the constraint matrix to a vector
All $C^{D}_{i,j,g}$ are again re-written as vectors, $\vec{v}^{D}_{i,j,g}$, using:

$$\vec{v}^{D}_{i,j,g} = \mathbf{L}(C^{D}_{i,j,g})$$

The constraint vectors, $\vec{v}^{D}_{i,j,g}$, represent the constraints for node *i* and arc *j* in the graph *g* in *D* and each has the length $|\vec{n}| \times |\vec{c}|$.

Step 10: Combining all the database constraint information
To create a more efficient system, we use an optimisation. We combine all arc constraint vectors across all nodes in a graph into one vector, by superimposing the vectors:

For all i and g;

$$\vec{v}^{D}{}_{i,g} = \bigcup_{\forall j} \vec{v}^{D}_{i,j,g}$$

where the operation $\bigcup$ superimposes (logically ORs) vectors on top of each other. There is now one vector, $\vec{v}^{D}_{i,g}$, that holds all of the information for each graph in the database. Doing this speeds up the performance of the method, with no loss in accuracy. Note that $\vec{v}^{D}_{i,g}$ is a two dimensional matrix structure made up of $g_{\max} \times i_{\max}$ column vectors (number of graphs × number of nodes in each graph) each of $\left|\vec{v}^{D}\right|$ size, i.e. a CMM.

We now have a single data structure that holds all the constraints in the database $\vec{v}^{D}_{i,g}$ and one data structure that holds the input query $\vec{v}^{Q}_{r,s}$. By using these two data structures the similarity between the graphs may be computed as shown next.

Step 11: Holding the support for each node

The vectors representing the input query and the database can now be used to build the support list, as shown in Table 1. A set of vectors, $\vec{R}_{\mathbf{r}}$, is needed to hold the support scores for each node in the query graph. $\vec{R}_{\mathbf{r}}$ is a set of $r_{\max}$ vectors, one for each node in the query graph. Each vector $\vec{R}$ gives the support for each node onto a particular node and graph in the database, D. Each $\vec{R}$ is $r_{\max} \times g_{\max}$ in size.

Step 12: Calculating the support

The process of support calculation follows that in Section 3.1. First there is an initialisation stage, then there is a process of support calculation, followed by a threshold operation and this process iterates until the process terminates.

Step 13: Initialisation of the support

The initialisation checks to see if the node constraints in the query match with any nodes in the database.

If the node i and constraint n at that node matches any node constraint n in the database graph, then the relevant element in $\vec{R}_t$ is set to 1.

Step 14: Using the arc constraints to update the support

A computation of the support values is achieved by performing a match between the vectors $\vec{v}^{D}_{i,g}$ and $\vec{v}^{Q}_{r,s}$ and placing the result in $\vec{R}^{t}_{r}$, which is the state of $\vec{R}_t$ at time t. This is done as follows:

```
For each node $r$ in the query graph $Q$
        For each arc $s$ from node $r$
For each graph $g$ in the database $D$
    For each node $i$ in $g$
        $R_r^t = \left(\vec{v}_{i,g}^D \cdot \vec{v}_{r,s}^Q\right)$ and $R_r^{t-1}$
```

where $\vec{R}_r^t$ is the result at time t (or after initialisation given in Section 13) and $\vec{R}_r^{t-1}$ is the result at the previous iteration after thresholding (below). The operation $\left(\vec{v}_{i,g}^D \cdot \vec{v}_{r,s}^Q\right)$ is the dot product between the two vectors, each returning a single value into the corresponding vector location in $\vec{R}_r^t$. The *and* operation only allows this value to be entered where the corresponding bit in $\vec{R}_r^{t-1}$ is at 1.

Step 15: Thresholding the support

The result $\vec{R}_r^t$ is then thresholded to remove (eliminate) any value with poor support scores as described in Section 3.1.5. An elimination threshold T is set. Any values above this are set to 1, any others are set to zero. This becomes $\vec{R}_r^{t-1}$ in the next iteration.

Step 16: Termination of the process

If the number of bits set in $\vec{R}_r^{t-1}$ fails to change in two iterations, then the process is terminated. Otherwise the process returns to step 14.

7 Calculation of the Quality of Match

After k iterations the RBE process completes and the remaining nodes with support will be given in the vector $\vec{R}_r^t$. This information can be extracted for all graphs and a list produced of the graphs that match the input with the minimum number of relaxed constraints. Note that no information on the quality of the match is given. As pointed out already, this has to be calculated after the RBE process using a separate distance measure that assigns a score to each graph. In practice this operation may not be a graph based comparison, as it can use a highly compute intensive process due to the small list of matching items that are left. From this, it is clear that the RBE process can be used as a pre-filter to an existing exhaustive match process. In this way, the method can be added to existing methods to give a considerable shortened execution time from the system as a whole.

Since all plausible matches are retained, the sensitivity is always 1. Specificity is affected by a number of factors. For example, a graph which is less than fully connected at each node may result in false positives due to insufficient propagation of removed implausible matches throughout the graph, with the benefit being lower overall execution time (see Fig. 5). Using a threshold for support of less than the full connectivity at each node may also result in false positives, but will allow the system to undertake inexact matches. Given the difficulty of determining an objective set of true positives and true negatives for a system of graphs other than trivial cases or for very specific examples, no figures on the specificity of the method are presented in this chapter.

8 The Performance of the Method

The time complexity of RBE is indeterminate as it depends on the exact constraints present in the graphs and the graph structure. In general terms the processing speed is $O(n^2)$ where n is the number of nodes in the graph. For a particular set of graphs, the number of operations will be proportional to:

$$\left[\sum_{All\, g} N_g \times (N_g - 1)\right] iter$$

where N_g is the number of nodes in a graph g in the database and *iter* is the number of iterations needed to complete a search.

Because the structure of the graph affects the performance of the method, performance can only be calculated empirically, as has been done in Fig. 5. To calculate these results we generated a number of random graphs and stored them in the database. Each graph had the same connectivity (number of arcs from each node).

The random graphs were stored in the database. Each graph in the database has 100 nodes. One thousand randomly generated graphs were stored in the database. Thus the set of stored graphs is a small subset of the set of possible graphs, and assumed to be evenly distributed within the total graph space. Sparse distribution of graphs in the set of all possible graphs is common in many real matching scenarios although real data often has a non-uniform distribution with the graph space as measured by similarity scores. Each query was taken from the set of stored graphs, either as a complete graph, or as a sub graph of a stored graph.

For each run we generated one query graph, varying the number of nodes in the query on each run (as shown on the x-axis). In each case we ran the program to completion. The threshold, T, was varied for each plot. The x-axis is the number of nodes in the query and database, N_g. The y-axis shows the relative time (%) to complete a search.

We looked at two cases of connectivity and its effect on the result. The top line (dotted) is where the connectivity in the graph is $N_g - 1$, i.e. a fully connected graphs. The bottom line (solid) shows where the connectivity is maintained at 9 for all examples.

The experimental set-up used was version 3.1.0.1 of the 'Graphmatcher' software library with version 2.8.0 of the AURA library, compiled with gcc 3.4.6 with –O2 run on a system with 2 quad core 2Ghz Xeon processors per node each with 16GB RAM running 64 bit Scientific Linux 4.5, these nodes forming Cortex II.

In Fig. 5 it can be seen that query time is linear on query size for fixed connectivity of 9. As the number of nodes rise, the connectivity rises, and as this happens the performance rises exponentially.

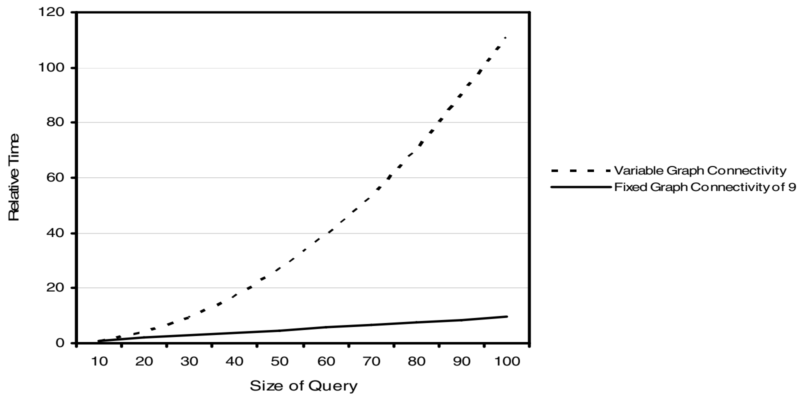


Fig. 5. Showing the relative time to compute a match against average number of nodes in the query. Two plots are shown on this graph: the solid line gives results for a fixed connectivity, i.e. 9 in all instances, irrespective of the size of *Q*. The dotted line gives results for connectivity equal to one less than the query size.

9 Applications

The AURA based RBE method has been used on a number of applications. The first of these was for molecular matching [14] followed by trademark logo retrieval [1]. The reason for applying the method to these problems is that they are computationally difficult. Details of the methods can be found in each paper cited above. The following describes the general approach to the use of the method on image problems and describes the molecular matcher application in more detail.

The main task in applying the method to a problem is to decide on the local and relational constraints to use. In image recognition problems, the local constraints (node constraints) are usually some local property of the image calculated using feature detectors. These can be edge angle, curvature of the surface, texture properties or something similar. In the trademark system, complex image primitives are used that relate to gestalt properties. In faces, local curvature of the face can be used. In the molecular matcher, the curvature of a surface drawn over the molecule can be used, or the Coulomb charge at that node.

The relational (arc) constraints capture some information that relates one node to another. In many applications it is a simple measure of the distance between the nodes. In the trademark system, the relations are more complex. For example, if two nodes represent small edge segments of a line, and those lines are co-linear, then the relation is ‘lines are collinear’.

There are a number of techniques which may be used to select the characteristics for the nodes and constraints such as Principal Components Analysis (PCA) or the use of genetic algorithms. This information must then be mapped onto the system as binary vectors.

10 Mapping Data onto Binary Vector

When information is represented to the system it is in the form of binary vectors. This allows the rapid computation. Each bit in a vector represents some value. For example, in image recognition a node can have a set of node constraints based on edge angle under the node. This could be any angle between 0^o and 179^o. This information would be represented in a binary constraint vector $\vec{N}$,

$$\vec{N} = \left(\mathrm{b}_0, \mathrm{b}_1 ..., \mathrm{b}_{\mathrm{Nmax}}\right)$$

where each b_i is a bit with a value 1 or 0. In this case $\left|\vec{N}\right|$ is 179, a bit for each angle of the edge. A one in the vector location represents the presence of that value, a zero represents 'unknown'. This is different from other binary representations where a zero would mean 0 (i.e. the value 0). In our case, the vector can contain multiple bits sets suggesting that the edge (or what ever the vector represents) can possibly take one of the set of values. In effect, the vector contains a 'vote' for the setting of that variable to that value. A 1 is one 'vote' and 0 is no vote. If other nodes suggest that the element should be 0 rather than 1, then the system allows this. This is a powerful concept used throughout AURA based CMM systems.

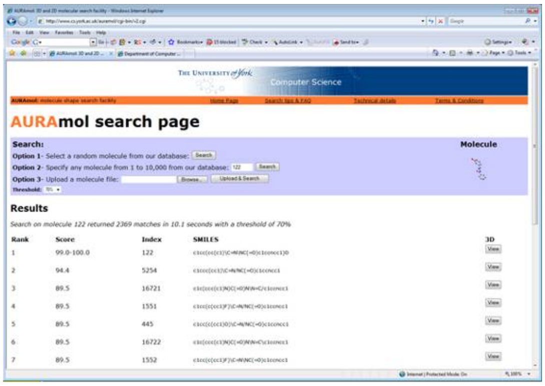
For some applications these vectors can become quite large. However, as they are mostly all 0's, they can be represented efficiently, as discussed below.

The quantisation used to encode these values which, by its nature, introduces some level of approximation. In the above example the values for 10.0° to 10.999…° would be encoded as the same binary vector. In some cases this may be sufficient however it means that the system will consider 9.999…° and 10.0° as different values. Coding systems should be chosen carefully to ensure that there is no unintended negative effect on the specificity or sensitivity of the system. The appropriate coding scheme will depend on the relationships and the desired behaviour in terms of what values should be seen as similar to other values.

11 The Cybula Molecular Matcher

The RBE method has been used in many applications as pointed out already. To give an idea of how it is applied, we consider the molecular matcher system mentioned above. Here we consider the general approach to the problem to allow others use the method.

The molecular matching system was designed to support computational chemists select appropriate drug like molecules for trials. When a drug is developed, other companies often would like to copy the drug using a different molecule. To do this, one approach is to find and trial similar molecules to the one already being produced. There are many approaches to this. In our work we investigated the use of the three dimensional shape to identify similarly shaped molecules that could be trialled within GSK Ltd. and developed the RBE method during this work. The system (see http://www.cybula.com) now runs continually on hardware (described below) at The University of York and allows users to enter drug-like molecules and search for similar examples against a database of over 100,000 taken from the US National Cancer Institute open database of molecules with actual and potential anti-cancer properties. The system has been running for over two years. A screen shot of the system is shown in Fig. 6.



AURAmol search page

Search:

Option 1- Select a random molecule from our database: Search

Option 2- Specify any molecule from 1 to 10,000 from our database: 122 Search

Option 3- Upload a molecule file: Browse... Upload & Search

Threshold: [illegible]

Molecule

Results

Search on molecule 122 returned 2369 matches in 10.1 seconds with a threshold of 70%

Rank	Score	Index	SMILES	3D
1	99.0-100.0	122	c1cc(cc(c1)\C=N\NC(=O)c1ccncc1)O	View
2	94.4	5254	c1ccc(cc1)\C=N\NC(=O)c1ccncc1	View
3	89.5	16721	c1c(ccc(c1)N)C(=O)N\N=C/c1ccncc1	View
4	89.5	1551	c1cc(c(cc1)F)\C=N\NC(=O)c1ccncc1	View
5	89.5	445	c1cc(c(cc1)O)\C=N\NC(=O)c1ccncc1	View
6	89.5	16722	c1c(ccc(c1)N)C(=O)N\N=C\c1ccncc1	View
7	89.5	1552	c1cc(c(cc1)F)\C=N\NC(=O)c1ccncc1	View

Fig. 6. The molecular matcher interface.

The system is aimed at small molecules (up to 100 atoms). The first stage is to select a representation for the input data. In our implementation we use the Van der Waals surface around molecules. This is then sampled at regular points over this surface. Each point forms a point of the graph. We use the distance between the points as the pair wise arc constraint. There is no node constraint.

This is shown in Fig. 7 in a 2D example. The graph is then presented to the RBE method. After a number of iterations, the stopping criterion is met, with a number of graphs representing molecules remaining. The task is then to identify which molecules these relate to, which is a simple process of finding which molecules are still supported.

An important aspect of the RBE method is that the system does not directly show how well one graph matches another once the search is complete. The final ranking to show how closely two graphs match is based on the maximum common subgroup between the two graphs, and the method for ranking in AURAMol is by Tanimoto Coefficient [18].

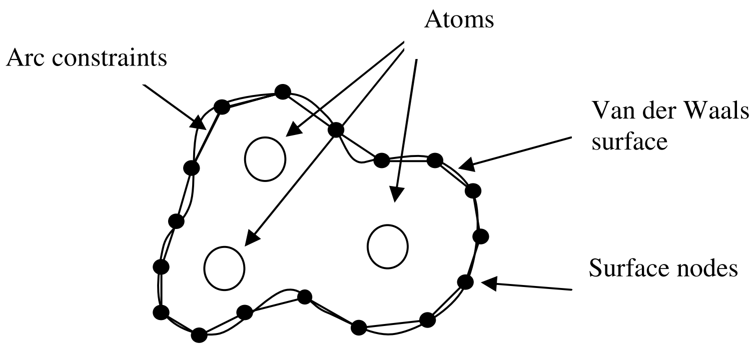


Fig. 7. A simple example of the molecular representation used showing a 2D slice through a 3D shape. The nodes and the arcs on the surface are used to represent the molecule to the system.

12 Implementation on Distributed and Parallel Hardware

The implementation of CMMs on both dedicated and general purpose hardware has always been an aim of our work. The initial work building CMMs in hardware memory systems came from the early work on associative neural networks [2]. In turn, that work developed out of an area called RAM based neural network or weightless networks [4]. These systems used RAM memories to store the weights in a neural network. With the development of the methods into binary CMMs, the work looked at implementation using both dedicated and FPGA based hardware [3] as well as VLSI [7]. Specialised hardware was investigated because a conventional processor does not implement a binary vector add (see later). After this we settled on the implementation using reprogrammable FPGAs and developed the PRESENCE card [5], and an implementation of CMMs in FPGA hardware. Fig. 8 shows an image of the second version of this card, PRESENCE II (PII). Currently we are designing PRESENCE III.

The aim of the card development was to allow the CMMs to be implemented effectively. As well as the use of an FPGA for parallel vector operations, the important features needed from the card were high performance access to memory, a large amount of memory (4GB on a card) and direct access to the card from the host PC.

In addition PII contained a Digital Signal Processor to allow more efficient use of the card for complex operations by providing a system for pre-processing tasks directly on the card thus reducing host to card transfers over the PCI bus. The ability to reduce contention on the PCI bus allows greater scalability of the number of cards within a single system.

The PII card was then used in a Sun server which houses up to 9 cards as shown in Fig. 9.

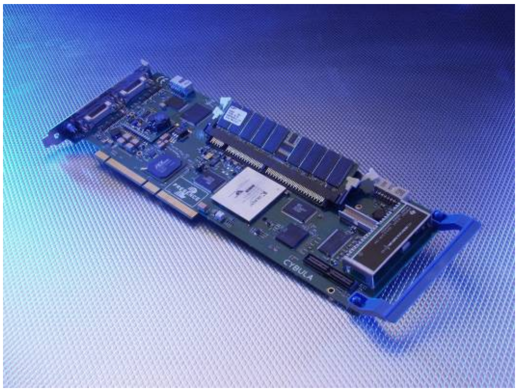


Fig. 8. The PRESENCE II Card used to run RBE.

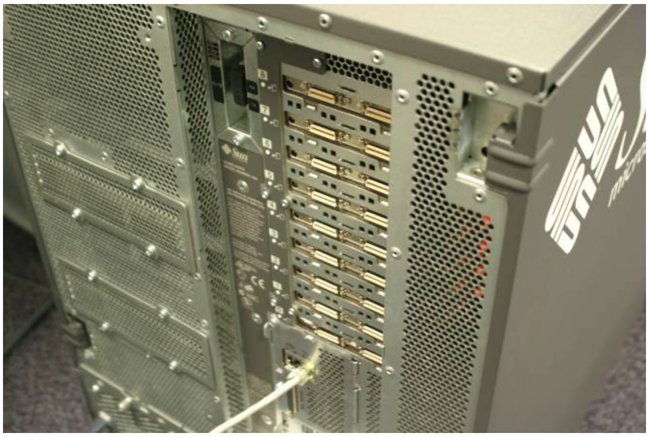


Fig. 9. The Cortex$_{tm}$ machine used to run many PRESENCE cards in parallel.

This allowed up to 20GB of binary CMMs in one system, called Cortex$_{tm}$. The latest Cortex machine is Cortex III which houses 8 Presence II cards in four high performance Xeon based servers. A version of Cortex running the RBE method on a molecular problem has been running for 2 years (as of September 2008) open to all users on the Internet.

Our work then progressed to develop a distributed version of the system that runs over multiple machines based on grid software. This allowed us to experiment with multiple databases in different locations [11].

The software (AURA) that supports these systems allows a CMM to be implemented per card, and multiple CMMs supported by many applications. We developed a software system called the AURA software library which allows the CMMs to be distributed on the hardware. In addition, we developed a multi-pattern matching system called the Pattern Match Controller (PMC) that allows multiple machines to be used to undertake pattern matching, which includes CMMs [6].

The result of this work is a fully scalable platform for associative memory based systems based on CMMs. The next section describes the important hardware mapping uses that we addressed in implementing RBE on Cortex.

12.1 Implementing CMMs in Hardware

Because CMMs are binary arrays they are particularly simple to implement in hardware. The main issues for implementation are the efficient use of memory (because the CMM data structures can be large) and the evaluation of the main matrix recall operation. The AURA library that supports the implementation of CMMs in our work includes CMMs represented in a reasonably efficient packed binary format. In the software each set of bits in a matrix is packed into each word in memory. A more efficient format is also used that records the occurrence of a binary bit by storing its position in the matrix. Although this value will be larger than storing a binary bit, when the matrix is sparse this is more efficient. This latter form is called Compact Bit Vector (CBV) representation and is basically a sparse vector method. CBV methods are used when the vectors in the system are particularly sparse otherwise the packed method is used. The CBV approach uses less memory, but can be slower and/or more expensive to implement in hardware.

To be able to undertake recall quickly on CMMs we use a row based evaluation method to evaluate the function $\vec{r} = \vec{x}\mathbf{M}$ used earlier. The input pattern, $\vec{x}$, is typically in many applications quite sparse, i.e. has very few bits set to one. This means that evaluating the equation is best done by taking each bit in $\vec{x}$ and, in effect, accessing the row in **M** that it points to. This row in the CMM is then summed into a vector $\vec{o}$, of real valued elements. As the CMM is binary, this is a fast operation as only binary elements are added. This process is illustrated in Fig. 10.

In this example, the input vector, $\vec{x}$, is applied to the CMM which contains 6×4 elements. Each bit in the input vector is accessed and if a bit is set to one, the row it points to is added into the output vector, $\vec{o}$. This is shown for the first row in the matrix.

The PRESENCE hardware implements the complete CMM on the card. The host just sends binary vectors to the card and receives vectors back. The system uses arrays of counters to implement the output vector, $\vec{o}$. Ancillary functions for converting the output vectors to binary via thresholding methods are also included.

To implement RBE, our approach uses the CMM structures on PII as explained above along with management functions on the host machine that prepares the vectors to be supplied to the CMMs. Setting up the CMMs is done on the host machine. The cards then run step 14 in the section on the complete RBE process, which is the most compute-intensive stage.

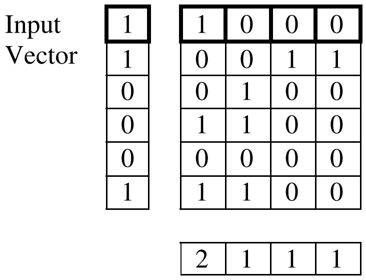


Fig. 10. The evaluation of the CMM based on row indexing.

13 Conclusions

Using the methods given here it is possible to build high performance search systems for complex relational information encoded in graphs. Although the match process is a heuristic, it has been shown to produce reasonably good results in many of the applications we have tried. Our current work continues to improve the implementation. In particular, we have developed a new FPGA implementation and we continue to develop new applications of the method. The software is available to allow users to experiment with the graph matcher [10], which is free to use in non-commercial applications.

Acknowledgments Many people have helped in the work reported here. Aaron Turner has worked on the optimisation and extension of the methods, Mike Weeks, John Kennedy and Mike Freeman on the FPGA and DSP implementation, John Kennedy, Anthony Moulds and Zygmunt Ulanowski on the PRESENCE card. Mick Turner and Stephan Klinger contributed to the basic RBE methods and their application to molecular matching. Simon Hickinbotham implanted the client-server system and interface for AURAMol. Simon Davidson helped check the text. The work was funded in part by UK EPSRC research council, GSK, DTI and Cybula Ltd.

References

[1] Alwis, S., Austin, S.: A novel architecture for trade mark image retrieval systems. In: Mira, J. (ed.) IWANN 1999. LNCS, vol. 1607, pp. 361–372. Springer, Heidelberg (1999)

[2] Austin, J., Stonham, T.J.: Distributed associative memory for use in scene analysis. Image Vision Computing 5(4), 251–260 (1987)

[3] Austin, J., Kennedy, J., Lees, K.: A neural architecture for fast rule matching. In: Proceedings of the Artificial Neural Networks and Expert Systems Conference, Dunedin, New Zealand, pp. 255–260 (1995)

[4] Austin, J.: RAM-based neural networks. World Scientific, River Edge (1998)

[5] ACA group, AURA web pages (2008), http://www.cs.york.ac.uk/arch/neural-networks/technologies/aura (accessed January 1, 2009)

[6] Austin, J., Davis, R., Fletcher, M., Jackson, T., Jessop, M., Liang, B., Pasley, A.: DAME: Searching large data sets within a grid-enabled engineering application. Proceedings of the IEEE - Special Issue on Grid Computing 93(3), 496–509 (2005)

[7] Bermak, A., Austin, J.: VLSI implementation of a binary neural network - two case studies. In: Proceedings of the 7th International Conference on Microelectronics (Microneuro), p. 374. IEEE Computer Society Press, Los Alamitos (1999)
[8] Bunke, H.: Graph matching: Theoretical foundations, algorithms, and applications. In: Proceedings of the International Conference on Vision Interface, pp. 82–88 (2000)
[9] Bunke, H.: On a relation between graph edit distance and maximum common subgraph. Pattern Recognition Letters 18(8), 689–694 (1997)
[10] Cybula, Molecular matcher web site (2008), http://www.cs.york.ac.uk/auramol/ (accessed January 1, 2009)
[11] Fletcher, M., Jackson, T., Jessop, M., Liang, B., Austin, J.: The signal data explorer: A high performance grid based signal search tool for use in distributed diagnostic applications. In: Proceedings of the 6th IEEE International Symposium on Cluster Computing and the Grid, pp. 217–224. IEEE Computer Society, Los Alamitos (2006)
[12] Hancock, E., Kittler, J.: Edge-labeling using dictionary-based relaxation. IEEE Transactions on Pattern Analysis and Machine Intelligence 12(2), 165–181 (1990)
[13] Hebb, D.O.: The organization of behavior. Wiley, New York (1949)
[14] Klinger, S.: Chemical similarity searching with neural graph matching methods. PhD The University of York, UK (2006)
[15] Lomas, D.: Improving automated postal address recognition using neural networks, PhD Thesis. The University of York, UK (2002)
[16] Simpson, G.G.: Mammals and the nature of continents. American Journal of Science 241, 1–31 (1943)
[17] Turner, M., Austin, J.: A neural relaxation technique for chemical graph matching. In: Niranjan, M. (ed.) Proceedings of the 5th International Conference on Artificial Neural Networks, pp. 7–9 (1997)
[18] Tanimoto, T.T.: IBM Internal Report (November 1957)
[19] Turner, M., Austin, J.: Matching performance of binary correlation matrix memories. Neural Networks 10(9), 1637–1648 (1997)
[20] Turner, M., Austin, J.: Graph matching by neural relaxation. Neural Computing and Applications 7(3), 238–248 (1998)
[21] Ullmann, J.R.: An algorithm for subgraph isomorphism. Journal of the ACM 23(1), 31–42 (1976)
[22] Waltz, D.: Understanding line drawings of scenes with shadows. In: Winston, P.H. (ed.) The Psychology of Computer Vision. McGraw-Hill, New York (1975)

Extremal Optimisation for Assignment Type Problems

Marcus Randall[1], Tim Hendtlass[2], and Andrew Lewis[3]

Abstract Extremal optimisation is an emerging nature inspired meta-heuristic search technique that allows a poorly performing solution component to be removed at each iteration of the algorithm and replaced by a random value. This creates opportunity for assignment type problems as it enables a component to be moved to a more appropriate group. This may then drive the system towards an optimal solution. In this chapter, the general capabilities of extremal optimisation, in terms of assignment type problems, are explored. In particular, we provide an analysis of the moves selected by extremal optimisation and show that it does not suffer from premature convergence. Following this we develop a model of extremal optimisation that includes techniques to a) process constraints by allowing the search to move between feasible and infeasible space, b) provide a generic partial feasibility restoration heuristic to drive the solution towards feasible space, and c) develop a population model of the meta-heuristic that adaptively removes and introduces new members in accordance with the principles of self-organised criticality. A range of computational experiments on prototypical assignment

School of Information Technology
Bond University
Queensland
Australia
mrandall@bond.edu.au
Faculty of Information and Communication Technology
Swinburne University of Technology
Victoria
Australia
thendtlass@swin.edu.au
Institute for Integrated and Intelligent Systems
Griffith University
Queensland
Australia
a.lewis@griffith.edu.au

problems, namely generalised assignment, bin packing, and capacitated hub location, indicate that extremal optimisation can form the foundation for a powerful and competitive meta-heuristic for this class of problems.

1 Introduction

Extremal Optimisation (EO) [6, 8] is a meta-heuristic search technique that has its origins in the science of self-organised criticality (SOC). SOC has been used to describe behaviour in systems as diverse as geophysics, economics and biological evolution. It is only recently that these concepts have been applied to solving optimisation problems [6].

This chapter investigates and examines the use of EO for a class of problems known as the assignment type problems (ATPs). We use these problems as our benchmark problems as they often have difficult constraints. Initially we look in detail at how EO selects and performs moves in search space on the bin packing problem. We then further develop EO to show that with suitable support heuristics it is capable of producing solutions comparable to those of more established meta-heuristics.

The remainder of this chapter is organised as follows. Section 2 briefly explains the idea of SOC while Section 3 gives an overview of the general tenets of EO. Section 4 describes assignment type problems. Given that there has been little work done on the analysis of EO's search behaviour in the literature, a detailed case study examination of EO transitions on one of the test problems, bin packing, is undertaken in Section 5. Section 6 shows how EO can be used more generally on ATPs. A number of topics are addressed that EO potentially needs to help it become competitive with more established meta-heuristics. These are, specifically, transition operators, partial feasibility restoration, and population models. Computational experiments across a range of problem types and instances are discussed in Section 7. Finally, the conclusions and future research directions are given in Section 8.

2 Self-organised Criticality

Self-organised critical behaviour can be observed in systems in which, over long periods of time, seemingly only small changes take place. However, these small changes gradually build to a critical state that can eventually trigger large events in a domino fashion. The most commonly related example of self-organised criticality is the sand pile model [2]. Adding a grain of sand at a time to the pile builds it up slowly to a point where the addition of another grain will push its downhill neighbour grains, which in turn pushes other grains – in effect triggering a sand slide or *avalanche*. The pattern of this behaviour is present in a diverse range of natural and artificial systems, including the flow of rivers, and the formation of mountain landscapes and coastlines and economic fluctuations in stock markets [4].

There are two important characteristics of self-organised critical systems. The first is that they do not require fine tuning of parameters to exhibit complex, self-organised behaviour. The second is that the avalanche magnitude divided by the log of the time between avalanches of this size is roughly constant and follows a pattern whereby larger events are exponentially less likely than smaller events. A good illustration of this is the Guttenberg-Richter Law for the size of earthquakes [2], an earthquake of size 4 on the (logarithmic) Richter scale will occur ten times more frequently than an earthquake of size 5.

Our interests in this chapter are the applications of SOC within biological evolutionary systems that can be ultimately simulated in order to provide models for combinatorial optimisation. Bak [2] has noted that the concept of survival of the fittest, or conversely, the selection and elimination of the few most poorly adapted species in a particular environment, displays the characteristics of SOC. In other words, there is no central organising agent nor finely tuned system in Nature that manages the survival or extinction of the species. The latter is referred to as *adverse selection* [7] and is a property that can be modelled in terms of SOC. The Bak-Sneppen model [3] represents a simplified system of interacting species. The underlying purpose of the model is to demonstrate emergent behaviour in evolutionary selection processes. One key simplification is that species are represented by a single fitness value, which is not derived from a genetic representation of the species. All species are assigned a fitness value in the range $[0, 1]$, where 0 indicates the least fit. At each step of the evolutionary process, the species with the lowest fitness value has this fitness changed to a random value. As in this simple model there is no structure to a species, this is the biological equivalent to allowing the original species to become extinct and a new species to take its place.

The model also recognises that species do not exist in isolation. For example, if a species becomes extinct, the species directly above and below it in the food chain will be affected. Therefore the neighbours (as defined by a set of lattice or ring sites) of a species that has changed will have their fitness values updated to random values as well. After many steps of this process, all species will have had their fitnesses increased and the probability increases that the species that have their fitnesses replaced have them replaced with lower values. Again it is probable that the new worst fitness is one of those just introduced and that the replacement process will result in a sharp reduction in a neighbour's fitness. Within a few steps like this the average fitness of the species collapses, and then the process of the gradual increase of fitness values begins again. This sequence of events is referred to as a *punctuated equilibrium* [3]. That is, apparent equilibrium in the system is punctuated by avalanches. Such events allow the species to potentially sample all of configuration space.

This process can be shown with a remarkably simple computer simulation. Algorithm 1 shows Bak-Sneppen's [4] ring model[1] in which each species affects its two neighbour species. Neighbours are defined in terms of position on the ring. The values of the worst species on the ring (and its neighbours) are replaced by other random values each iteration, and the worst fitness of all of the current species is reported together with the best worst species fitness found in any iteration so far. Figure 1 shows a typical run of the algorithm. The connected line "envelope" function represents the highest value of the worst species value found up to that particular iteration. The jumps from a previous maximum to the next mark the occurrence of an avalanche. The reason for this can be deduced from consideration of the distribution of values within the population. For the Bak-Sneppen model, the population has a uniform random distribution of fitness values between a current, lower bound, $\lambda_{min}(t)$ and 1, as might be represented by values within the solid outline in the histogram in Figure 2. For the lower bound to "jump" from its present value, $\lambda_{min}(t)$ to the value at the upper end of the filled region in the figure, $\lambda_{min}(t+1)$, *all* the species that currently lie in the filled region must migrate above that bound (and the histogram will expand upward to fill the shaded region in Figure 2.) This evidently cannot be achieved in a single iteration; several iterations will be required as successive species "cascade" to higher fitness levels.

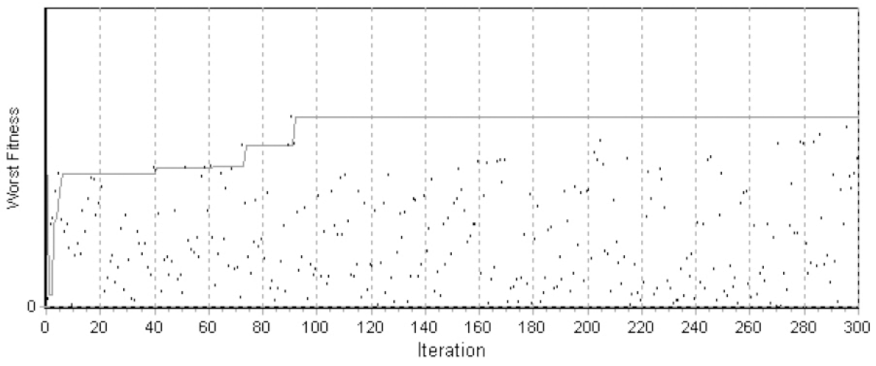


Fig. 1 The output of a typical run of the SOC algorithm. The connected "line" is the envelope function. Each point in the graph represents the worst value at that particular step.

To apply the concepts of SOC to optimisation problems, it is necessary to define a mapping between a fitness value and the structural components of a species. Essentially, this puts back the details that Bak and Sneppen

[1] The neighbour of the last species is the first species, and vice versa.

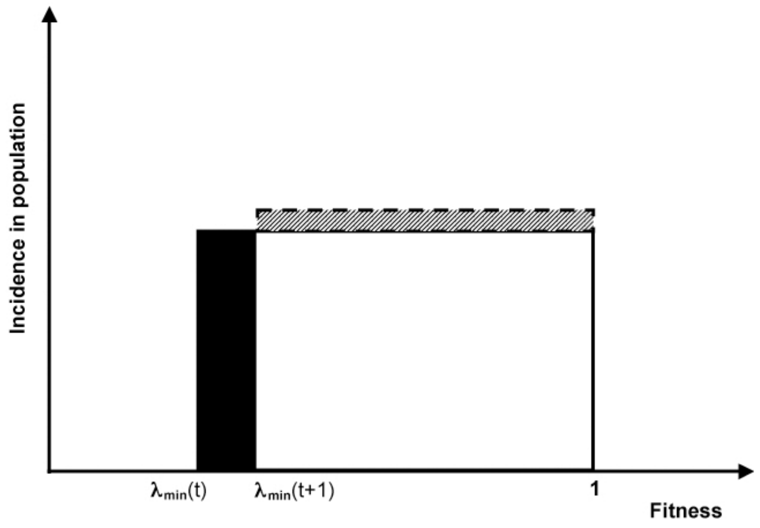


Fig. 2 Distribution of fitness values for a sample population of species in the Bak-Sneppen model.

Algorithm 1 The SOC algorithm.

```
Generate a random vector *species* in the range $[0, 1]$
**for** each generation **do**
    Find the species with the worst/lowest value
    Find the neighbour directly below this worst species
    Find the neighbour directly above this worst species
    Generate new random values for these three species
    Report the worst value of any species
    If this worst value is higher than any worst value found in previous iterations,
    update the best worst value found so far. Report the current best worst value.
**end for**
**end**
```

discarded in their model. One such example is extremal optimisation and is discussed in detail next.

3 Extremal Optimisation

Extremal optimisation is one of a number of emerging Nature inspired metaphors for solving combinatorial and continuous optimisation problems. As it is relatively new and unexplored, compared to other techniques such as ant colony optimisation (ACO) [13], genetic algorithms (GAs) [18] and particle swarm optimisation (PSO) [22], there exists wide scope to test and to extend its capabilities. Unlike its counterparts, the canonical algorithm manipulates a single solution rather than a population of solutions. Additionally, it never converges as the single solution is continually changing (see Section 5 for a demonstration of this).

EO is loosely based on the principles of the Bak-Sneppen model and simulates the notion that some species flourish while others do not [6, 8, 10]. This form of selection is also present at the genetic level. We can use a mapping between 'genes' (or species structural components) and 'solution components' to describe the general operation of EO to combinatorial optimisation problems. Solution components are the building blocks of the solution, some examples being an agent assigned to a particular job for generalised assignment, or the inclusion of an item in a knapsack for the knapsack problem. In the original version of the EO algorithm, at each iteration, the component whose fitness is worst, would be replaced by another solution component generated at random. In essence, however, this choice of always selecting the worst component to modify leads to too greedy a search, and consequently its performance was poor. Like other meta-heuristic algorithms, an element of randomness (in the form of probabilistic selection) was introduced. This became known as $\tau-$EO. Components are ranked from worst (rank 1) to best (rank n). The parameter τ and the rank controls the selection probability for each solution component [8]. This is achieved using Equation 1. It is evident that lower ranks will receive larger values than higher ranks.

$$P_i \propto i^{-\tau} \qquad 1 \leq i \leq n \tag{1}$$

Where:

i is the rank of the component,
P_i is the probability ($P_i = [0, 1]$ when normalised) that component i is chosen and
n is the number of components.

Values of τ close to or equal to zero produce a random search strategy. Conversely, allowing $\tau = \infty$ gives the original EO algorithm. Algorithm 2 shows the mechanics of a single $\tau-$EO iteration.

Algorithm 2 A single $\tau-$EO iteration. Note that vector P need only be calculated once according to Equation 1.

```
Rank the solution components from worst to best
$j$ = Select a ranked component using roulette wheel selection on normalised $P$
Assign $x_j$ a random (legal and different) value
end
```

This procedure is performed a fixed number of times or until a particular solution quality is reached.

3.1 Existing EO Applications

Compared to other recent meta-heuristics, particularly ACO and PSO, extremal optimisation has received relatively little attention. Below is a representative summary of existing applications of EO.

Boettcher and Percus [6, 8] have described and carried out limited experimentation on the travelling salesman problem (TSP). However, more successful application has been in graph (bi)partitioning [6] and the max-cut (spin glass) problems [8]. For these at least, EO can locate optimal and near optimal solutions for the investigated test cases and is comparable to other meta-heuristics.

Randall and Lewis [31] present an extended form of EO in which it is used as a sub-ordinate heuristic by another meta-heuristic known as Evolutionary Population Dynamics (EPD). Experiments on small multi-dimensional knapsack problem instances showed that EO with EPD achieved equal or better results than EO on almost all tests cases. The results using EO with EPD were also compared to tests using a standard ant colony optimisation solver. EO with EPD was been found to deliver near-optimal results faster than the existing ant colony algorithm.

Randall [29] has performed an initial investigation of EO for the generalised assignment problem (GAP). In that paper a simple population model for EO was presented along with a heuristic that altered solutions so as to reduce the amount of their infeasibility (a process known as *partial feasibility restoration*). This heuristic helped EO to produce very good quality solutions. In fact, both the canonical EO and population versions were able to find statistically significantly better solutions than a state-of-the-art ant colony system (ACS) implementation, a very efficient meta-heuristic for this problem.

A variation of the bi-partitioning problem used for community detection is solved using EO by Duch and Arenas [14]. Specifically, they use it to optimise the modularity and to identify the communities of complex networks. However, as reported in Xiaodong, Cunrui, Xiandong and Yanping [33], this implementation is sensitive to the initial solution, prone to being trapped in local optima, and not yet competitive with particle swarm optimisation.

Middleton [25] proposes a modification to standard EO that makes it better capable of solving the Ising spin glass problem. This is referred to as *jaded EO* and increases the fitness of a spin in proportion to the number of times it has been flipped. Empirical results, in comparison to standard EO, show that it is significantly better.

Beyond the applications to benchmark problems, some work has been done to adapt the standard EO algorithm to dynamic combinatorial optimisation. As an example, Moser and Hendtlass [26, 27] apply EO to a dynamic version of the composition problem. During the course of solving the problem with EO, it may undergo a variety of transformations to its structure and/or data. Despite the EO solver not being made aware of specific changes, it is able to adapt to them more readily than a standard ACS solver. This, however, was the reverse for the static version of the problem.

Moser and Hendtlass [28] propose an EO implementation for dynamic aircraft landing. This problem consists of two related parts, namely determining the order/permutation of planes to land on a single runway, and assigning landing timeslots for the planes as they enter the horizon of air traffic control (but taking other planes into account). The latter is a deterministic problem, so EO is used for the former. A set of K candidate solutions is generated, where K is the number of aircraft on the horizon. Each of these is generated by swapping the landing orders of two aircraft. These candidates are ranked according to how close the landing times resemble the planes' target times. A new solution is chosen according to EO's rules. The results showed that this EO implementation could outperform a range of meta-heuristic applications.

Galski, de Sousa, Ramos and Muraoka [17] present an EO algorithm to find the optimal design of a simplified configuration of a thermal control system for a spacecraft platform. The objectives are to minimise the difference between target and actual temperatures on radiation panels as well as to minimise battery heater power dissipation. Using EO, two possible solutions were found. As the authors combined both objectives into a single function, the designs were really only able to satisfy the first objective. Future work will optimise both objectives using a Pareto front approach.

Another novel application of the meta-heuristic is to the protein folding problem. Shmygelska [32] implements a two stage process in which EO finds a good starting solution, in terms of pairwise arrangements of amino acids, from which a local Monte Carlo based search can find a refined solution. The results compare favourably with a known random search technique specific to this problem.

EO has also been adapted to solve continuous optimisation problems by Zhou, Bai, Cheng and Wang [34]. This implementation concentrates on the Lennard-Jones clustering problem. EO is used as a global optimiser, selecting probabilistic worst components (or "atoms") to change and then using a gradient based local search to improve the solution. The difficulty is that their approach does not scale well. Nevertheless, it is competitive with other heuristic methods for smaller problems.

It is clear from the above survey that to date EO has really only been applied to problems that are relatively unconstrained. The mechanics of the algorithm, particularly those concerned with giving a solution component a new random value, make it difficult for EO to naturally process more constrained problems. One class of such problems, the assignment type problems (discussed next), contain a range of interesting real-world constraints (such as capacity and group related constraints). In light of this, the extensions of the EO paradigm proposed in this chapter will allow it to be more widely and generally applicable.

4 Assignment Type Problems

"Assignment Type Problems" (ATPs) [12] are a collection of optimisation problems for which a number of items are to be assigned to groups subject to a set of resource/capacity constraints. They include the generalised assignment, bin packing and capacitated hub location problems to name a few. These three problems are broadly representative of ATPs and will form the test problems for the experimental work within this chapter. Brief descriptions of each follow:

- *Generalised Assignment Problem* – The generalised assignment problem [24] is a problem in which jobs are assigned to agents for these agents to perform subject to capacity constraints. Each job may be performed by one agent only. The aim is to minimise the total cost of assigning the jobs to the set of agents.
- *Bin Packing Problem (BPP)* – The bin packing problem has different variations. The one considered here is the standard one-dimensional version [21]. A set of items, each of which has a particular weight, is packed into a number of bins. Each bin has the same weight capacity. Two objective functions are possible, both of which are used in this chapter. The first is used in Section 5 and treats bin packing as a constraint satisfaction problem in which the total amount of excess weight for a fixed number of bins, is minimised. The second is used in the remainder of the chapter and explicitly minimises the number of bins.
- *Capacitated Single Allocation Hub Location Problem (CSAHLP)* – The CSAHLP belongs to a general class of hub and spoke problems for which the aim is to efficiently transfer large quantities of commodities (such as passengers, mail and telecommunication traffic) between each node pair of a network. A subset of the nodes (called *hubs*) act as consolidation centers for bulk transfers. The CSAHLP is a difficult variant of this problem in which the number of hubs is not fixed, and each hub has a limited flow capacity. The mathematical formulation of it may be found in Equations 6 – 14 of Randall [30]. Furthermore, this paper describes a number of generic support heuristics (such as node-to-hub allocation and feasibility restoration). These will be used in our enhanced version of EO.

5 A Detailed Examination of EO on Bin Packing

As previously mentioned, EO is a relatively new heuristic under-explored compared to many others. Despite using a seemingly simple move mechanism, the following step-by-step analysis reveals that it does not always bias towards the most favourable solutions. An archetypal example of an assignment problem, bin packing, is used to demonstrate this behaviour. This is solved as a constraint satisfaction problem, in which the amount of excess weight in the bins is to be minimised, with zero being considered optimal.

EO changes its solution in a series of moves with the single solution moving through problem space. At any time there are a series of legal moves available to it. The move that is made is one that pushes away from the current position by randomly altering a poor component of the current solution. The direction in which the push takes the solution is random, but by always altering bad component values. The long term trend is towards better solutions (ones with lower constraint violation) but in a very non-monotonic way.

In terms of bin packing, EO will select one overfull bin and randomly take one item from that bin and transfer it to another randomly chosen bin. If the total amount of excess weight in all the overfilled bins is thus reduced this might be referred to as a good move, if the total amount of excess weight is increased this is a bad move (the occasional moves in which the total of excess weight does not change may be referred to as a neutral move). For the problem u120_00 [5, 16], checking all possible moves that could be made at each point of a thousand step search[2] through the problem space shows the ratio of good to bad moves to be 0.51 to 1. If on the other hand the bin from which an item is to be removed may be chosen probabilistically from the N available bins (ranked from worst to best), then the probability of choosing bin n is

$$P_n = \left(R\left(\frac{w_n}{\sum_{i=1}^{N} w_i} \right) \right)^{-\tau}$$

where w_i is the weight of bin i, R is the ranking function and τ, the only user specified parameter for EO, determines how much the algorithm concentrates on the most overfilled bins. If τ is set to 1 then overfull bins are treated equally. If τ is set to 1.4 as recommended in Boettcher and Percus [9] the good to bad ratio is changed to 1.17 to 1.

Knowing the ratio of good to bad moves is only part of the picture, one also needs to know how good and how bad these moves might be. Figure 3 shows histograms of both the good and bad moves for EO. Note that the distributions of the two types of moves are very different. The bad moves are worse (on average) than the good moves are good.

For comparison, consider the choices available to two other algorithms, random hill climbing and a genetic algorithm on this same problem.

Random hill climbing also uses a single solution and creates an endless series of random candidate solutions, and, with a certain probability, a candidate solution replaces the current solution if it is better (i.e., has fewer

[2] All the figures in this section are based on the results from one hundred independent repeats, each consisting of one thousand actual moves (sufficient to allow stagnation to occur if it is going to occur). At each actual move either all possible legal moves or one thousand random test moves are trialled but then reversed after the data have been collected for the statistics.

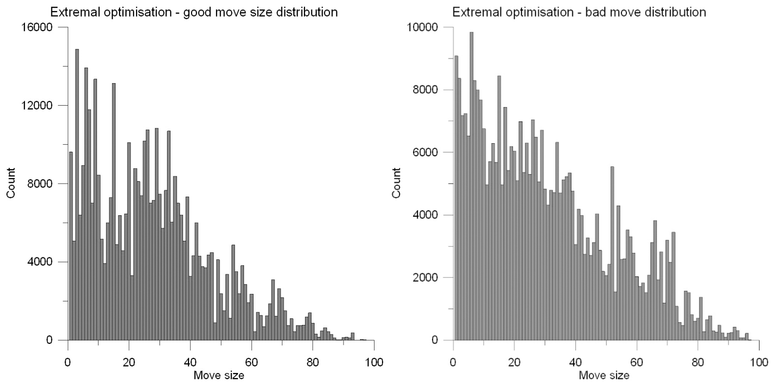


Fig. 3 Histograms of good (left) and bad (right) moves available to EO.

constraint violations). The replacement probability should be between 0.5 and 1, the higher this figure the greedier the algorithm. For u120_00 and a probability of 1, the ratio of good possible moves to bad possible moves is 0.08 to 1. Figure 4 shows the histograms of the good and bad moves for a replacement probability of 1 (i.e., always replace if better). The low number of good moves (cf. the number of bad moves) means that few changes are kept and that the time taken to find the relatively small sequence of good moves necessary to climb a local optimum may be high.

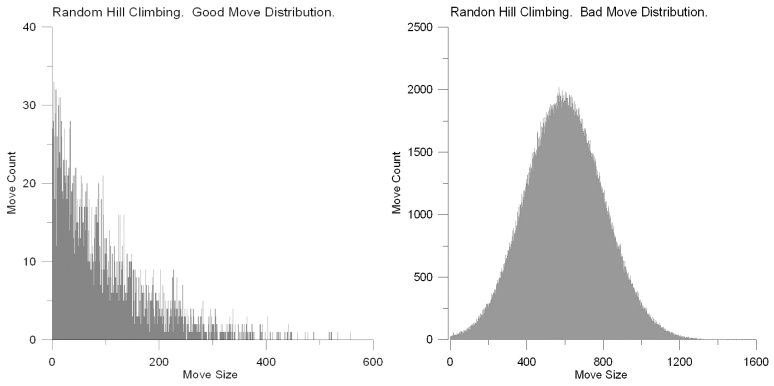


Fig. 4 Histograms of good (left) and bad (right) moves available to random hill climber.

A genetic algorithm uses a population of solutions and builds new solutions by combining parts of existing solutions together with some mutation. For the results quoted here, two parents are used and selected by a simple tournament between two candidates, with a probability of 0.7 of choosing the fittest candidate as a parent. Using single point crossover between the two parents with a probability of 0.1 of randomly moving one item in the solution to another bin (mutation), the ratio of good to bad moves for u120_00 is 5.7 to 1. Figure 5 shows the histograms of these good and bad moves. The high ratio of good to bad moves means that a GA will have the least trouble of the three algorithms considered in putting together the sequence of good moves to find a good quality solution.

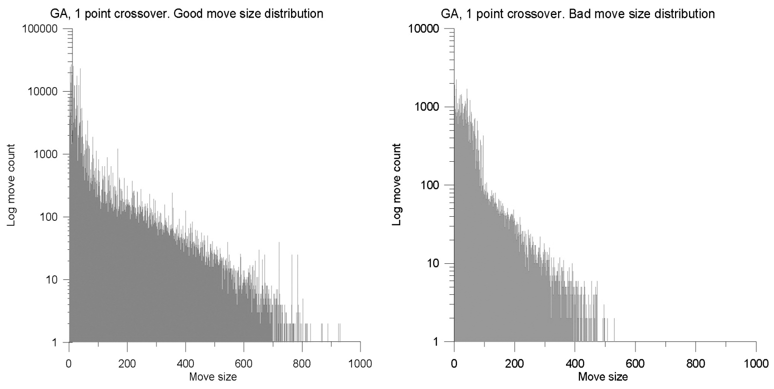


Fig. 5 Histograms of good (left) and bad (right) moves available to a genetic algorithm.

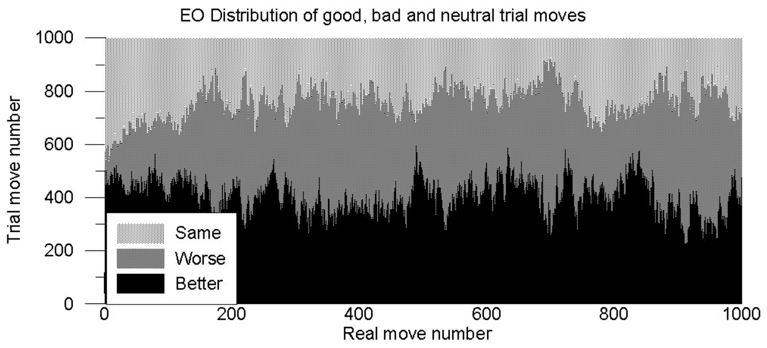


Fig. 6 Relative availability of good, bad and neutral moves during a run of EO.

Figures 6, 7 and 8 show the relative proportion of good, bad and neutral moves available at each of the thousand steps that make up a single run of each algorithm. Figure 6 shows that the relative prevalence of these moves for EO does not vary markedly during the run, while Figure 7 shows that

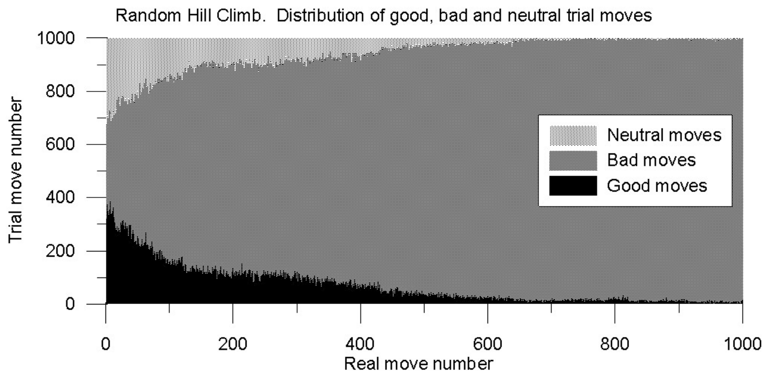


Fig. 7 Relative availability of good, bad and neutral moves during a run of random hill climber.

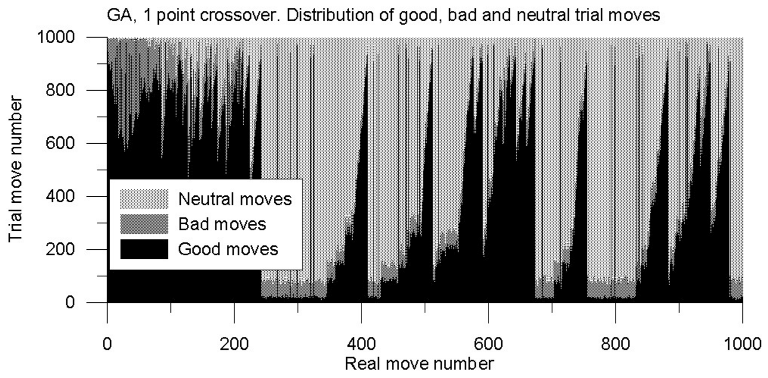


Fig. 8 Relative availability of good, bad and neutral moves during a run of a genetic algorithm.

the random hill climber initially has many good moves available but rapidly stagnates in a local suboptimal position. Initially the GA has many good moves available but then stagnates. There follows a cyclic series of steps. At each step an increasing number of good moves become available as more of the population climb a (probably) suboptimal peak. There is then a period of stagnation ending with a sudden breakthrough as progress is made towards a better (but probably still sub-optimal) peak after which the cycle repeats. With each breakthrough the fitness of the local sub-optimal peak improves so that the probability of another breakthrough decreases (i.e., the time between breakthroughs increases).

Considering that it will take a number of good moves in succession – a number that depends on how large each of the good moves is – it is clear that the probability of this occurring is far higher for EO (with the good to bad ratio of 1.17:1) than with random hill climbing (0.08:1) except in the early stages of the search. A genetic algorithm has more good moves available with an overall average of 5.2:1, a ratio that is far higher for some short periods. What is in favour of using EO is the fact that, unlike the other two (greedier) algorithms, it does not reach one (or cycle round a series of) local sub-optimal position(s) and then stagnate (forever in the case of the random hill climber or until a fortuitous breakthrough occurs for the genetic algorithm). EO's single solution never stagnates but continually moves in problem space as is evidenced by the fact that the good to bad ratio is consistent during the run.

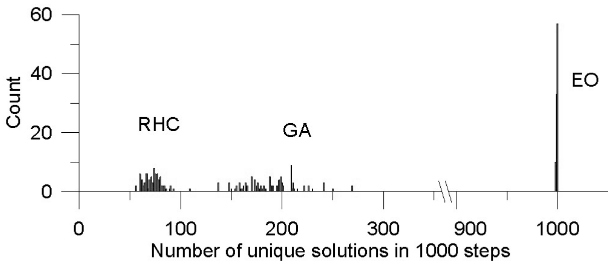


Fig. 9 The number unique places visited by each algorithm during 1000 steps. The histograms are built from the results of 1000 independent repeated runs.

Figure 9 shows that EO very rarely revisits previously explored positions, unlike the other two algorithms. The low probability of EO constructing a series of good moves means that EO should not be expected to work well if run alone. The fact that it does not stagnate suggests that it would be a good component of a meta algorithm that:

- contains another element that can lift the good to bad ratio so that EO will be driven further up local peaks (as long as this does not also inhibit EO's ability to migrate endlessly through problem space), and
- runs other, more local, search algorithms in collaboration with EO, primarily relying on EO to find regions of interest for these other techniques to explore more fully.

An endless series of random solutions would show a very low probability of revisiting the same place in problem space in any reasonable length run and one must ask what advantage there is in using EO over a series of fully random solutions. The answer lies in the average quality of the places in solution space visited by the two. Table 1 shows that there is a clear difference in the average quality of the start points each algorithm would provide for other, more local, search algorithms while Figure 10 shows how the fitness varies during a typical run. Although the ratio of the fitnesses is only a little over 2, if fitness was to be graphed against the number of solutions with each fitness value, an inverse bell shaped distribution would be produced.

If the number of solutions with this or worse fitness is considered, extremal optimisation is ahead by orders of magnitude.

	Random Moves	Extremal Optimisation
Average fitness	1833.3	783.6

Table 1 Average fitness of a thousand moves. As this is a minimisation problem, the smaller value obtained by EO is better.

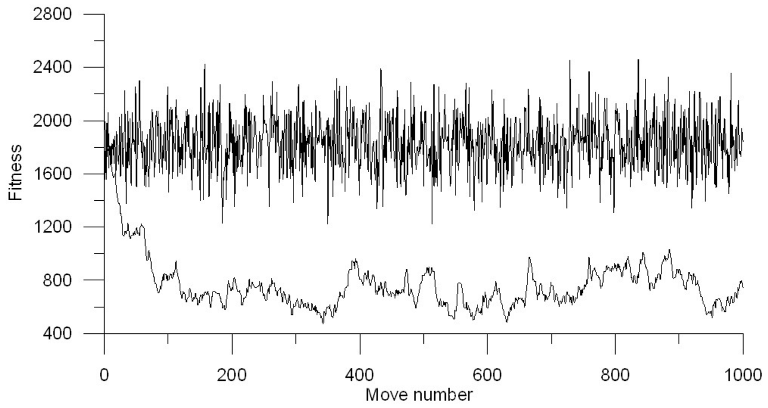


Fig. 10 The fitness of the single solution during the steps of the EO algorithm (lower trace) compared with one thousand randomly chosen solutions. Note the upper trace is not that of a random hill climber.

The discussion above has been written using data obtained from runs attempting to solve a bin packing problem. A similar series of runs, this time using a graph colouring problem, exhibit similar behaviour. This indicates that the observations made are predominantly a result of the behaviour of the algorithms used and not so much of the problems themselves. Thus the observations might be expected to substantially hold for a wide range of assignment problems.

6 Applying EO to ATPs

The previous section showed the performance of canonical EO on one of the test problems. EO was capable of exploring search space nearly without revisiting previous solutions. However, the results were not of an acceptable standard. Too many of the potential moves lead to poor quality solutions. Consequently, the level of constraint violation (for that particular version of bin packing) was such that feasible solutions could not be easily produced. Given these results and the fact that there has been relatively little application of EO to combinatorial problems, there exists scope to extend it while still retaining its fundamental characteristics. This section describes modifications and enhancements to EO that make it better able to solve assignment type problems.

There are three important elements that EO potentially needs to help it become competitive with more established meta-heuristics. These are a) transition operators and constraint handling techniques, b) a population framework and c) local search.

6.1 Transition Operators and Constraint Handling

The related topics of of transition operators and constraint handling within meta-heuristic search algorithms have been much discussed and explored in the literature. Three broad methods of constraint handling have often been applied:

1. *Use a penalty approach* – Penalise the objective value according to the amount of overall constraint violation.
2. *Allow the solver to search across feasible and infeasible space* – Report the best feasible solution at the end of the search.
3. *Restore feasibility* – Using a special purpose algorithm to transform an infeasible solution to a feasible one.

Method 1 has been used in many meta-heuristic implementations with mixed success. The major difficulty is to determine the best form and weightings of the penalties. A recent penalty based EO [19] showed that it is nearly, but not quite, competitive with state-of-the-art heuristics.

Methods 2 and 3, however, can easily be applied to EO, with Method 2 particularly taking advantage of the natural EO algorithm (as shown in Randall and Lewis [31] and Randall [29]). As EO only makes a small change at each iteration – allowing many transitions to be made in a computationally reasonable time – it may not matter that the solution is feasible at all times. Overall, many feasible solutions will be potentially produced depending on the difficulty of the constraints. Increasing the proportion of feasible solutions may be accomplished by the use of the partial feasibility restoration algorithm (discussed next). At each iteration of the algorithm, the feasibility status of the solution is determined. The choice of solution component, using EO's rules, is as follows for two of the test problems:

- *A feasible solution* – An EO move is performed to optimise the objective function. A move changes a poor solution component value to a random one. For the GAP, a poor assignment of an agent to a job is replaced by the assignment of that job to a random agent. However, for the BPP, the objective is to minimise the number of groups, therefore a random item is taken from a relatively small sized bin and reassigned to a random one.

- *An infeasible solution* – The focus changes to moving the solution back to a feasible state. As such, a component value to change is chosen according to the amount of infeasibility it contributes to the solution. This will vary from problem to problem. The specifications for each problem are:
  - GAP: This is given as the amount of resources required for a job assigned to an agent. Only overloaded/infeasible agents are examined. The job assignments with resource requirements that most closely match the the amount of agent infeasibility are more likely to be chosen.
  - BPP: A light item from an overfull bin is most likely to be selected. This is the case as it is easier to move light items to fill the spare capacity of other bins. A number of such moves will decrease its infeasibility.

The CSAHLP is a different matter. Based on the work of Randall [30], there exists a node allocation heuristic and feasibility restoration algorithm. The former will best allocate non-hub nodes to hub nodes. If this fails, the latter will restore feasibility.

6.1.1 Partial Feasibility Restoration

It may take a considerable number of EO iterations to move from infeasible space to feasible space using the methods mentioned above. To effectively make use of the available EO iterations, a general purpose heuristic can be developed to reduce the amount of infeasibility of a solution (the prototype for which is given in Randall [29]). Partial feasibility restoration is a simple, non-degenerative, parameter-free process. In some ways, it resembles standard local search, except it tries to minimise infeasibility rather than optimise the problem's objective. It is represented by Algorithm 3. Note that it is applicable across ATPs and does not guarantee that a feasible state will result. It is an $O(MN)$ algorithm where M is the number of groups, and N is the number of items. This algorithm will be used for the GAP and BPP. As mentioned previously, CSAHLP uses its own full feasibility restoration algorithm.

The amount of contribution of infeasibility of an item within a group is, naturally enough, calculated differently for different problems. In the case of BPP, it is an item whose weight most nearly matches the amount of overloading for a bin. For the GAP, the item is simply the one whose resource requirement covers the amount of the agent's infeasibility.

Algorithm 3 Generalised partial feasibility restoration for assignment type problems.

```
for all groups do
  if this group is infeasible then
    Determine the item within the group whose resource requirement most closely matches the group's amount of infeasibility
    Find a new group that can take this item without itself becoming infeasible
    if such a group exists then
      Update the solution, its cost (if any) and the amount of its infeasibility
    else
      Do nothing
    end if
  end if
end for
```

6.2 A Population Model

A simple population extension mechanism to EO was presented by Randall [29]. In it, a fixed number of solutions/individuals form a population. At preset intervals throughout the execution of the search, a population interaction would occur. This interaction identified the worst member of the population (as measured by the objective function) and deleted it. Besides it, the two nearest neighbours (in terms of common solution component values) were also removed in accordance to the Bak-Sneppen model. Three new solutions were randomly generated, added to the population, and the search resumed.

While the above approach produced statistically significantly superior results to a standard implementation on GAP instances, its main drawback was that it required the user to specify the number of times interactions would occur throughout the search process. It would be better if the system could calculate this information for itself based on dynamic information about the search.

As EO does not converge, solution similarity between the population members would not be a natural cause to trigger an interaction. It is better to look for a population in which solution qualities widely diverge. This indicates that the least fit members should be eliminated. The equivalent in Nature would be a cull of the weak of a population so as to maximise the overall chances of a species' survival. The probability that an interaction will occur at an iteration can be calculated according to Equation 2. Note as well that the probability increases as the iterations pass since the last population interaction.

$$p = \left(1 - \frac{cost(best)}{cost(worst)}\right)^{1/l} \tag{2}$$

Where:

l is the number of iterations since the last population interaction occurred.

In terms of the solution replacements, a small augmentation of Randall [29] is proposed here. Instead of replacing all three solutions (i.e., the worst solution and its two closest neighbours) with random solutions, one of these is now a copy of the best found solution. This balances the need for diversity of solutions against that of exploration around the best (known) solution. Again, as EO does not converge, there is no danger of creating a population of similar, stagnating solutions. Rather, it will allow EO to search more extensively in good neighbourhoods.

6.3 Local Search

Meta-heuristic search algorithms, on the whole, are capable of coarse grain search. In essence, they provide very good starting points for fine grained searching known as *local search*. This point was demonstrated for EO in Hendtlass and Randall [20]. As such, local search is solely driven toward optimising the objective function, moving only in feasible space, and making purely greedy moves. However, it is useful for obtaining locally optimal solutions, which standard EO does not guarantee. Local search can be performed each time a feasible solution is produced.

The details of the local search implementation for each problem is given below.

- BPP – The algorithm of Alvim, Aloise, Glover and Ribeiro [1] (as reported in Levine and Ducatelle [23]) initially determines the two least loaded bins. The items from these are moved to a "free list", and the bins removed. Three types of exchange operations are attempted. In the first, two items from the free list are exchanged with two items from a bin (subject to the capacity constraint being satisfied). This process is repeated for all bins and this is attempted for combinations of two bin items for one free item and then one bin item for one free item. After this, all free items that can be fit into existing bins have been. A new bin is created with the left over items. The aim of this procedure is to reduce the number of overall bins by at least one. Even though this algorithm is designed for the BPP, it can equally be applied to problems such as graph colouring. A more detailed description of the algorithm (as well as an example) can be found in Levine and Ducatelle [23].
- GAP – Two effective operators [29] will be used. "Move" moves an item from one agent to another. The job and agent are chosen such that the (negative) change in the objective function is the greatest. This is a variable length search stopping when an improving move cannot be found. "Swap" works in a similar way except that at each iteration, two items are chosen

such that their swap will lead to the most improvement in the objective function.

- CSAHLP – There are six local search operators that are appropriate for the CSAHLP [15]. Note that some of the operators discussed below are described in terms of 'clusters'. A cluster refers to a group of nodes that are all assigned to a particular hub.

  1. *Relocate Node* – A node is reassigned to a different hub.
  2. *Swap Nodes* – Two nodes swap the clusters to which they belong.
  3. *Create a New Cluster* – A non-hub node is made a hub node. No nodes (apart from itself) are assigned to this new hub.
  4. *Split a Cluster* – Half of the nodes of a cluster are reassigned to a new cluster. A node from the new cluster is chosen as the hub.
  5. *Merge Clusters* – Two clusters are merged into one. The hub node of one of the clusters becomes the hub node of the new, enlarged cluster.
  6. *Relocate Hub* – The hub of a cluster is reassigned to another node.

  According to Randall [30], the most effective way to apply these operators, at each (EO) iteration, is to first randomly order the transition operators in a list. Each operator is applied, attempting all possible transitions, and all improving moves are kept. This process is continued until an improving move cannot be produced by any of the operators.

7 Computational Experiments

In this section, the effectiveness of EO, along with its support heuristics and population model, is evaluated on benchmark GAP, BPP and CSAHLP problems.

The test problem instances are drawn from three benchmark sets from the literature:

- GAP – These are the large-sized set of Chu and Beasley [11]. They have also been used in the study by Randall [29].
- BPP – The test set of problems from the OR-Library [5] are used. These range in size from 120 to 500 items.
- CSAHLP – These instances are from the benchmark set proposed by Ernst and Krishnamoorthy [15]. A fuller explanation of these problems can be found in Randall [30].

The computing platform used to perform the experiments is a 3GHz Pentium 4 based PC. Each problem instance is run across ten random seeds. The experimental programs are coded in the C language and compiled with `gcc`.

Each problem instance is allowed to run for 500000 iterations. Two sets of results are generated, one for the single solution version and one for the population variant. In terms of the population approach, the total number of iterations is divided amongst the population members, rather than being that number of generations. This helps to ensure a fairer comparison with the single EO versions.

The two other parameters that need to be set are τ and the population size. A value of 1.4 is used for τ as this has been found to give good quality results in a number of previous studies [9, 20, 29, 31] and balances the need for exploration with that of exploitation. Twenty individuals constitute a population in these experiments. However, further investigation will explore the effect of different values of both these parameters.

Tables 2, 3 and 4 show the results for GAP, BPP and CSAHLP respectively. The results are expressed as relative percentage deviations (RPD) from the optimal/best-known cost, i.e., $RPD = \frac{cost}{optimal} \times 100$. Thus a result of 0 corresponds to the optimal/best-known cost. “min”, “med” and “max” denote minimum, medium and maximum respectively.

Table 2 The single and population results for the GAP. Note that the first number in the problem name (first column) represents the number of agents while the second is the number of jobs. For instance, ‘A5-100’ is of Type A with 5 agents and 100 jobs. The single results are reproduced from Table 2 (last set) of Randall [29].

		Single			Population		
Name	Optimal	min	med	max	min	med	max
A5-100	1698	0	0	0	0	0	0
A5-200	3235	0	0	0	0	0	0
A10-100	1360	0	0	0	0	0	0
A10-200	2623	0	0	0	0	0	0
A20-100	1158	0	0	0	0	0	0
A20-200	2339	0	0	0	0	0	0
B5-100	1843	0.71	1.03	1.41	0	0.3	0.33
B5-200	3553	0.42	0.48	0.56	0.06	0.07	0.2
B10-100	1407	0	0	0	0	0	0
B10-200	2831	0.6	0.76	1.02	0	0.11	0.18
B20-100	1166	0.09	0.17	0.26	0	0	0.09
B20-200	2340	0.17	0.21	0.26	0	0.11	0.21
C5-100	1931	0.36	0.6	0.78	0	0.05	0.31
C5-200	3458	0.29	0.52	0.67	0	0.04	0.14
C10-100	1403	0.71	1.1	1.28	0.07	0.14	0.29
C10-200	2814	0.46	0.82	1.03	0	0	0.21
C20-100	1244	0.72	1.05	1.13	0	0.08	0.32
C20-200	2397	0.92	1.17	1.29	0	0	0.13
D5-100	6373	1.37	1.54	1.65	0.25	0.45	0.69
D5-200	12796	1.52	1.63	1.71	0.12	0.27	0.61
D10-100	6379	2.15	2.56	2.76	0.49	0.74	1.47
D10-200	12601	1.24	1.54	1.6	0	0.04	0.13
D20-100	6269	2.14	2.47	2.58	0.33	0.45	1.36
D20-200	12452	1.55	1.69	1.81	0	0.17	0.48

Table 3 The single and population results for the BPP. Note that the problem name (first column) indicates the number of items. For instance, u120_00, is the first problem of the set of problems that have 120 items in each.

		Single			Population		
Name	Optimal	min	med	max	min	med	max
u120_00	48	0	0	2.08	0	0	0
u120_01	49	0	0	0	0	0	0
u120_02	46	0	0	0	0	0	0
u120_03	49	0	2.04	4.08	1.02	2.04	2.04
u120_04	50	0	0	0	0	0	0
u120_05	48	0	0	0	0	0	2.08
u120_06	48	0	0	2.08	0	0	2.08
u120_07	49	0	0	0	0	0	0
u120_08	51	0	0	1.96	0	0	0
u120_09	46	0	1.09	2.17	0	2.17	2.17
u120_10	52	0	0	0	0	0	0
u120_11	49	0	0	2.04	0	0	0
u120_12	48	0	2.08	2.08	0	2.08	2.08
u120_13	49	0	0	0	0	0	0
u120_14	50	0	0	0	0	0	0
u120_15	48	0	0	2.08	0	0	0
u120_16	52	0	0	1.92	0	0	0
u120_17	52	0	0	5.77	0	1.92	1.92
u120_18	49	0	0	0	0	0	0
u120_19	50	0	0	2	0	0	0
u250_00	99	0	0.51	1.01	0	1.01	1.01
u250_01	100	0	0	1	0	0	1
u250_02	102	0	0.49	0.98	0	0.98	0.98
u250_03	100	0	0	0	0	0	1
u250_04	101	0	0	0.99	0	0.99	0.99
u250_05	101	0	0.99	1.98	0.99	0.99	1.98
u250_06	102	0	0	0	0	0	0
u250_07	104	0	0	3.85	0	0	0
u250_08	105	0.95	0.95	0.95	0.95	0.95	1.9
u250_09	101	0	0.99	2.97	0.99	0.99	0.99
u250_10	105	0	0	1.9	0	0.95	0.95
u250_11	101	0.99	0.99	0.99	0.99	0.99	0.99
u250_12	106	0	0	0.94	0	0	0.94
u250_13	103	0	0	0.97	0	0	0.97
u250_14	100	0	0	1	0	1	1
u250_15	105	0.95	0.95	3.81	0.95	1.9	1.9
u250_16	97	0	0	3.09	1.03	1.03	1.03
u250_17	100	0	0	1	0	0	0
u250_18	100	1	1	1	1	1	1
u250_19	102	0	0	0.98	0	0	0.98
u500_00	198	0.51	0.51	2.53	0.51	1.01	1.01
u500_01	201	0.5	0.5	0.5	0.5	1	1
u500_02	202	0	0.5	0.99	0.5	0.74	0.99
u500_03	204	0.49	0.49	1.47	0.98	0.98	0.98
u500_04	206	0	0.49	1.94	0.49	0.49	0.97
u500_05	206	0	0.97	2.91	0.49	0.73	0.97
u500_06	207	0.48	0.72	1.45	0.97	0.97	1.45
u500_07	204	0.49	1.23	3.43	0.98	1.47	1.96
u500_08	196	0	0.26	1.02	0.51	0.51	1.02
u500_09	202	0	0	0.5	0	0.5	0.99
u500_10	200	0	0.5	0.5	0.5	0.5	1
u500_11	200	0.5	0.5	3	0.5	1	1.5
u500_12	199	0.5	0.5	1.01	0.5	1.01	1.01
u500_13	196	0	0.51	1.53	0	0.51	0.51
u500_14	204	0.49	0.49	8.33	0.49	0.49	0.98
u500_15	201	0.5	0.5	0.5	0.5	0.5	1
u500_16	202	0	0	0.99	0	0.5	0.5
u500_17	198	0.51	0.51	0.51	0.51	1.01	1.01
u500_18	202	0	0.5	1.98	0.99	1.49	1.49
u500_19	196	0.51	0.51	1.02	0.51	1.02	1.53

Table 4 The single and population results for the CSAHLP. Note that the problem name indicates the number of nodes in the instance. The cost of 50TT corresponds only to the best known cost. For this problem, the single version of the algorithm could only generate solutions in two of its runs, while the other eight could not generate feasible solutions at all.

		Single			Population		
Name	Optimal	min	med	max	min	med	max
10LL	224250.1	0	0	0	0	0	0
10LT	250992.3	0	0	0	0	0	0
10TL	263399.9	0	0	0	0	0	0
10TT	263399.9	0	0	0	0	0	0
20LL	234691	0	0	0	0	0	0
20LT	253517.4	0	0	0	0	0	0
20TL	271128.2	0	0	0	0	0	0
20TT	296035.4	0	0	0	0	0	0
25LL	238978	0	0	0	0	0	0
25LT	276372.5	0	0	0	0	0	0
25TL	310317.6	0	0	0	0	0	0
25TT	348369.2	0	0	0	0	0	0
40LL	241955.7	0	0	0	0	0	0
40LT	272218.3	0	0	0	0	0	0
40TL	298919	0	0	0	0	0	0
40TT	354874.1	0	0	0	0	0	0
50LL	238520.6	0	0	0	0	0	0
50LT	272897.5	0	0	0	0	0	0
50TL	319015.8	0	0	0	0	0	0
50TT	417441	0	0	0	0	0.06	0.41

As reported in Randall [29], the single solution results in Table 2 were statistically better than a state-of-the-art ACO implementation. In all cases, however, the population approach for the GAP was able to find equivalent or better quality results for all problem instances. The new population results are also superior to the previous population results [29]. This may be attributed to the new model's feature of allowing more concentrated search around best found solutions. This is only practicable because EO will not converge on these solutions again – as would be the case for other meta-heuristics.

For both the single and population versions of EO for bin packing, the algorithms could achieve very good quality results, being at most a few percent away from the optimal results. Unlike the GAP, there is no clear distinction between the single and population results for the minimum results. However, the population version's maximum is less deviant than the other as it is always at most two percent away from the optimal result. These results are very comparable to the ACO implementation of Levine and Ducatelle [23].

The performance of EO on CSAHLP in comparison to ACO [30] was more or less equivalent. Both methods required relatively few iterations to achieve optimal solutions. Much of this may be attributed to the powerful local search heuristics. However, as noted in Randall [30], ACO (and therefore EO) provide a powerful coarse grain optimisation framework. This is evidenced by the fact that a random descent heuristic could not achieve the same level of results. Additionally, the work by Hendtlass and Randall [20] has shown, for bin packing, that pure local search extensions to EO are necessary to produce competitive computational results, despite the computational costs these may incur.

8 Conclusions

EO is a relatively new and simple meta-heuristic that is based on the elimination of poorly performing solution components, rather than the explicit incorporation of necessarily good values. As such it does not converge on certain solutions, giving it some advantages over more traditional meta-heuristics such as GAs and ACO. However, application of the canonical algorithm will often lead to relatively poor performances, this having been demonstrated in this chapter with the bin packing problem (Section 5). In order to lift performance so that EO becomes comparable with other algorithms requires additional support heuristics. Generalised algorithms have been proposed herein that can handle constraints, partially restore feasibility and create and maintain a population of solutions. These combined with local search have indeed shown that EO is capable of producing very good quality solutions.

An area that we are currently investigating is concerned with the management of populations. One of the key questions is that of population size. It may be possible to allow the population to shrink or grow according the progress of the search.

References

[1] Alvim, A., Aloise, D., Glover, F., Ribeiro, C.: Local search for the bin packing problem. In: Extended Abstracts of the 3rd Metaheuristics International Conference, pp. 7–12 (1999)
[2] Bak, P.: How Nature Works. Springer, New York (1996)
[3] Bak, P., Sneppen, K.: Punctuated equilibrium and criticality in a simple model of evolution. Physical Review Letters 71, 4083–4086 (1993)
[4] Bak, P., Tang, C., Wiesenfeld, K.: Self-organized criticality: An explanation of 1/f noise. Physical Review Letters 59, 381–384 (1987)
[5] Beasley, J.: OR-Library: Distributing test problems by electronic mail. Journal of the Operational Research Society 41, 1069–1072 (1990)
[6] Boettcher, S., Percus, A.: Extremal optimization: Methods derived from co-evolution. In: Proceedings of the Genetic and Evolutionary and Computation Conference, pp. 825–832. Morgan Kaufmann, San Francisco (1999)
[7] Boettcher, S., Percus, A.: Combining local search with co-evolution in a remarkably simple way. In: Proceedings of the Congress on Evolutionary Computation, pp. 1578–1584. IEEE Service Center, Piscataway (2000)

[8] Boettcher, S., Percus, A.: Nature's way of optimizing. Artificial Intelligence 119, 275–286 (2000)
[9] Boettcher, S., Percus, A.: Extremal optimization for graph partitioning. Physical Review E 64 (2001)
[10] Boettcher, S., Percus, A.: Extremal optimization: An evolutionary local search algorithm. In: Bhargava, H., Ye, N. (eds.) Computational Modeling and Problem Solving in the Networked World. Interfaces in Computer Science and Operations Research, pp. 61–77. Kluwer Academic Publishers, Dordrecht (2003)
[11] Chu, P., Beasley, J.: A genetic algorithm for the generalised assignment problem. Computers and Operations Research 24, 17–23 (1997)
[12] Costa, D., Hertz, A.: Ants can colour graphs. Journal of the Operational Research Society 48, 295–305 (1997)
[13] Dorigo, M., Di Caro, G.: The ant colony optimization meta-heuristic. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 11–32. McGraw-Hill, London (1999)
[14] Duch, J., Arenas, A.: Community detection in complex networks using extremal optimization. Physical Review E 72 (2005)
[15] Ernst, A., Krishnamoorthy, M.: Solution algorithms for the capacitated single allocation hub location problem. Annals of Operations Research 86, 141–159 (1999)
[16] Falkenauer, E.: A hybrid grouping genetic algorithm for bin packing. Tech. Rep. CP 106 - P4, CRIF Industrial Management and Automation, 50 Av. F.D. Roosevelt, B-1050 Brussels, Belgium (1994)
[17] Galski, R., de Sousa, F., Ramos, F., Muraoka, I.: Spacecraft thermal design with the generalized extremal optimization algorithm. In: Orlande, H., Colaco, J. (eds.) Proceedings of Inverse Problems, Design and Optimization, pp. 61–75 (2004)
[18] Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Reading (1989)
[19] Gómez-Meneses, P., Randall, M.: Extremal optimisation with a penalty approach for the multidimensional knapsack problem. In: Li, X., et al. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 229–238. Springer, Heidelberg (2008)
[20] Hendtlass, T., Randall, M.: Extremal optimisation for bin packing. In: Li, X., et al. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 220–228. Springer, Heidelberg (2008)
[21] Kampke, T.: Simulated annealing: Use of a new tool in bin packing. Annals of Operations Research 16, 327–332 (1988)
[22] Kennedy, J., Eberhart, R.: The particle swam: Social adaptation in social information-processing systems. In: New Ideas in Optimization, pp. 379–387. McGraw-Hill, London (1999)
[23] Levine, J., Ducatelle, F.: Ant colony optimisation and local search for bin packing and cutting stock problems. Journal of the Operational Research Society 55, 705–716 (2004)
[24] Martello, S., Toth, P.: An algorithm for the generalized assignment problem. In: Proceedings of the 9th International Federation of Operational Research Societies' Conference, Hamburg, Germany, pp. 589–603 (1981)
[25] Middleton, A.: Improved extremal optimization for the ising spin glass. Physical Review E 69 (2004)

[26] Moser, I., Hendtlass, T.: On the behaviour of extremal optimisation when solving problems with hidden dynamics. In: Ali, M., Dapoigny, R. (eds.) IEA/AIE 2006. LNCS, vol. 4031, pp. 292–301. Springer, Heidelberg (2006)

[27] Moser, I., Hendtlass, T.: Solving problems with hidden dynamics - comparison of extremal optimisation and ant colony system. In: Proceedings of the Congress on Evolutionary Computation, pp. 1248–1255 (2006)

[28] Moser, I., Hendtlass, T.: Solving dynamic single-runway aircraft landing problems with extremal optimisation. In: Proceedings of the IEEE Symposium on Computational Intelligence in Scheduling, pp. 206–211 (2007)

[29] Randall, M.: Enhancements to extremal optimisation for generalised assignment. In: Randall, M., Abbass, H.A., Wiles, J. (eds.) ACAL 2007. LNCS, vol. 4828, pp. 369–380. Springer, Heidelberg (2007)

[30] Randall, M.: Solution approaches for the capacitated single allocation hub location problem using ant colony optimisation. Journal of Computational Optimization and Applications 39, 239–261 (2008)

[31] Randall, M., Lewis, A.: An extended extremal optimisation model for parallel architectures. In: Proceedings of the 2nd IEEE International e-Science and Grid Computing Conference (Workshop on Biologically-inspired Optimisation Methods for Parallel and Distributed Architectures: Algorithms, Systems and Applications). IEEE Computer Society, Los Alamitos (2006)

[32] Shmygelska, A.: An extremal optimization search method for the protein folding problem: The Go-model example. In: Proceedings of the Companion to the Genetic and Evolutionary Computation Conference, pp. 2572–2579. ACM, New York (2007)

[33] Xiaodong, D., Cunrui, W., Xiangdong, L., Yanping, L.: Web community detection model using particle swarm optimization. In: Proceedings of the Congress on Evolutionary Computation, pp. 1074–1079 (2008)

[34] Zhou, T., Bai, W., Cheng, L., Wang, B.: Continuous extremal optimization for Lennard-Jones clusters. Physical Review E 72 (2006)

Niching for Ant Colony Optimisation

Daniel Angus

Abstract Evolutionary Computation niching methods, such as Fitness Sharing and Crowding, are aimed at simultaneously locating and maintaining multiple optima to increase search robustness, typically in multi-modal function optimization. Such methods have been shown to be useful for both single and multiple objective optimisation problems. Niching methods have been adapted in recent years for other optimisation paradigms such as Particle Swarm Optimisation and Ant Colony Optimisation. This paper discusses niching techniques for Ant Colony Optimisation. Two niching Ant Colony Optimisation algorithms are introduced and an empirical analysis and critical evaluation of these techniques presented for a suite of single and multiple objective optimisation problems.

1 Introduction

In natural ecologies, a population of organisms is rarely spread uniformly (within its environment), but rather is typically distributed across a wide spatial area and divided into local or sub-groups. Resources available to individuals across a geographical distribution can differ, and sub-groups of a species may specialise to exploit these differences. The effect of this speciation (or population level natural selection) is referred to as 'niching' in the field of population ecology and population genetics [18, 37].

In a computational sense, niching may permit a more effective use of available resources by a search algorithm by either implicitly or explicitly dividing

The University of Queensland
Brisbane
Queensland
Australia
d.angus@uq.edu.au

and searching different areas of the search space in parallel [31]. Such automatic resource re-allocation is useful in preserving population diversity for an extended search time. Niching techniques have proved particularly useful for problems that require multiple solutions to be located for a search algorithm, such as multi-modal and multi-objective optimisation problems.

Niching as an Evolutionary Computation (EC) concept was first formally applied to the Genetic Algorithm (GA), but has also been applied to other EC algorithms such as Particle Swarm Optimisation (PSO) [5, 6]. Some of the better known niching methods include Crowding [29, 30], Fitness Sharing [20, 21] and Clearing [33].

Previous work has applied Fitness Sharing and Crowding to Ant Colony Optimisation (ACO) [1, 2, 3, 4], all with promising results. This chapter discusses these works with a review of a variety of applications to single and multiple objective problem domains. The chapter is organised into three main sections which discuss, in order, the fundamentals of niching, an introduction to ACO and how to apply niching principles to ACO, and finally an empirical and theoretical analysis of several niching ACO algorithms.

2 Niching

Solution diversity is often mentioned as an important factor in population-based algorithm design. For some problems diversity may not be required to obtain optimal solutions while in others it is critical. In this section a particular diversity preservation technique called *niching* is introduced. Some historical aspects of niching in EC are introduced and examples of two specific niching techniques are presented.

2.1 Niching in Evolutionary Computation

The canonical GA tends to focus its search in an individual area of the search space. If multiple near optimal solutions are sought then such algorithms may prove ineffective in achieving this goal. Niching aims to diversify the search focus of an EC algorithm to multiple areas of the search space. In Goldberg [20] three key features of niching algorithms are presented:

1. Stable maintenance of subpopulations. Once an optimal area of the search space has been located a niching algorithm should maintain several population members in that location so as not to lose (forget) this area of interest.
2. The size of a subpopulation decreases according to the fitness of the area of interest. Considering the limited resources of an algorithm, exploration

of any area of the search space should be proportional to the potential quality of solutions returned from a niche.

3. Subpopulations should not compete. Since resources are most likely to be of a fixed size, a niching algorithm should not allow subpopulations to 'fight' for dominance of one area of the search space.

2.2 Crowding: Modifying the Replacement Mechanism

The *Crowding Factor* model was introduced as a diversity maintenance scheme by De Jong [11]. This scheme was not a niching method according to the criteria outlined in Sec. 2.1, as it was used to increase population diversity, not to locate and maintain multiple optimal areas of the search space [26]. However, the Crowding Factor model laid the foundation for much of the later work on niching and was most significantly reworked as a niching strategy by Mahfoud [29, 30].

Mahfoud's technique, *Deterministic Crowding*, was designed to slow convergence and maintain diversity by limiting dominant building blocks in a population. The replacement policy in Deterministic Crowding involves a candidate solution competing (based on solution quality) for a place in the population with the most similar of its parent solutions. If a candidate solution is better than its most similar parent, the parent is replaced, otherwise the candidate solution is discarded.

Another variation of Crowding is *Restricted Tournament Selection*(RTS) [26]. RTS is similar to Deterministic Crowding, but instead of a candidate solution only being compared against its parents, it is compared to a subset of the entire population. The size of this subset in called the window size and the most similar solution from this subset is sought and is replaced if the candidate solution is of higher fitness.

2.3 Fitness Sharing: Modifying the Selection Mechanism

Instead of modifying the replacement mechanism to introduce niching behaviour, as in Crowding, Fitness Sharing [21, 20] modifies the selection mechanism. Since most Genetic Algorithms use a fitness proportionate selection mechanism to select parent solutions, Goldberg and Richardson [21] found that modifying the selection of parents will avoid convergence to one area of the search space thereby introducing niche formation.

Fitness Sharing derates[1] solutions which occupy the same or a similar position in the search space. For example the adjusted quality (Q') of two solutions which occupy the same position in the search space will be half of their original quality (Q). The result of this adjustment is to remove the selection bias present through a solution being represented multiple times. The removal of this bias allows simultaneous convergence to multiple areas of the search space.

Niche formation, specifically with regard to Fitness Sharing, was explained by Goldberg and Richardson [21] using a variation of the k-armed bandit problem [27, 11, 38]. The problem involves a poker machine with k handles, each handle having set pay-out odds. There is also a population of gamblers, who wish to maximise their individual winnings from the poker machine. The variation introduced is that after every gambler has selected a handle they must share their winnings with everyone else who chose their particular handle. For example if there are two handles each having expected payouts of \$25 and \$75 and 100 gamblers all pull the better handle they would each receive $\$75/100 = \$0.75$. If however the population divides across both handles proportionate to the expected payout of those handles the expected payout per gambler will be $\$75/75 = \$25/25 = \$1.00$. From this simple experiment it was shown that modifying the payout function in the k-armed bandit problem can introduce a reward for niche formation.

2.4 Advantages and Disadvantages of Niching

Most search algorithms are designed (or were initially designed) to find a single optimal solution to a difficult problem. This design trait is borne of the problems that exist in the literature where the goal tends to be optimisation of a single objective. Niching algorithms tend to be best applied in situations where multiple optimal solutions are required rather than a single optimum, such as in multi-modal and multiple objective problems.

Alternatively, niching algorithms can also be applied in situations where a single optimum solution is desired. This may be because some search advantage is gained over non-niching algorithms due to the specific search landscape of the problem. Niching algorithms in this sense may permit a more effective use of available resources by a search algorithm by either implicitly or explicitly dividing and searching different areas of the search space in parallel [31]. Such automatic resource re-allocation is useful in preserving useful population diversity for an extended search time.

Niching can also be advantageous if applied in a dynamic optimisation problem. In Schoeman and Engelbrecht [39] a PSO algorithm was adapted with a niching operator for application to a series of dynamic multi-modal

[1] Derate, a commonly used term in niching literature means 'to decrease the fitness of'.

function optimisation problems. The results showed that with moderate changes to the problem under study the algorithm was able to track peaks by maintaining stable niches around them. The algorithm was able to perform good resource reallocation as the peaks were removed, which is good for the long-term applicability of such a technique.

Since niching techniques spread the population across a wider search area, they can often waste computation by continually searching areas of the search space where no interesting optima exist. Also recombination of solution components from different niches can lead to the introduction of 'lethals' [8] which are solutions created between two optima but not located on an optima themselves. Some niching techniques introduce extra parameters in addition to the base algorithm. These parameters often come without good heuristics to set them and thus require extra sensitivity analysis to ensure best performance [43].

3 Niching for Ant Colony Optimisation

To be able to easily implement standard niching methods such as Crowding and Fitness Sharing with ACO, the ability to readily measure the distance (or difference) between solutions is required. Since most standard ACO algorithms encode solution quality information into a pheromone matrix without storing the actual solutions, access to required distance information between multiple generations of solutions is impractical. While it may be possible to achieve a niche-like behaviour in ACO, it is much more straightforward to use Population-based ACO to implement standard niching techniques such as Crowding and Fitness Sharing, since in PACO, access to a multi-generational population in the traditional EC sense is guaranteed. In this section an introduction to ACO and Population-based ACO is offered and two previously introduced Population-based ACO algorithms (Crowding PACO and Fitness Sharing PACO) imbued with niching behaviour [1] are discussed.

3.1 Ant Colony Optimisation

Ant Colony Optimisation (ACO) [12, 15], is an optimisation methodology based on the foraging behaviour of Argentine ants. All ACO algorithms are responsible for the scheduling of three processes:

- Ants generation & activity
- Pheromone trail evaporation
- Daemon actions

Each of the processes listed allow for flexibility in their implementation, as a result many different ACO algorithms have been proposed in recent years. Notable examples include Ant Systems [16], Ant Colony Systems [14] and $\mathcal{MAX} - \mathcal{MIN}$ Ant Systems [41]. A visual representation of the process organisation of ACO is provided as Fig. 1.

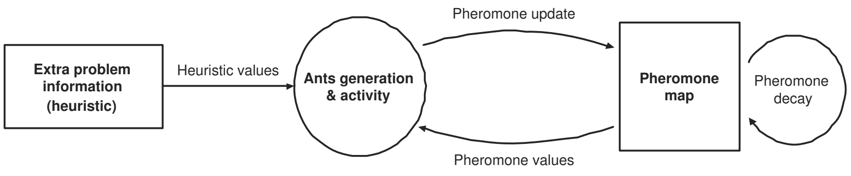


Fig. 1: Process organisation of the Ant Colony Optimisation Metaheuristic Framework

3.1.1 Pheromone Mapping

The pheromone mapping is the means by which solution components are able to be ranked and selected based on past usefulness. The pheromone mapping connects pheromone values from a pheromone map (usually a matrix structure) to specific solution components. The assumption usually being that if a prior solution is good then at least some of its parts (solution components) should also be good, and therefore a remixing of these components with other good components may lead to an optimal or near-optimal solution. A first step in defining an ACO algorithm is to define this pheromone mapping.

The problem domain will dictate how the pheromone mapping should be defined. In applying an ACO algorithm to a combinatorial optimisation problem such as the travelling salesman problem (TSP) it is not of interest which specific components are included, as any feasible solution will include every city once (and only once), it is the order of these components which is important in finding an optimal solution. For the TSP the transition points (edges/arcs) between the specific components can be assigned a specific pheromone value to reflect which order of cities is best. If a solution includes an edge connecting city a to city b and this solution is good then this goodness should be reflected by a higher pheromone level on this specific edge and other edges included in this solution.

3.1.2 Ants Generation and Activity

This process is responsible for the creation of new candidate solutions to the optimisation problem being addressed by the ACO algorithm, as well as the updating of pheromone values via the pheromone mapping. Through the execution of this process pheromone information is updated (increased or decreased depending on the implementation). A temporary population of (artificial) ants is used to construct feasible solutions to the problem being addressed. Each ant is evaluated upon the completion of a feasible solution and the solution information encoded through the pheromone mapping. After this encoding each individual ant is discarded and a new 'empty' ant is created in its place. This process is repeated until some stopping criterion is met.

An ant has the following properties:

- An ant searches for a minimum (or maximum) cost solution to the optimisation problem being addressed.
- Each ant has a memory used to store all solution components used to date, so that the candidate solution can be evaluated at the completion of solution construction; the memory can be used as a tabu list such as in the case of the TSP so that no component is reused.
- An ant can be assigned a starting position, for example an initial city in a TSP.
- An ant can include any feasible solution component (an example of a feasible solution component in a TSP would be a city which has not already been included in the candidate solution) until such time that no feasible components exist or a termination criterion is met (usually correlating to the completion of a candidate solution).
- Ants include solution components according to a combination of a pheromone value and a heuristic value which are associated with every solution component in the problem, the choice of which solution component is usually a probabilistic one.
- When including a new solution component in the growing candidate solution the pheromone value associated with the transition between these components (arc/edge in a TSP), or the solution component itself can be altered (*online step-by-step pheromone update*).
- An ant can retrace a candidate solution at the completion of a solution, updating the pheromone values of all transitions and/or solution components used in the solution (*online delayed pheromone update*).
- Once a candidate solution is created, and after completing online delayed pheromone update (if required) an ant dies, freeing all allocated resources.

3.1.3 Pheromone Trail Evaporation

Like the biological ant colony, the artificial ant colony employs a pheromone evaporation mechanism. This mechanism serves as a useful way of 'forgetting' older search bias [13]. As ACO algorithms use positive reinforcement, if pheromone was allowed to accumulate without decay the system would very quickly converge on a single solution since this solution would continue to be reinforced. This evaporation process can be thought of as a global pheromone update (it decreases all pheromone values by a set percentage), different to the previously mentioned local pheromone update process that increases or decreases specific pheromone values.

3.1.4 Daemon Actions

Daemon actions can be used to perform specialised functions which often require more knowledge than an individual ant is allowed [13, 17]. For example, a daemon action could inspect all solutions generated in one search cycle, identify the best solution and increment the pheromone values of its solution components more than the regular pheromone update (*offline pheromone update*). An alternative daemon action could be the application of a local search procedure.

3.2 Population-Based Ant Colony Optimisation

The Population-based Ant Colony Optimisation (PACO) [23, 24] algorithm was first introduced as a single objective optimisation algorithm designed for dynamic optimisation problems. PACO was later extended for a multiple objective optimisation problem, the single machine total tardiness problem with changeover costs [22, 25]. The defining difference between PACO and the canonical ACO algorithm is in the area of solution storage. Whereas most traditional ACO algorithms (e.g., Ant Systems [16], Ant Colony Systems [14], $\mathcal{MAX}-\mathcal{MIN}$ Ant Systems [41]) store solution information from an (artificial) ant in a pheromone matrix only, PACO stores solutions in a population and then uses this population to make adjustments to the pheromone matrix. At any time the pheromone matrix will be a direct reflection of some or all of the stored population.

In the single objective PACO algorithm, as solutions enter the population a positive update on the pheromone matrix is performed, and as solutions leave the population a negative update is performed to adjust the pheromone matrix values. This process removes the requirement for a traditional ACO decay operation and results in a significant speed improvement over traditional pheromone maintenance operations [24]. PACO still uses the traditional ACS greedy transition rule [14] to construct new solutions. Figure 2, taken from Angus [4], provides a visual summary of the Population-based ACO algorithm using similar terminology to that defined in Dorigo et al. [17] and Cordón et al. [7], this can be contrasted with ACO as presented in Fig. 1.

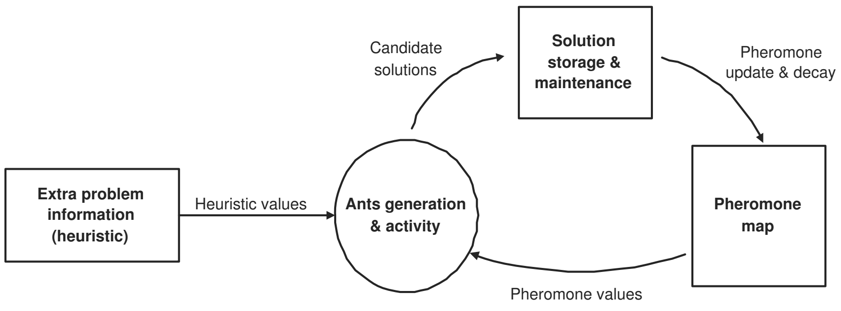

Candidate solutions
Solution storage & maintenance
Pheromone update & decay
Extra problem information (heuristic)
Heuristic values
Ants generation & activity
Pheromone map
Pheromone values

Fig. 2: Population-based ACO process organisation (reproduced from Angus [4])

3.3 Niching Ant Colony Optimisation Algorithms

As mentioned, at time of writing two examples of Niching ACO algorithms can be found in the literature. These algorithms both extend the PACO algorithm and are presented in this section.

3.3.1 Crowding PACO

In ACO algorithms the pheromone matrix tends to reflect the experience of the entire population, and unlike the GA, PACO algorithms do not select parent solutions for new solution construction. Rather, new individuals gain experience via the pheromone matrix as they progress towards a solution. Given this, Restricted Tournament Selection [26] is a suitable Crowding model since it compares new solutions against a subset of the entire population. The size of the subset selected is denoted as the Crowding window size, thus the application of this form of niching introduces one extra parameter to the basic PACO algorithm. The Crowding window size can be set anywhere between one and the size of the population. A general pseudocode representation of Crowding PACO (CPACO) is outlined in Alg. 1.

3.3.2 Fitness Sharing PACO

To implement Fitness Sharing with Population-based ACO a temporary pheromone matrix scheme is used. Rather than simply adding the best new solution and removing the oldest solution as in PACO, all solutions from a generation are added to the population, and the oldest population members are removed. After this addition/removal process is complete, the Fitness

Algorithm 1 Crowding Population-based Ant Colony Optimisation: CPACO

```
1: Uniformly initialise pheromone map values to $\tau_{init}$
2: for $j = 1$ to $h$ do
3:     Create and evaluate random solution
4:     Insert random solution into history
5:     Add random solution information into pheromone map $(+\Delta\tau)$
6: end for
7: while stopping criterion not met do
8:     Construct $m$ new solutions $(s^{new})$
9:     Evaluate solutions
10:    Crowding history update
11: end while
12: procedure CROWDING HISTORY UPDATE
13:    for $j = 1$ to $m$ do
14:        Select random subset (size=$c$) of solutions ($s$) from population ($p$)
15:        for $k = 1$ to $c$ do
16:            $d =$ distance $\left(s_j^{new}, s_k\right)$
17:            if $d <$ leastDistance then
18:                leastDistance $= d$
19:                $s_{\text{closest}} = s_k$
20:            end if
21:        end for
22:        if $s_j^{new}$.quality $> s_{\text{closest}}$.quality then
23:            Remove $s_{\text{closest}}$ information from pheromone map $(-\Delta\tau)$
24:            Remove $s_{\text{closest}}$ from population
25:            Add $s_j^{new}$ to population
26:            Add $s_j^{new}$ information into pheromone map $(+\Delta\tau)$
27:        end if
28:    end for
29: end procedure
```

Sharing function is applied to all solutions in the current population to derate their respective fitness values. These solutions are then used to construct a pheromone matrix which is used to create the next generation of solutions using the standard ACS pseudo-random proportional rule. A general pseudocode representation is provided as Alg. 2. In this pseudocode example p represents the population, and p_i the i^{th} member of that population.

3.4 Alternatives to Niching

As indicated by Horn [28] it is prudent to discuss alternative diversity preservation mechanisms alongside niching, as niching is itself a form of diversity preservation. Focusing on ACO algorithms, the issue of diversification versus intensification has been a driving force behind the development of ACO and

Algorithm 2 Fitness Sharing Population-based Ant Colony Optimisation: FSPACO

```
 1: while stopping criterion not met do
 2:     Construct temporary pheromone matrix
 3:     Construct Solutions
 4:     Update history (Replace oldest solutions)
 5:     De-rate quality
 6: end while
 7: procedure DE-RATE QUALITY
 8:     for $j = 1$ to $p_{\text{size}}$ do
 9:         nicheCount = 0
10:         for $k = 1$ to $p_{\text{size}}$ do
11:             $d = \text{distance}\,(p_j, p_k)$
12:             if $d < \sigma$ then
13:                 shareValue = $(1 - (d/\sigma)^{\alpha})$
14:             else
15:                 shareValue = 0
16:             end if
17:             nicheCount = nicheCount + shareValue
18:         end for
19:         $h_j$.quality = $h_j$.quality/nicheCount
20:     end for
21: end procedure
```

the balance between these factors is often cited as one of the distinguishing features of different ACO algorithms. This is particularly evident in algorithms such as Ant Colony Systems (ACS) [14] that use elite solutions to promote intensification and localised decay to ensure diversification.

Some research specifically targets the issue of diversification by adding features to basic ACO algorithms which introduce randomness [32, 19, 34, 35] dependent on different criteria such as the measured diversity of the population. For the problems addressed those techniques allow the modified algorithms to overcome some of the issues associated with premature convergence to suboptimal solutions.

While these cited approaches show improvement over the basic ACO algorithms, they are very different to niching. These modifications usually delay convergence to a single area of the search space. While niching aims at increasing diversity, it does not hold off convergence or prevent it, nor does it purposely introduce extra randomness. Niching aims at creating and maintaining stable subpopulations. This is quite different to slowing convergence or introducing randomness since niching algorithms can be tuned to be highly convergent, but convergent to multiple areas of the search space.

4 Applications of Niching Ant Colony Optimisation

To date only a handful of applications of niching ACO algorithms can be found in the literature. In this section three example applications are reproduced from the literature. The discussion focuses on highlighting the formation of niche behaviour and its effect on algorithm performance, including quality of solutions obtained and algorithm complexity.

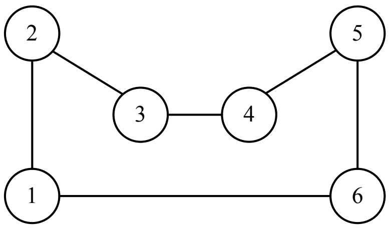
4.1 Travelling Salesman Problem

At the outset applying a niching algorithm to the TSP does not seem to be a good idea since it has been shown that the TSP seems to benefit from greedy search behaviour, focused on one area of the search space at a time [42]. In their work on $\mathcal{MAX} - \mathcal{MIN}$ Ant System (MMAS), Stützle and Hoos [42] comment that Ant Systems (AS) performs poorly on the TSP in comparison to later ACO algorithms because AS does not exploit good solutions strongly enough. This is true since most of the best performing ACO algorithms for the TSP include variations such as greedy transition rules or pheromone update rules that strongly bias the reinforcement of elite solutions. Since elitism tends to correlate strongly with an increase in search efficacy this may indicate something about the problem: that the n-best solutions to a TSP all contain similar elements and thus are located in a similar area of the search space. In Angus [4] the best 100 solutions for the Burma14 problem were shown to occupy a similar space in the overall search space having on average 9 similar edges (out of 14) to all of the other best solutions. A baseline random sample indicated that from a random sample of 100 solutions only 2 out of 14 edges are the same on average when selecting from across the entire search space.

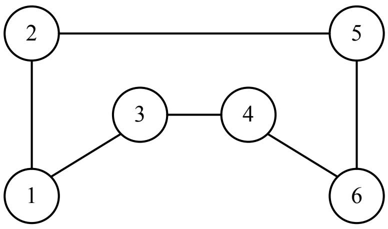
Since niching algorithms strive to maintain diversity, it is intuitive that niching algorithms are not suited to solving problems such as the single-objective TSP that benefit from strong convergent behaviour. However, cases may exist where a niching algorithm would be suited to solving the TSP. In Angus [1] a fabricated TSP, the Crown6 TSP, with two spatially separated optima was constructed. The Crown problem is a symmetric, 2-Dimensional Euclidean TSP containing 6 vertices which has the interesting property of containing two distinct yet equal global optima (Fig. 3). These optima are also a reasonable distance apart only sharing 3 out of 6 edges.

In Angus [2] the MMAS algorithm was tested[2] using 50 ants per iteration, and the number of times either of the two optima were found per iteration was recorded. This experiment was repeated 100 times for consistency of reported results. In every algorithm trial MMAS converged to one of the two

[2] Using the standard parameters as in Stützle and Hoos [41].



(a) A - 1,2,3,4,5,6 length = 541



(b) B - 1,2,5,6,4,3 length = 541

Fig. 3: Optimal solutions to Crown problem

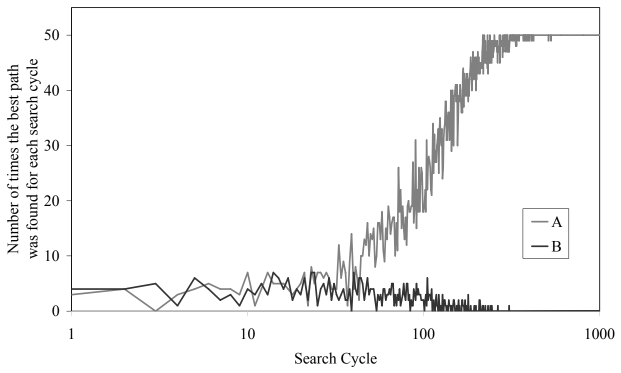
optima, while the niching PACO algorithms were shown to converge to both optima. Three graphs indicating single experimental runs (that are indicative of normal algorithm behaviour) of each of the algorithms are reproduced in Fig. 4a, Fig. 4b and Fig. 4c.

This Crown problem was a trivially small although quite unique TSP, with regard to the presence of two distinct optima. For completeness the niching PACO algorithms were also tested on several small-medium sized TSP from TSPLIB [36] to observe the effect of niching on the algorithm performance (with regard to locating the optimal solution). The results obtained, while not terrible were not as good as the MMAS algorithm with regard to the best solution found. This result was to be expected due to the reasons provided earlier.

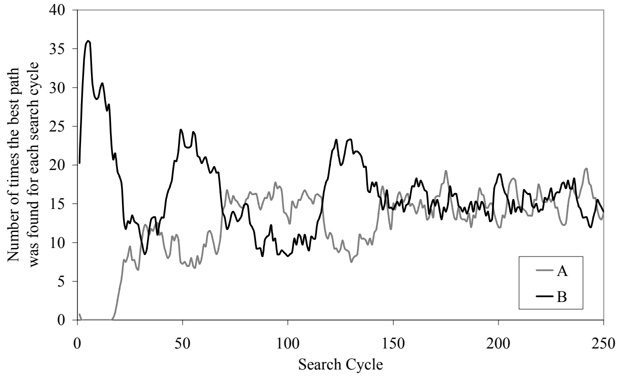
The overall findings from this study were that while the standard TSP instances of TSPLIB did not benefit from niching, the special case Crown problem did. While this problem was entirely contrived it did demonstrate the niching algorithms' strength in the simultaneous location and maintenance of multiple optima on a combinatorial problem domain. For the overwhelming majority of single objective TSP in TSPLIB [36] this property is probably not required, however there do exist other variations to the basic TSP that require the location and maintenance of diverse solutions, one such variation being the multiple objective TSP reported in Sec. 4.3.

4.2 Multimodal Function Optimisation

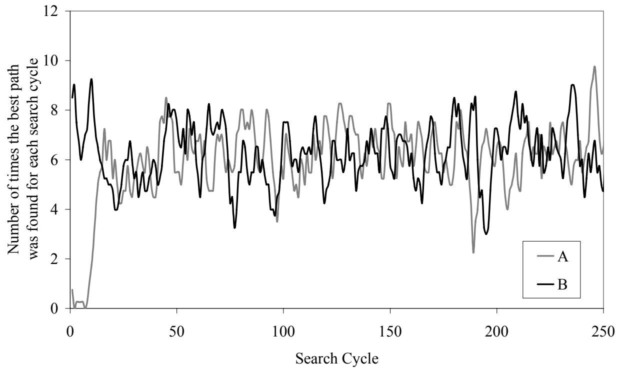
Multimodal function optimisation is a problem type that is useful in analysing niching behaviour. These problems have been used in many prior niching algorithm investigations mostly due to the ability to readily design challenging benchmark problems with features such as multiple diverse optima. The problems are also easily scaled to multiple dimensions which can rapidly increase the computational complexity, however such scaling does not necessarily come at a cost to the ability to analyse and comment on algorithms



(a) MMAS



(b) FSPACO



(c) CPACO

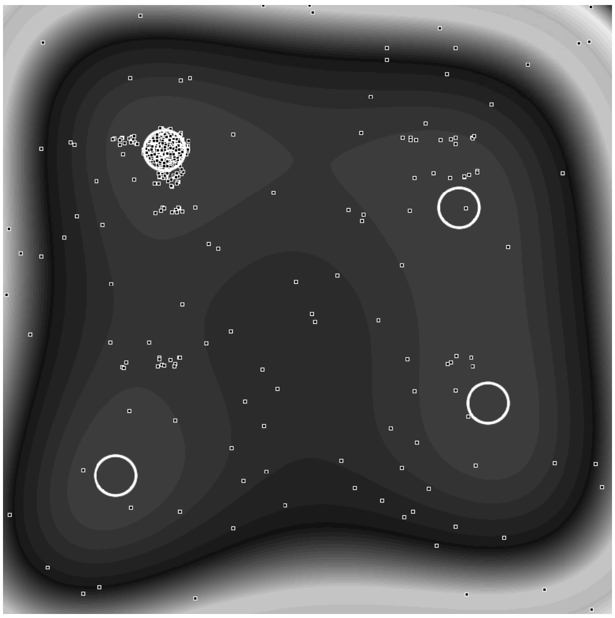
Fig. 4: Crown problem: Occurrence of the two optimal paths over time for a single (although indicative) experiment run (A & B are the two distinct optimal paths). Each algorithm tested constructs 50 solutions per iteration. Different running times are reported to better illustrate specific algorithm behaviour, importantly though the convergence characteristics do not change past the maximum number of iterations reported on the graphs.

applied to them. This is because these problems allow for easy visualisation of their search space, while still preserving their neighbourhood relationships.

The domain of function optimisation is somewhat different to traditional combinatorial optimisation problems like the TSP. Variations of ACO algorithms have been applied to this domain with success [40]. In Angus [1] the Population-based ACO algorithm was modified for the function optimisation problem and was imbued with each of the fitness sharing and crowding niching techniques. Primarily this investigation was a qualitative one, aimed at observing sustained niche formation for a prolonged period of time. The Crowding PACO algorithm tested was successful in performing this task, while the Fitness Sharing PACO algorithm results were less impressive, attributed to high sensitivity in the parameter selection. Several figures indicating the distribution of all solutions for an entire algorithm run are reproduced from Angus [1] as Fig. 5. These figures indicate the fitness of the function as shaded, the dots represent all evaluated solutions (across multiple generations) for the entire algorithm run. In this example the algorithms tested were allowed 50,000 solution evaluations and each used a population size of 100, meaning that they were iterated for a total of 500 generations. As can be seen from these figures the sampling behaviour of the niching algorithms are spread across all peaks of interest, which is the expected behaviour of a niching algorithm. As a control the ACO algorithm without niching was found to converge and thus sample only one distinct peak.

A similar experiment was performed in Angus [4] to evaluate the effect of crowding window size on the stability of niche formation and maintenance. The problem used was the Shekel's foxholes problem which is comprised of 25 distinct optima of varying height. An optimal solution with a niching algorithm would be the distribution of a population across all 25 optima. In the study it was shown that decreasing the window size increased the frequency of replacement errors which lead to the loss of some of the optima. At an extreme this behaviour led to the loss of all but one optima. The measure used to analyse this behaviour was the max-peak ratio which is a sum of the relative distances from each optima to the nearest population member. Figure 6 indicates the effect of decreasing the crowding window size of CPACO for the Shekel's foxholes problem in two dimensions.

In general, results from Angus [1, 4] suggest that the Niching PACO algorithms were effective at solving a range of multi-modal function optimisation problems. These studies also compared the Niching PACO algorithms against state-of-the-art Niching EC algorithms and found that the results were comparable.


(a) Control (PACO)

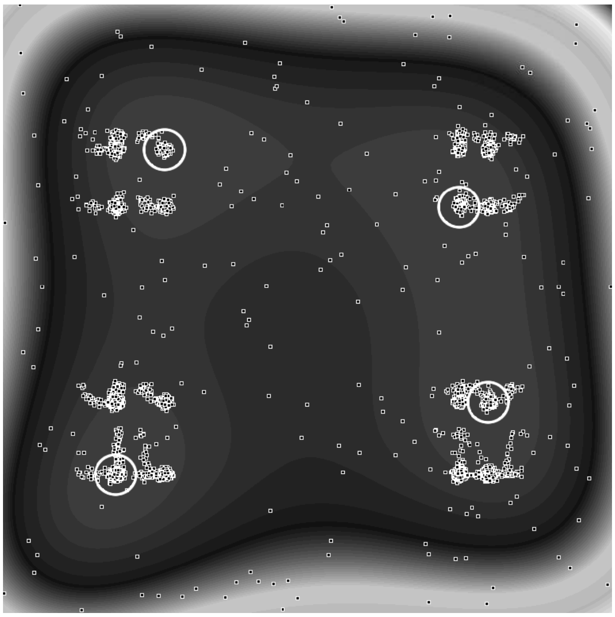
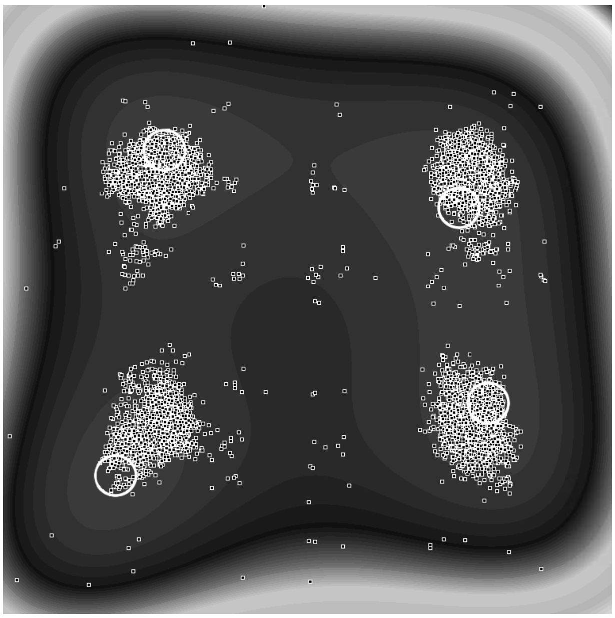

(b) CPACO

Fig. 5: Diagrams indicating all of the 50,000 solutions generated in a single run (indicative of the usual search behaviour) by each algorithm applied to Himmelblau's Function (white circles indicate the four distinct optima, small white squares represent solutions).



(c) FSPACO

Fig. 5: (cont'd) Diagrams indicating all of the 50,000 solutions generated in a single run (indicative of the usual search behaviour) by each algorithm applied to Himmelblau's Function (white circles indicate the four distinct optima, small white squares represent solutions).

4.3 Multiple Objective Travelling Salesman Problem

Multiple Objective Optimisation (MOO) problems involve the simultaneous optimisation of two or more objective functions and most classic single objective optimisation problems have MOO variants. These problems often require a different approach to that of single objective optimisation due to decision makers often requiring multiple Pareto optimal, or near-Pareto optimal solutions; these Pareto optimal solutions being the best trade-off solutions. Given that more than one solution is required these problems are often solved using niching algorithms.

In Angus [2] the CPACO algorithm was extended for a suite of multiple objective travelling salesman problems (MOTSPs). The resulting algorithm, Multiple Objective CPACO (MO-CPACO) uses the key CPACO features of crowding replacement and probabilistic step-wise solution construction. To construct solutions MO-CPACO uses a weighted combination of heuristic information from all objective functions, and pheromone information which is based on the current state of the population. Due to there being more

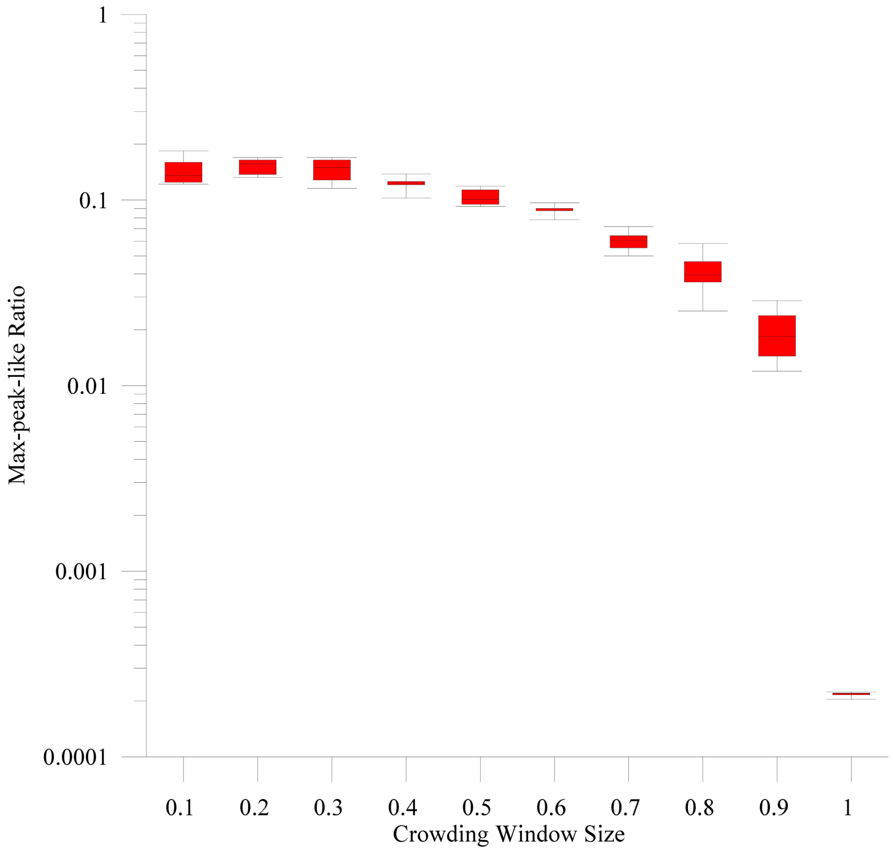


Fig. 6: Multiple box plots indicating the effect on the max-peak performance metric when the crowding window size of CPACO is varied from 0.1 to 1.0. For this max-peak performance metric zero indicates best performance while one indicates the poorest performance. Each parameter setting was repeated 100 times and allowed 50,000 solutions evaluations per individual run.

than one objective function the amount of pheromone deposited by each ant into the pheromone matrix is determined by a Non-Dominated Sorting technique [10] which ranks solutions according to their proximity to the Pareto front. This means that there is one heuristic matrix per objective and one pheromone matrix overall. All solutions are evaluated by each objective function after being created. These candidate solutions are then compared against a subset of the population. The closest solution (in terms of objective space distance) is replaced if the candidate solution is better in all objectives (crowding replacement). More specific algorithm details can be found in Angus [2].

In Angus [2] the MO-CPACO algorithm was tested on several multiple objective TSPs. Among the problems tested were the two objective KroAB100, KroAB150 and KroAB200 problems, each with 100, 150 and 200 cities respectively. These problems are taken from the TSPLIB [36]. As a comparison an original Multiple Objective ACO algorithm without niching was also tested with an equal number of solution evaluations and the same population size. Summary attainment surface comparison was used to evaluate any differences and these differences were evaluated using non-parametric statistical techniques as the data were non-normally distributed. These summary attainment surfaces show the best solutions obtained averaged over 100 independent trials. Given that every solution in the final population contains two objective values these solutions are plotted against both objectives simultaneously so as to easily visualise the trade-off between each objective. Like many multiple-objective problems these MOTSP don't have a single optimal solution, instead they contain several Pareto-optimal solutions.

Figures 7, 8 and 9, reproduced from the original investigation indicates that the MO-CPACO algorithm was able to obtain a better attainment surface than its non-niching equivalent. This finding suggests that the addition of niching to this ACO algorithm for these problems led to greater search efficacy. Such conclusions as to the usefulness of niching in multiple objective algorithm design are also commented on in Deb [9].

4.4 Key Findings

The problems listed in this section were so chosen for the original studies since they are applications which best demonstrate the advantages and disadvantages of adding niching to an ACO algorithm. The empirical results of these studies indicate that the use of niching benefited some problem domains while offering no substantial advantage to others. Some key findings are included here.

For the single objective TSP tested in Angus [1] there seemed to be a marked decrease in obtained solution quality. This was explained through a study of the neighbourhood relationship of an indicative TSP problem instance which highlighted that the TSP may offer a natural advantage to algorithms that tend to direct search effort in one area of the search space. A simple TSP problem (Crown6) was constructed to demonstrate that Niching ACO algorithms could exhibit niche formation on the TSP, but ultimately for this problem domain there seems to be no advantage in the application of these niching techniques.

Unlike the single objective TSP, multi-modal function optimisation problems do contain multiple spatially separated optima. For this particular problem domain Niching ACO algorithms were shown to be able to maintain stable niches [1, 4]. In these studies the Crowding PACO algorithm exhibited

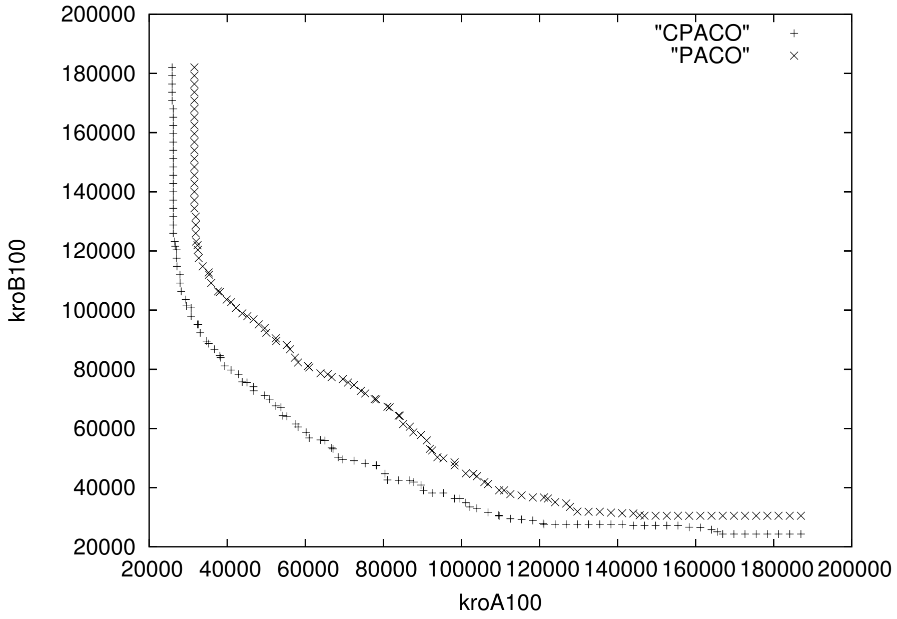


Fig. 7: 1% (best) attainment surface for kroA100 and kroB100 using MO-PACO (non-niching) & MO-CPACO (niching)

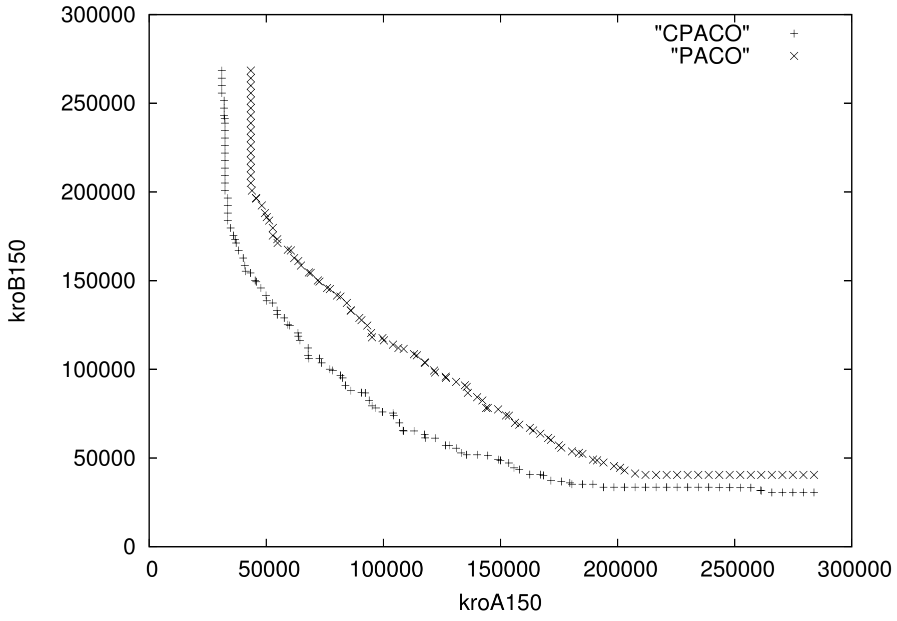


Fig. 8: 1% (best) attainment surface for kroA150 and kroB150 using MO-PACO (non-niching) & MO-CPACO (niching)

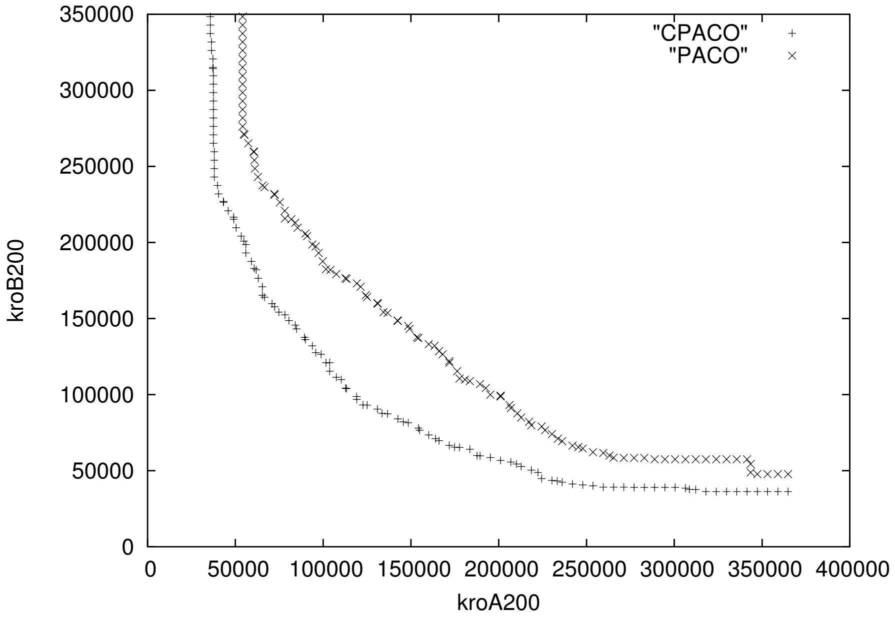


Fig. 9: 1% (best) attainment surface for kroA200 and kroB200 using MO-PACO (non-niching) & MO-CPACO (niching)

the best performance, while the Fitness Sharing PACO algorithm did not seem to perform as well, probably due to the parameter selection.

While the single objective TSP gave poor results for the Niching ACO algorithms, the application of the Crowding PACO algorithm to the multiple objective TSP [2, 4] saw a marked increase in algorithm efficacy and computational efficiency. For the problems tested, the Crowding PACO algorithm was able to maintain a very good distribution of solutions through the use of a niching population replacement operation.

5 Conclusion

This chapter has discussed current research into the application of niching to ACO algorithms. Important findings were the difficulty of applying basic niching techniques such as Crowding and Fitness Sharing to standard ACO algorithms. This difficultly was overcome through the use of the Population-based ACO algorithm paradigm. Once applied the niching algorithms were shown to be effective at sustaining multiple sub-populations distributed across points of interest in a search space. The key findings suggest that like many niching EA these niching ACO algorithms are best applied to multi-modal or multiple objective problem domains.

Acknowledgements The author acknowledges time spent at Swinburne University of Technology preparing the research outlined in this publication. Specifically the assistance provided by Prof. Tim Hendtlass and other members of the Complex Intelligent Systems Lab, Swinburne University of Technology.

References

[1] Angus, D.: Niching for Population-based Ant Colony Optimization. In: 2nd International IEEE Conference on e-Science and Grid Computing, Workshop on Biologically-inspired Optimisation Methods for Parallel and Distributed Architectures: Algorithms, Systems and Applications (2006)

[2] Angus, D.: Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem. In: 2007 IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making (MCDM 2007), pp. 333–340. IEEE, Piscataway (2007)

[3] Angus, D.: Population-based ant colony optimisation for multi-objective function optimisation. In: Randall, M., Abbass, H.A., Wiles, J. (eds.) ACAL 2007. LNCS, vol. 4828, pp. 232–244. Springer, Heidelberg (2007)

[4] Angus, D.: Niching ant colony optimisation. PhD thesis, Swinburne University of Technology (2008)

[5] Brits, R.: Niching strategies for particle swarm optimization. Master's thesis, Department of Computer Science, University of Pretoria, South Africa (2002)

[6] Brits, R., Engelbrecht, A.P., van den Bergh, F.: Scalability of niche PSO. In: Proceedings of the 2003 IEEE Swarm Intelligence Symposium (SIS 2003), pp. 228–234 (2003)

[7] Cordón, O., Herrera, F., Stützle, T.: A review of the ant colony optimization metaheuristic: Basis, models and new trends. Mathware & Soft Computing 9(2,3) (2002)

[8] Deb, K., Spears, W.M.: C6.2: Speciation methods. In: Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.) Handbook of Evolutionary Computation. Institute of Physics Publishing (1997)

[9] Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. KanGAL report 200001, Indian Institute of Technology, Kanpur, India (2000)

[10] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000)

[11] DeJong, K.A.: An analysis of the behaviour of a class of genetic adaptive systems. PhD thesis, University of Michigan (1975)

[12] Dorigo, M.: Optimization, learning and natural algorithms. PhD thesis, Politechico di Milano, Italy (1992)

[13] Dorigo, M., Di Caro, G.: The ant colony optimization meta-heuristic. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimisation, pp. 11–32. McGraw-Hill, London (1999)

[14] Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computing 1(1), 53–66 (1997)

[15] Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press, Cambridge (2004)

[16] Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man and Cybernetics, Part B 26(1), 29–41 (1996)

[17] Dorigo, M., Bonabeau, E., Theraulaz, G.: Ant algorithms and stigmergy. Future Generation Computer Systems 16, 851–871 (2000)

[18] Eldredge, N.: Macroevolutionary Dynamics: Species, Niches and Adaptive Peaks. McGraw-Hill, New York (1989)

[19] Gambardella, L.M., Taillard, E., Agazzi, G.: MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. Tech. rep., IDSIA (1999)

[20] Goldberg, D.E.: Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley, Reading (1989)

[21] Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the Second International Conference on Genetic Algorithms, pp. 41–49 (1987)

[22] Guntsch, M.: Ant algorithms in stochastic and multi-criteria environments. PhD thesis, Universität Fridericiana zu Karlsruhe (2004)

[23] Guntsch, M., Middendorf, M.: Applying population based ACO to dynamic optimization problems. In: ANTS 2002: Proceedings of the Third International Workshop on Ant Algorithms, pp. 111–122. Springer, London (2002)

[24] Guntsch, M., Middendorf, M.: A population based approach for ACO. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R. (eds.) EvoIASP 2002, EvoWorkshops 2002, EvoSTIM 2002, EvoCOP 2002, and EvoPlan 2002. LNCS, vol. 2279, pp. 72–81. Springer, Heidelberg (2002)

[25] Guntsch, M., Middendorf, M.: Solving multi-criteria optimization problems with population-based ACO. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 464–478. Springer, Heidelberg (2003)

[26] Harik, G.R.: Finding multimodal solutions using restricted tournament selection. In: Eshelman, L. (ed.) Proceedings of the Sixth International Conference on Genetic Algorithms, pp. 24–31. Morgan Kaufmann, San Francisco (1995)

[27] Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introduction With Applications to Biology, Control, and Artificial Intelligence. MIT Press, Cambridge (1975)

[28] Horn, J.: The nature of niching: Genetic algorithms and the evolution of optimal, cooperative populations. PhD thesis, University of Illinois (1997)

[29] Mahfoud, S.W.: Crowding and preselection revisited. In: Männer, R., Manderick, B. (eds.) Parallel Problem Solving from Nature 2 (PPSN2), pp. 27–36. North-Holland, Amsterdam (1992)

[30] Mahfoud, S.W.: Niching methods for genetic algorithms. PhD thesis, University of Illinois (1995)

[31] Mahfoud, S.W.: Niching methods. In: Back, T., Fogel, D.B., Michalewicz, Z. (eds.) Evolutionary Computation 2: Advanced Algorithms and Operators, pp. 87–92. Institute of Physics Publishing, UK (2000)
[32] Nakamichi, Y., Arita, T.: Diversity control in ant colony optimization. Artificial Life and Robotics 7(4), 198–204 (2004)
[33] Petrowski, A.: A clearing procedure as a niching method for genetic algorithms. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 798–803. IEEE, Los Alamitos (1996)
[34] Randall, M.: Maintaining explicit diversity within individual ant colonies. In: Recent Advances in Artificial Life, ch. 17. World Scientific, Singapore (2005)
[35] Randall, M., Tonkes, E.: Intensification and diversification strategies in ant colony system. Complexity International 9 (2002)
[36] Reinelt, G.: Tsplib95 (1995), http://www.iwr.uni-heidelberg.de/groups/comopt/software/tsplib95
[37] Ricklefs, R.E.: Ecology. Thomas Nelson & Sons Ltd. (1973)
[38] Robbins, H.: Some aspects of the sequential design of experiments. Bulletin of the American Mathematical Society 55, 527–535 (1952)
[39] Schoeman, I., Engelbrecht, A.: Niching for dynamic environments using particle swarm optimization. In: Wang, T.-D., Li, X.-D., Chen, S.-H., Wang, X., Abbass, H.A., Iba, H., Chen, G.-L., Yao, X. (eds.) SEAL 2006. LNCS, vol. 4247, pp. 134–141. Springer, Heidelberg (2006)
[40] Socha, K.: ACO for continuous and mixed-variable optimization. In: Dorigo, M., Birattari, M., Blum, C., Gambardella, L.M., Mondada, F., Stützle, T. (eds.) ANTS 2004. LNCS, vol. 3172, pp. 25–36. Springer, Heidelberg (2004)
[41] Stützle, T., Hoos, H.: Improvements on the Ant System: Introducing the $\mathcal{MAX}-\mathcal{MIN}$ Ant System. In: Third International Conference on Artificial Neural Networks and Genetic Algorithms. Springer, Norwich (1997)
[42] Stützle, T., Hoos, H.: $\mathcal{MAX}-\mathcal{MIN}$ Ant System. Future Generation Computer Systems 16(8), 889–914 (2000)
[43] Watson, J.P.: A performance assessment of modern niching methods for parameter optimization problems. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.) Genetic and Evolutionary Computation Conference, vol. 1, pp. 702–709. Morgan Kaufmann, Orlando (1999)

Using Ant Colony Optimisation to Construct Meander-Line RFID Antennas

Andrew Lewis[1], Marcus Randall[2], Amir Galehdar[3], David Thiel[3], and Gerhard Weis[1]

Abstract A method increasingly used to uniquely identify objects (be they pieces of luggage, transported goods or inventory items in shops and warehouses), is Radio Frequency IDentification (RFID). One of the most important components of RFID systems is the antenna and its design is critical to the utility of such tracking systems. Design engineers have traditionally constructed small antennas using their knowledge and intuition, as there is no simple analytical solution relating antenna structure to performance. This, however, does not guarantee optimal results, particularly for larger, more complex antennas. The problem is ideally suited to automated methods of optimisation. This chapter presents an overview of the automatic design of antennas using the meta-heuristic known as Ant Colony Optimisation (ACO). Apart from a description of the necessary mechanics ACO needs to effectively solve this problem, a novel local search refinement operator and a multi-objective version of the problem are also described. The latter is used to optimise both antenna efficiency and resonant frequency. Computational results for a range of antenna sizes show that ACO is a very effective design tool for RFID antennas.

Institute for Integrated and Intelligent Systems
Griffith University
Queensland, Australia
`{a.lewis@, gerhard.weis@student}griffith.edu.au`
School of Information Technology
Bond University
Queensland, Australia
`mrandall@bond.edu.au`
Centre for Wireless Monitoring and Applications
Griffith University
Queensland, Australia
`{s2145033@student.,d.thiel@}griffith.edu.au`

1 Introduction

The automatic identification of items is a real-world problem that appears in a diverse range of applications, including, but not limited to, luggage tagging, warehouse inventory tracking, vehicle identification on user-pay motorways, and potentially products sold in shops (such as supermarkets). One technology used for these purposes is radio frequency identification (RFID). In this approach a receiver attached to an item (known as a *tag*) is exposed to an electro-magnetic field that generates a response from which the item may be uniquely identified. One of the problems that engineers have encountered is the design of compact antenna structures for the tags since, for their practical application in tracking a wide range of different items, they are often very small (possibly less than half a centimetre squared). Until recently, engineers would manually derive antenna designs according to their intuition and knowledge of the field. This practice is time consuming and may not yield the best results but, since no analytical solutions are known to derive antenna performance from an arbitrary structure, some form of empirical approach to design has been necessary.

However, intelligent automated processes can be used in the design and evaluation of such antennas. Moreover, search strategies, based on biological paradigms, are able to efficiently and intelligently search large numbers of configurations. One such approach, ant colony optimisation (ACO) [5] is very good at solving similar problems to that of the design of RFID antennas. Recent work by Randall, Lewis, Galehdar and Thiel [14] has shown that it is indeed possible and practicable to allow an ACO meta-heuristic-based software system to design and evaluate such antennas.

This chapter presents an overview of the use of ACO in the design of RFID antenna structures, including a simple, initial application, the addition of a local refinement operator and the further augmentation of the model to handle multiple, conflicting design objectives by use of Pareto-dominance relations (Sections 4, 5 and 6 respectively).

2 RFID Antennas

The idea of radio frequency identification (RFID) was first developed in 1948 [18]. Recently there have been many enhancements to this basic idea and the concept has found applications in many areas. For practical reasons there is a need for smaller tags with longer "read range". The read range is "the maximum distance at which an RFID reader can detect the backscattered signal from the tag" [15]. This vital factor can be increased by designing antennas with higher gain and this is directly related to the antenna efficiency. Design engineers seek the smallest, most efficient antenna structures for their

RFID tags. Most antennas are screen-printed on thin plastic and so are 2D structures.

Thus, the design of efficient antennas is an emerging and important area [7, 8]. Often, RFID antennas are designed by constructing a dipole antenna in which each arm of the dipole is a "meander line", that is, the conducting element is folded upon itself in a convoluted, space-filling path, maximising the length of the element while minimising the area it occupies. A simple example is shown in Figure 1. One approach to achieve this is to lay out the antenna element on a Cartesian grid, constructing the meander line by joining points in the grid into a continuous path.

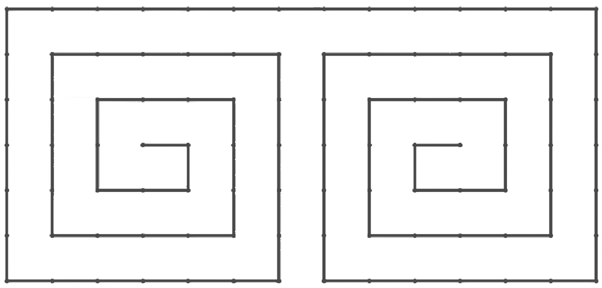


Fig. 1 A simple meander line antenna.

This process of constructing a meander line antenna has similarities to, but is not the same as, two well known problems. The first of these is the travelling salesman problem (TSP). Consider a set of cities, with known distances between each pair of cities. The aim of the TSP is to find the shortest path to traverse all cities exactly once and return to the starting city. The major difference between this and the problem in this paper, is that there is a potential connection between every pair of cities, whereas a meander line on the Cartesian grid may only advance to an adjacent (non-diagonal) grid point. Both TSP and a meander line of maximal length give rise to Hamiltonian walks.[1]

The second problem is that of producing self-avoiding walks (SAWs) [9, 12, 17]. This finds application in chemistry for which linear polymer molecules in a good solvent are required [17]. As the name suggests, a path is constructed that moves from one point to the next without crossing itself. While this sounds very similar to building meander lines, a SAW has an infinite grid size (as polymers exist in continuous space) whereas meander line RFID antennas are severely constrained in size for practical reasons.

There has been a large interest in solving TSP using the meta-heuristic ant colony optimisation as its mechanics lend themselves naturally to this

[1] It is evident the meander line problem is $\mathcal{NP}$ hard as it is reducible to TSP and vice-versa.

problem. Ants are agents that iteratively construct solutions and ideally suited to path planning problems (such as the TSP and vehicle routing problem variants). In this chapter, we outline the process and effectiveness for using ACO on the real-world problem of designing meander line antennas for RFID devices.

3 The Ant Colony System Algorithm

The original concept of ant colony optimisation of Dorigo [4] demonstrated that optimisation problems could be solved using a modelling of the foraging behaviour of Argentine ants. ACO is, in fact, a collection of meta-heuristic techniques, one of which is the ant colony system (ACS) [6]. This version has been shown to have good performance and is robust enough to be applied across a range of combinatorial optimisation problems, particularly path planning problems. It is used as the basis of our search algorithm.

ACS can best be described by its application to solving TSP as this is a well understood optimisation problem and, as previously shown, a close relation to the problem of constructing meander line antennas. Consider a TSP with N cities. Cities i and j are separated by distance $d(i, j)$. To ensure some level of solution diversity, m ants are placed randomly on these cities. m is usually a lot smaller than N; it is often arbitrarily set at ten (across a range of applications.) In discrete time steps, all ants select their next city and then simultaneously move to it. Like natural ants, they deposit a substance known as *pheromone* to communicate with the colony about the utility (goodness) of the edges they traverse. The pheromone on edge (i, j) is denoted by $\tau(i, j)$.

A combination of pheromone and problem specific heuristic information (conventionally denoted as a function of η) is the basis on which ants construct solutions. For each step of ACS, Equations 1 and 2 are used to select the next city to visit. In the equations this is referred to as s, the current city as r, and the ant as k. The first branch of Equation 1 is a greedy selection technique that will choose the city that has the best combination of short distance and large pheromone level. If subsequent ants always choose previously followed paths, the degree to which they explore the problem space will be restricted and stagnation of the search process in local minima will be highly likely. To avoid this there is a probability q that Equation 2 will be used to select the next city instead. This equation generates a weighted probability for all remaining cities. A roulette wheel selection function, R, is then used to choose s. These two equations collectively are called the pseudo-proportional rule.

$$s = \begin{cases} \arg\max_{u \in J_k(r)} \left\{ \tau(r,u)[\eta(r,u)]^{\beta} \right\} & \text{if } q \leq q_0 \\ \text{Equation 2} & \text{otherwise} \end{cases} \tag{1}$$

$$s = R\left(\begin{cases} \frac{\tau(r,s)[\eta(r,s)]^{\beta}}{\sum_{u \in J_k(r)} \tau(r,u)[\eta(r,u)]^{\beta}} & \text{if } s \in J_k(r) \\ 0 & \text{otherwise} \end{cases} \right) \tag{2}$$

Note that $\eta(r,u) = d(r,u)$ and q_0 is a parameter bounded between 0 and 1. Values of q_0 close to 1 correspond to a greedy search, whereas values close to 0 represent a random search. Each ant maintains a memory of its previous steps. In the case of both TSP and meander lines, ant k can only select available values from $J_k(r)$ where r is the current city or grid point respectively.

The parameter β governs the relative importance of the heuristic information η. In the case of the TSP, this value will be a negative one so that shorter edges are favoured. The use of the pheromone information biases the search towards selecting edges that are well traversed (i.e., have a high pheromone level).

As in natural ant systems, the pattern of pheromone values is constantly evolving to reflect the collective intelligence and memory of the colony. However, unlike real ants these changes do not occur continuously. Instead, two separate update phases are used. They are referred to as local and global pheromone updating. For the former, the pheromone level on the selected edge is updated according to Equation 3. This is done after all ants have completed a single step.

$$\tau(r,s) \leftarrow (1-\rho) \cdot \tau(r,s) + \rho \cdot \tau_0 \tag{3}$$

Where:

ρ is the local pheromone decay parameter, $0 < \rho < 1$ and
τ_0 is the initial amount of pheromone deposited on each of the edges. This value cannot be 0 (as the pheromone matrix could not change), but is usually set as a positive value close to 0.

Global updating of pheromone takes place once all ants have constructed a solution. Edges that compose the best solution (so far) are rewarded with an increase in their pheromone level while the pheromone on the other edges is evaporated (decreased). This is expressed in Equation 4.

$$\tau(r,s) \leftarrow (1-\gamma) \cdot \tau(r,s) + \gamma \cdot \Delta\tau(r,s) \tag{4}$$

$$\Delta\tau(r,s) = \begin{cases} \frac{1}{L} & \text{if } (r,s) \text{ is an edge within the best solution found during the} \\ & \text{entire search process} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Where:

$\Delta\tau(r,s)$ reinforces the pheromone on the edges of the best solution (see Equation 5),

L is the length of the best (shortest) tour to date and
γ is the global pheromone decay parameter, $0 < \gamma < 1$.

An in-depth pseudocode description of the ACS algorithm can be found in Dorigo and Gambardella [6].

4 Meander Line Antennas and ACS

As stated previously, a meander line antenna may be designed by laying out antenna elements on a finite grid of points. To form the line, each point can be connected to points directly horizontally and vertically above and below the point. This gives up to four neighbouring points, with boundary points naturally having fewer than this number. Each point must form part of the meander line. This forms half the antenna from which the dipole is constructed using a mirror image. Figure 2 shows a 5×5 grid as well as a feasible meander line half-antenna.

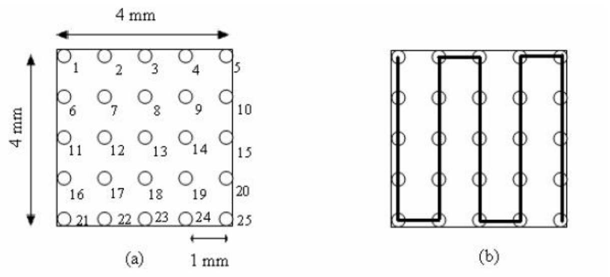


Fig. 2 Diagram (a) defines the grid and its numbering system on a 5×5 grid. (b) shows a feasible meander line half-antenna element. ©2007 IEEE

For the first design problem considered, that of a small antenna with maximal efficiency, the following design parameters were used. Let n be the number of grid points in one side of a square grid. By restricting the design to dipole antennas with identical, mirrored arms it is only necessary to construct one meander line arm, on approximately half the area of the total antenna. The track grid separation was set at 1 mm and the antenna half-area was set to 4×4 mm^2. Tracks were mapped to the $n \times n$ grid points (as shown in Figure 2 for $n = 5$). The dipole was centre-fed along the line of symmetry. By connecting all grid points, a meander line with an half length of $n^2 - 1$ mm is produced. By connecting two half areas together through a 1 mm bridge (and feed) a dipole meander antenna is formed so the final antenna occupies $(n - 1) \times (2n - 1)$ mm^2 and has total length of $(2n^2 - 1)$ mm.

As in Galehdar et al. [8], some restrictions were set to limit the number of possible structures. There are two important differences between applying ACO to the TSP and this problem. First, the solution will be in the form of a Hamiltonian walk, *not* a closed circuit. More importantly, not each antenna point neighbours every other point (as would be the case in TSP). Instead, a neighbour is a point that is located one millimeter north, east, south or west on the grid. Neighbouring points on the border of the grid will number less than four, as there is no wrap-around. An additional restriction is that the meander line starts on one of the grid boundary points of the half-area. Such restrictions make it difficult for ACO (an essentially blind form of search) to construct solutions that include each point of the grid.

The algorithm proceeds as follows. Given a boundary starting point on the grid, each ant chooses a direction in which to move one grid point at each step. In general a move may be considered to up to four neighbours, corresponding to the four directions, {north, east, south, west}. Neighbours are excluded if they either take the meander line beyond an edge or have already been used as part of the line. There are three ways that an ant is guided in the choice of its next direction to travel:

1. *Pheromone* – The pheromone matrix is an $n^2 \times 4$ structure. The latter dimension is direction. The pheromone component of the probablistic equations is given as $\tau(c, d)$ where c is the current grid position and d is the direction.
2. *Lookahead function* – In order to move each ant in such a way as to maximise its degrees of freedom, the number of unused neighbours of each of the neighbours of the current point is calculated. The greater this number, the higher the probability of choosing that direction.
3. *Straight line segment function* – If the path an ant chooses is allowed to fold upon itself, the likelihood increases that the ant will become trapped in a loop with no unused neighbouring points (and will thus be unable to move further.) Encouraging straight line segments helps to reduce the possibility of such premature termination, ensuring that all n^2 points are visited by the meander line. This function is easily calculated by comparing the current candidate direction with the last direction that the ant has taken.

Modifying equations 1 and 2 to become suitable for this problem simply means that that the heuristic function η needs to be replaced by $f(c, d)^\beta \times s(d, d')^\beta$ where $f(c, d)$ is the number of free neighbours of the point reached by traversing direction d from the current position c and $s(d, d')$ is 2 if $d = d'$ (d' is the previous direction), 1 otherwise.

If the number of neighbours for an ant at a particular step of the algorithm is 0, it has become stuck and cannot continue. In this case, the solution is feasible if the length of the meander line is $n - 1$ segments, else it is shorter than this required length and is hence discarded. At this point, the ant simply "dies" and is not considered for the remainder of the iteration. Local pheromone updating does occur for the components that these “dead” ants have added to their solution. Fortunately, in ACS this serves to discourage other colony members from following the ant into the same, prematurely-terminating path. Only the best feasible ant at each iteration may globally (and positively) update the pheromone matrix.

The aim of this first experiment is to maximise the efficiency of the antenna design. However, the calculation of the efficiency of the antenna design is considered as a black box computation. The structure of a complete antenna is passed to an external program (the NEC [2] antenna analysis software) to compute its efficiency. Therefore, ants cannot use incremental objective information to guide their paths.

4.1 Computational Experiments

The computing platform used to perform the experiments was a Sun Microsystems v880. The ACS parameter settings were given by the set $\{\beta = 2, \gamma = 0.1, \rho = 0.1, m = 10, q_0 = 0.9\}$. These values have been found to be robust by Dorigo and Gambardella [6].

The evaluation of meander line antennas is a relatively computationally costly exercise. The efficiency is determined at the minimum resonant frequency of the antenna. In this experiment, the resonant frequency was found by “sweeping” the operating frequency across some range of interest, and finding the frequency at which the reactive part of the antenna’s impedance is (positively traversing) zero. This is a very time-consuming process; while for our computer platform each solution evaluation of this real-world problem, on the 5×5 grid, required less than a minute, for a 10×10 grid it took 13 minutes. Given that ants will generate solutions that they have previously come across, it was feasible, and desirable, to implement a solution cache to reduce the total run-time. This cache stored all of the uniquely generated solutions within a run of the solver. If an ant generates one of the solutions in the cache, a look-up on the objective cost value was used (rather than a full evaluation of the antenna design). For the problem size described in this paper, this had the effect of reducing the typical run-time to a more manageable two hours.

As a folded antenna of increasing length is packed into a given area (by increasing the grid density, i.e. the number of grid points used within a fixed area) an exponential number of designs can be derived. While exhaustive enumeration of all feasible designs is possible for smaller grid sizes [8] it is evident that this will quickly become impractical. For example, for a grid of 10×10 it has been estimated there are of the order of 10^{13} feasible solutions and intelligent search algorithms become a practical necessity. For larger grid sizes it is also important to use parallel resources to solve these problems in a timely manner. A model to achieve this for ant colony optimisation was outlined in Randall and Lewis [13].

In this experiment we solved for meander lines on a 5×5 grid, running the ant algorithm ten times (by varying the random initialisation seed) for each of three values of q_0 (which determines the degree of "greediness") and with and without application of the straightline segment function, a total of 60 independent searches. Due to resource limitations at the time of the experiment only 6 processors were used, searches for different random seeds being processed sequentially. Had additional computational resources been available for this experiment, it would have been a trivial matter to reduce the wall-clock time required by performing independent, parallel searches for the individual random seeds. In addition, as noted above, this time could be reduced further by implementing parallel ACO.

Whilst 5×5 might be considered a small problem by optimisation standards, it does in fact provide a useful and practically applicable antenna structure. The ACS part was coded in the C language and compiled with `gcc`. The evaluation of the antennas' efficiencies were simulated by the NEC antenna analysis software. Each search was permitted 2000 iterations.

The results of running the ten ACS searches are summarised in Table 1, which shows the maximum and minimum efficiencies obtained, and the number of unique solutions evaluated for each search. Overall, the results showed that the ACS search engine is a very effective way of producing good meander line antennas. On all trials the ants produced good solution improvement over time, as shown, for example, in Figure 3. Figure 3 shows the convergence history of one of the ten searches performed, i.e., the best efficiency obtained to date at each iteration. As may be seen, during some initial period no feasible solutions are generated. Once a feasible path was found it was significantly improved relatively rapidly. In fact, one of the ten searches conducted found a solution within 0.4 % of the global optimum after 637 iterations (approximately 146 seconds of computational time). Due to caching, only 9 unique solutions had to be evaluated for this search and the greatest number requested by any of the searches was 29. This was achieved in a little over 3 hours of wall-clock time for the whole experiment. Thus our implementation is capable of finding the near optimal antenna structures as verified by results from exhaustive enumeration in other experiments [7] during which over 1000 unique solutions were tested.

4.2 Overall Remarks

Producing efficient meander line antennas is an important design problem for RFID devices. By using a grid-aware ACO, we were able to develop an implementation which was able to find a near-optimal antenna design for a 5×5 antenna grid. This is a notable result as the ants design antennas

Table 1 Results of the ACS solver.

q_0	Unique Solutions	Iterations Required	Maximum Efficiency%
Without straightline bias			
0.9	9	637	83
0.5	14	10	82
0.1	17	323	82
With straightline bias			
0.9	7	13	81
0.5	12	138	81
0.1	14	1904	83

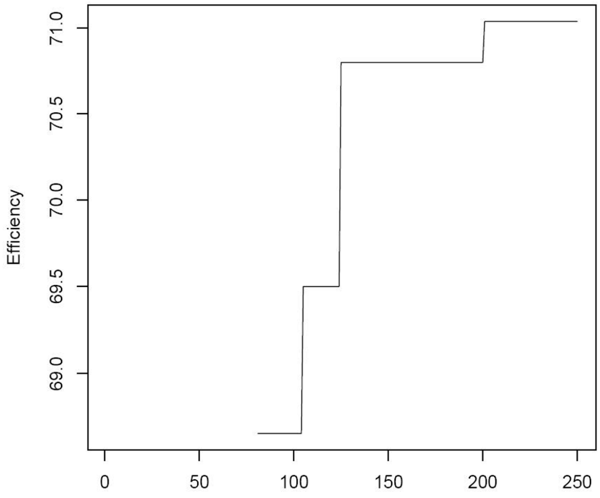


Fig. 3 Convergence history of a single run of the solver. ©2007 IEEE

based purely on a black box evaluator, without having any domain specific knowledge of antenna efficiency.

Given the proof of concept outlined in this section, investigation continued on efficient methods for producing antennas for larger grids, and means of improving antenna efficiencies using local refinement techniques.

5 Local Refinement Using the Backbite Operator

Local search is an integral part of ACO, and on the whole, greatly improves solution quality [6, 19, 20]. As an example, Stützle and Hoos [19] applied local search for a $\mathcal{MAX} - \mathcal{MIN}$ Ant System to the TSP and the quadratic assignment problem (QAP). For both problems, local search improved the runtime and solution quality compared to an ant system without this local refinement.

Refinement of meander lines cannot use the standard local search operators that are typically applied to the benchmark problems such as TSP and QAP. However, a well known approach to generate new Hamiltonian walks from another Hamiltonian walk is the *backbite operator* [11, 16]. Figure 4 shows how the backbite operator works.

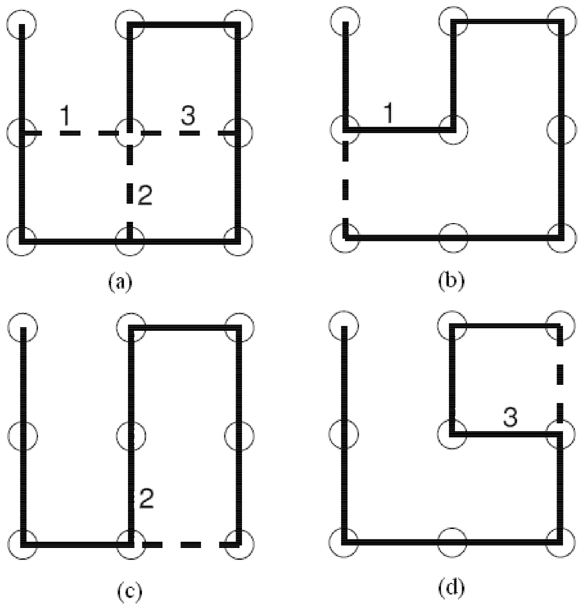


Fig. 4 (a) shows all possible backbite moves. (b) - (c) show all resulting structures after the backbite operation. ©2008 IEEE

Figure 4 (a) is the starting structure with the starting point in the upper left corner and the end point in the centre. To construct a new Hamiltonian walk we consider all neighbours at the end point. In this case there are three possibilities to insert a new connection. A notation convention for inserted links was adopted, as shown in the figure. Adding a new connection will create a circuit, which means another connection has to be removed. In Figure 4 (b) the connection shown as 1 in Figure 4 (a) has been added and the connection indicated by the dashed line must be removed. The new endpoint is in the lower left corner, where it is possible to perform further backbite operations. (In Figures 4 (c) and 4 (d) the connections of 2 and 3, respectively, were inserted).

If further backbite operations are performed, they can be described and arranged as an n-ary tree, where n has a maximum of 3, deriving from one particular origin structure. Figure 5 shows a portion of such a tree. The nodes in the tree can be labelled by extending the notation indicating the possible

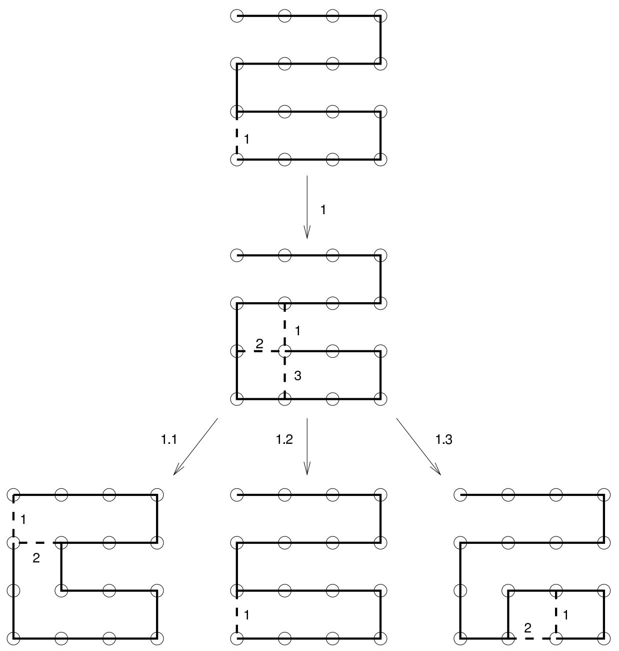


Fig. 5 The n-ary tree of possible structures derived using iterative backbite operations. ©2008 IEEE

inserted links at each stage. For example, a notation of 1.2 means that in the first step possibility 1 was used and in the following step possibility 2 was used. This tree, and labelling, can obviously be extended to arbitrary depth. In addition, it may be noted that all node structures can be fully and independently constructed based only on the root node structure. Thus a set of nodes to some predetermined depth can be constructed and all possibilities tested concurrently in a single evaluation step, given availability of sufficient, parallel computing resources. It only remains to be careful to prune the tree of loops.

A single iteration of the ACS algorithm, including the backbite operator, is outlined in Algorithm 1. The procedure shown will be repeated for some specified, maximum number of iterations. As shown, the backbite operation was applied as a hierarchical search after the ACS algorithm has produced a final solution. The local search was restricted to a given hierarchy search depth. All resulting structures were generated and the efficiencies calculated for each structure concurrently. Since the performance of the refined structures are evaluated *concurrently with the structure constructed by the ACS* the additional evaluations required for refinement can be performed without adding to the time required for a single iteration of the algorithm, provided sufficient parallel computing resources are available.

Algorithm 1 ACS algorithm including backbite operator

```
1: A population of ants begin construction of Hamiltonian walks on a Cartesian
   grid of specified size
2: for each ant, from a starting node on the edge of the grid do
3:    Add next node to a directed graph, according to the probabilistic selec-
      tion rules outlined in Section 3, coupled with next neighbour lookahead and
      straight line weighting, if appropriate
4:    Update local pheromone data
5:    If the ant cannot find a feasible next node, and all nodes have not been
      visited, terminate the ant for this iteration
6: end for
7: for ants that have constructed a valid antenna (a complete tour) do
8:    Iteratively apply the backbite operator to the antenna configuration, to a
      specified depth
9:    for original antenna configuration constructed by the ant and each configu-
      ration in the constructed tree of possible, refined antennas do
10:      Evaluate antenna gain using NEC antenna suite
11:   end for
12: end for
13: Update global pheromone
14: end
```

5.1 Computational Experiments

We solved for meander lines on grids ranging from 5×5 to 10×10, running the ant algorithm ten times (by varying the random initial seed used for the probabilistic operations of the algorithm). Each search was permitted 2000 iterations. Inspection of the results obtained showed that an iterative approach to application of the backbite operator – applying the operator, evaluating the resultant structures, and selecting the best for further refinement - was not, in fact, a desirable approach because efficiency can vary unpredictably between structures, leading to stagnation of the procedure in local optima. Construction of an entire tree of possible, refined configurations allowed the algorithm to search beyond intermediate structures of degraded performance.

For each grid size, the best antenna found was compared with those found without using the backbite operator. The following figures show the best structures found in previous work [14] on the left side and the structure further optimised using the backbite operator on the right side. The efficiency as a percentage is noted below each structure.

The 5×5 example in Figure 6 is quite impressive. Beginning with a search depth of 5 it took only 3 steps to find the global optimum as determined by Galehdar et al. [7] using exhaustive enumeration. The total number of antennas evaluated was 13.

Figure 7 shows the results for the 6×6 antenna derived using the algorithm without straightline bias described in earlier work [14] and Figure 8 shows

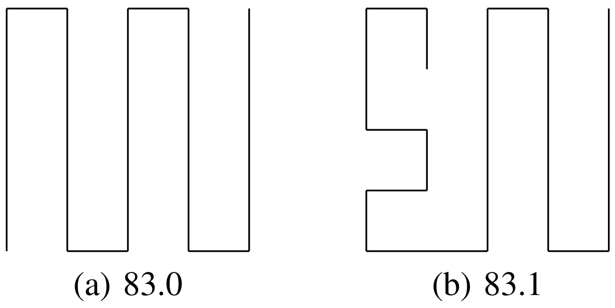


Fig. 6 (a) The original 5×5 antenna and (b) the optimised 5×5 antenna. ©2008 IEEE

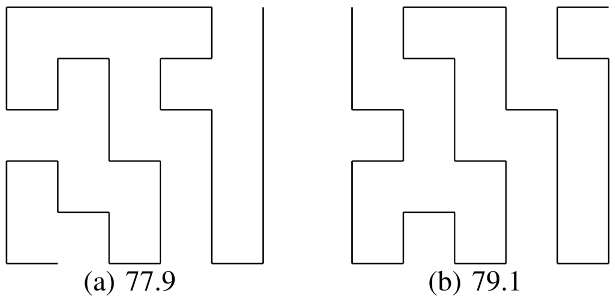


Fig. 7 (a) The original 6×6 antenna from the algorithm without straightline bias and (b) the optimised 6×6 antenna. ©2008 IEEE

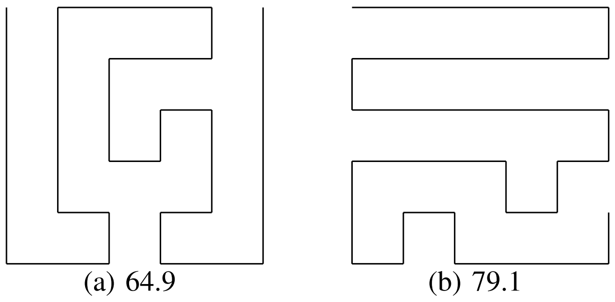


Fig. 8 (a) The original 6×6 antenna from the "straight" algorithm and (b) the optimised 6×6 antenna. ©2008 IEEE

the results for the 6×6 antenna obtained *with* straightline bias. For both the search depth was set to 25. For the first candidate antenna, the algorithm needed to evaluate a further 270 antennas and improved the antenna efficiency by 1.5%. For the second structure, there were an additional 395 antennas to evaluate. The efficiency was improved to 79.1%, the same as for the structure in Figure 7 (b), a relative increase of 21.9%. From these results it might be inferred that these both searches have now reached near-optimal results for this antenna size, but that the algorithm without straightline bias achieved a better result originally.

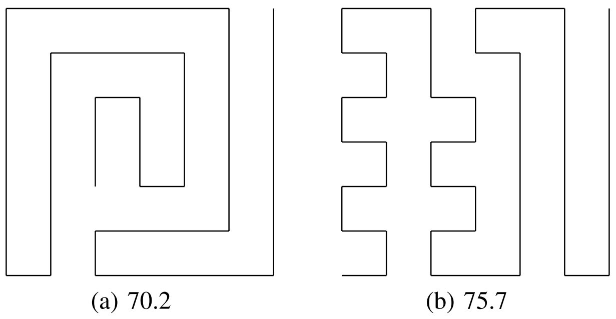


Fig. 9 (a) The original 7×7 antenna and (b) one of 7 equivalent, optimised 7×7 antennas. ©2008 IEEE

For the 7×7 case the algorithm had 891 antennas to evaluate (for a search depth of 25) and found 7 different, equally efficient antenna structures, one of which is shown in Figure 9. The increase in efficiency achieved was 7.8%.

To reduce computation time the search depth for 8×8 grids was reduced to 20. The local search had to evaluate 3193 different antennas and improved the efficiency 7.6%. The evaluation of this, the largest group of structures, took about one and a half hours using 20 desktop computers (a mixture of Athlon 64 and Pentium 4 computers in an *ad hoc* cluster.)

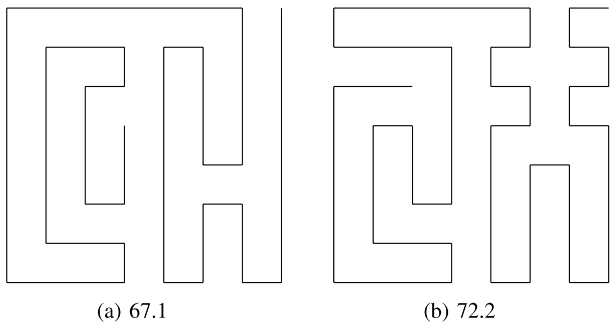


Fig. 10 (a) The original 8×8 antenna and (b) the optimised 8×8 antenna. ©2008 IEEE

For the 9×9 grids, the search depth was further reduced, because larger grids take increasing compute time to evaluate. For a search depth of 15, 1371 antennas had to be evaluated. A solution was found with efficiency increased by 16.8% compared to the original structure.

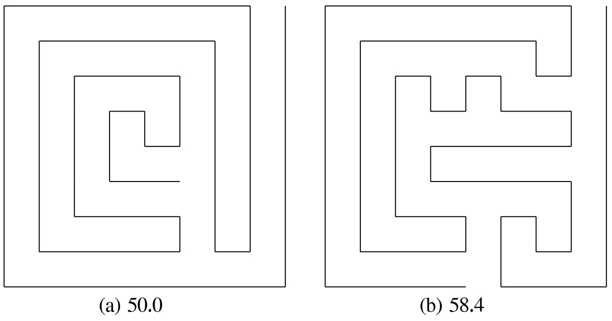


Fig. 11 (a) The original 9×9 antenna and (b) the optimised 9×9 antenna. ©2008 IEEE

For 10×10 grid, the search depth was further reduced to 15 and 1527 new antennas were evaluated. The efficiency was improved notably by 42.7%.

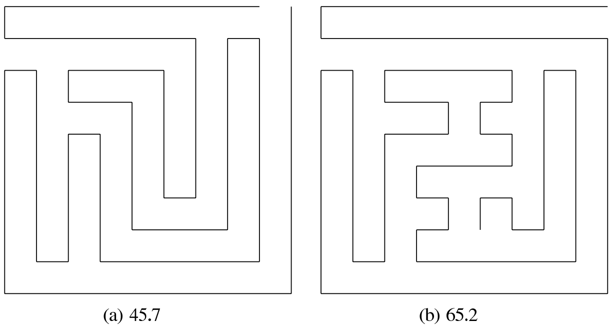


Fig. 12 (a) Original 10×10 antenna and (b) the optimised 10×10 antenna. ©2008 IEEE

In summary, use of the backbite operator for local structure refinement produced antennas with improved efficiency at all grid sizes. For smaller grids it either obtained the globally optimal solution, as verified by exhaustive enumeration, or can reasonably be assumed to have closely approached it for sizes for which complete enumeration of solutions is infeasible. It may be noted that in a number of cases, for these smaller grid sizes, the un-augmented algorithm had already derived very efficient structures. As grid sizes were increased, however, local refinement provided increasing degrees of improvement, indicating both the difficulty of the optimisation problem and the need for hybridisation with local search.

5.2 Overall Remarks

Overall it seems, that antennas with a simple, "serpentine" part and another more folded part are more efficient than the naive spiral and zigzag (or "plough") structures. The results show that the backbite operator applied in an hierarchical manner as a post-processing step is a suitable local search strategy for meander line antennas. Due to the independence of the evaluation function (i.e., the NEC tool), it is very easy to parallelise. Ideally, it would be preferable to be able to direct the local search more using domain specific knowledge, instead of generating a large number of solutions and testing them. However, no relationship between a small change in structure and the resulting change in efficiency could be found within the testing performed. Generally, the test runs have shown that starting with a good structure and applying the backbite operator hierarchically leads to resultant structures of higher efficiencies.

6 Optimising Efficiency and Resonant Frequency: A Multiobjective Approach

In previous sections we have demonstrated that heuristic search algorithms in general, and ACO in particular, can be used very effectively to improve the efficiency of small, meander line RFID antennas. In most real-world applications, however, an "optimal" solution involves simultaneously satisfying several objectives. Another issue of importance in the design of these antennas is that of the resonant frequency of operation. High Frequency (HF) RFID, which operates at 13.56 MHz, has been widely deployed for item management applications but fails where read distances of greater than 1m are required. More recently, Ultra High Frequency (UHF) has been investigated, providing for smaller antennas and longer read distances. UHF RFID has been developed to operate at a number of different frequencies, specifically, 433 MHz, 860 -956 MHz and 2.45 GHz. Two separate investigations were undertaken to design antennas for higher frequencies (3 - 5 GHz) and the lower band (433 MHz). Separate experiments were necessary because the parameters of the physical structure of antennas to operate in these difference frequency bands are quite different.

For the antenna design problem being considered maximising the antenna efficiency, η, and minimising the resonant frequency, f_0, were the two design objectives. In order to attain these objectives the ACO algorithm used in the previous investigations was modified. When optimising for a single design objective, it is simple to determine which ant has achieved the best solution, and should thus be allowed to globally update the pheromone matrix; it is the ant whose solution has the best value for the objective. However, when more than one objective is involved, the question of what is the "best" solution can be difficult to answer, particularly if objectives conflict. What is ideally required in this case is a method that delivers information on the trade-off between objectives.

To determine whether one solution is more attractive than another, a domination relation may be used. For solution vectors $\mathbf{x_1}$ and $\mathbf{x_2}$, when the following conditions are met:

- $\mathbf{x_1}$ is at least as good as $\mathbf{x_2}$ for all the objectives, and
- $\mathbf{x_1}$ is strictly better than $\mathbf{x_2}$ for at least one objective

then $\mathbf{x_1}$ is said to "dominate" $\mathbf{x_2}$ (denoted $\mathbf{x_1} \prec \mathbf{x_2}$). In the case where $\mathbf{x_1}$ and $\mathbf{x_2}$ dominate other solution vectors but not each other they are deemed mutually optimal solutions and referred to as Pareto-optimal. The set of Pareto-optimal solutions reflects the trade-off surfaces between the different objectives. This set of Pareto-optimal solutions is referred to as the Pareto-front.

This approach to optimisation of multiple objectives delivers not just a single solution but a set of (Pareto-optimal) solutions. Design engineers must still make some decision as to which particular solution is best for the requirements of a specific application. To do this they must choose between Pareto-optimal solutions based on some set of preferences ranking the different objectives. This choice can be generally be made in one of three ways [3]:

- *a priori* – a set of weights is determined for the different objectives and they are aggregated, usually by a simple algebraic sum. This aggregated objective function is then used to drive a single-objective optimisation algorithm. It is well known, however, that this approach commonly experiences difficulties finding solutions on Pareto-fronts of particular shapes [10] and can thus fail to deliver adequate solutions.
- *progressive* – the designer interactively supplies information about design preferences as the optimisation progresses. This is well-suited to problems where the objectives cannot easily be expressed in simple, numerical terms but tends to be laborious and time-consuming.
- *a posteriori* – the optimisation algorithm makes no attempt at ranking Pareto-optimal solutions but delivers the whole set to the designer for a decision to be made after the algorithm terminates. This can provide insight into the behaviour of systems in response to design parameters in addition to delivering particular solutions. The experiments described in this work used this approach.

Antenna structures constructed by the ants, and refined using the backbite operator, were now evaluated by the NEC software and two objective values returned, for η and f_0. The Pareto dominance relationships between different solutions were determined, and Pareto-optimal solutions accumulated in a continuously-updated archive. In the modified ACO, all ants that delivered Pareto-optimal solutions at an iteration were allowed to contribute an update to the pheromone matrix, the amount of their update being inversely

Algorithm 2 A single iteration of the multi-objective ACS algorithm.

```
1: A population of ants begin construction of the walks (antennas) on a Cartesian
   grid of specified size
2: while each ant has not completed construction of an antenna do
3:    for each ant, from a starting node on the edge of the grid do
4:       Add next node to a directed graph, according to the probabilistic selection
         rules outlined in Section 3, coupled with next neighbour lookahead and
         straight line weighting, if appropriate
5:       If the ant cannot find a feasible next node, terminate the ant for this
         iteration
6:    end for
7:    Update local pheromone data for all ants
8: end while
9: Apply the backbite operator to degree three to each solution/antenna
10: Determine if any of the population of solutions should be added to the Archive
11: For those solutions entering the archive, update the global pheromone
12: end
```

proportional to the number of ants contributing. Algorithm 2 gives an overall mechanical description of the multi-objective ACS implementation.

6.1 High Frequency Antenna

The first antenna structure to be investigated for both efficiency and resonant frequency was that for a high frequency, as the experiments undertaken to this point (on antennas of half-area 4×4 mm) yield antennas resonant between 3 and 5 GHz. Thus, the antenna design parameters mostly remain the same as for those described in Sections 4, and 5. However, the requirement that the antenna elements be a Hamiltonian walk on the Cartesian grid, i.e. visiting *all* grid points, was relaxed to allow the antenna wire to terminate before all grid points were included.

Computational Experiments

Once again we solved for meander lines on grids ranging from 5×5 to 10×10, running the ACO algorithm ten times (by varying the random initial seed used for the probabilistic operations of the algorithm). Each search was permitted 1000 iterations. The structures obtained by the ants were refined using the backbite operator, with a fixed tree depth of 3. An archive was maintained of all Pareto-optimal solutions obtained for each run. In addition, as the relaxation of the requirement that antenna elements be Hamiltonian walks on the grid (which meant *every* ant was able to produce a “feasible” solution requiring evaluation be the NEC software), the cache of previously-computed results was converted to a persistent database reusable across experimental runs, to further reduce the computational cost of performing the experiments.

With the increase in the number of feasible solutions to be evaluated, use of parallel computing resources became a practical necessity. An *ad hoc* computational grid was formed from a pool of about 20 computers, a mix of Intel P4 and Athlon X64 dual-core-based machines. The computers used were not dedicated to this task; jobs were distributed to machines according to their instantaneous ability to process them, taking into account machine load from other sources. For some experimental runs a number of computers may not have been available at all. The dual-core computers were always issued two jobs, when they were used. The computational details of the experiments are summarised in Table 2.

Table 2 Computational details of 4×4 mm experiment. (All times given in seconds. Averages are per processor core.)

Grid Size	q_0	Average CPU time per structure	Elapsed time	Average structures computed	Total structures computed	Total structures requested
5	0.1	0.39	3795	617	16430	31390
5	0.5	0.43	1810	141	3791	17690
5	0.9	0.40	442	17	449	11295
6	0.1	0.47	5795	1457	38601	48663
6	0.5	0.54	4382	646	16410	29967
6	0.9	0.59	1451	138	2428	14551
7	0.1	0.53	7345	2416	55364	60047
7	0.5	0.67	5789	1423	28327	39920
7	0.9	0.70	3163	338	7036	20240
8	0.1	0.81	10629	4096	57921	59966
8	0.5	0.79	9540	3282	45969	52868
8	0.9	0.94	4825	894	12519	25217
9	0.1	0.93	9787	2219	65573	66256
9	0.5	1.02	9672	1971	58894	61199
9	0.9	1.23	6205	674	19953	27894
10	0.1	1.10	11115	2275	67619	68137
10	0.5	1.28	11512	2075	62435	63972
10	0.9	1.51	9579	1084	32298	38577

Experiments for different values of q_0 at each grid size were processed sequentially, due to the limited computing resources available. In Table 2 the difference between the number of structure evaluations requested and the actual number computed is attributable to the effect of caching the results; it may be seen at each grid size that the number of unique solutions that need be evaluated progressively decreases as the persistent cache fills with each subsequent run. Very noticeable for the smaller grid sizes, this effect diminishes as grid size increases and the total search space the algorithm is exploring grows exponentially. It may also be noticed this effect remains significant with greater values of q_0: as the algorithm becomes "greedier" it tends to attempt to reuse paths already investigated.

Table 2 also reports structure evaluation times much lower than those reported in earlier sections. This is the result of a combination of two main factors:

- Relaxing the requirement that antenna paths be Hamiltonian leads to a reduction in the average length and complexity that must be simulated by the NEC software, which reduces the computation and time required.
- With the anticipated increase in the number of evaluations to be performed, the evaluation procedure was altered to reduce the computational cost. In the earlier experiments resonant frequency was determined by sweeping the operating frequency across a band of interest, evaluating antenna performance at a large number of discrete frequencies. This was modified to use a binary search algorithm within the same band, greatly reducing the number of discrete frequencies tested, and the time taken.

Figure 13 shows the ηf_0 Pareto-front. The resonant frequency range was 3.52 GHz $< f_0 <$ 9.55 GHz and efficiency range was 63.3% $< \eta <$ 98.4%. From the figure it may be seen that resonant frequency and efficiency are conflicting objectives: the highest efficiencies can only be obtained at high resonant frequencies, and *vice versa*. Figure 14 shows the structure of the antenna element with lowest resonant frequency, and Figure 15 shows the structure of the antenna element with highest resonant frequency

Figure 16 shows the ηf_0 Pareto-fronts for 5×5 (as "$\times$") and 9×9 (as "+"). It may be seen that the additional length possible using a 9×9 grid has allowed the production of antenna elements with lower resonant frequency, to a minimum of 2.41 GHz, but with a corresponding drop in efficiency, down to 32.4%. Figure 17 shows the antenna element structure that achieves this low resonant frequency. It may be seen that again this is a spiral structure, similar to the low-resonant-frequency structure for the 5×5 case shown in Figure 14.

Figure 18 is an expanded view of the ηf_0 Pareto-fronts for 5×5 (as "$\times$") and 9×9 (as "+") showing the difference at the lower frequencies. A close inspection shows that at a given operating frequency there is only a 3% difference in efficiency between antenna elements constructed on 5×5 and 9×9 grids or, alternatively, operating at a given, desired efficiency a difference in resonant frequency of only 4%. The remaining difference is the lower resonant frequency attainable from longer antennas produced using greater grid densities.

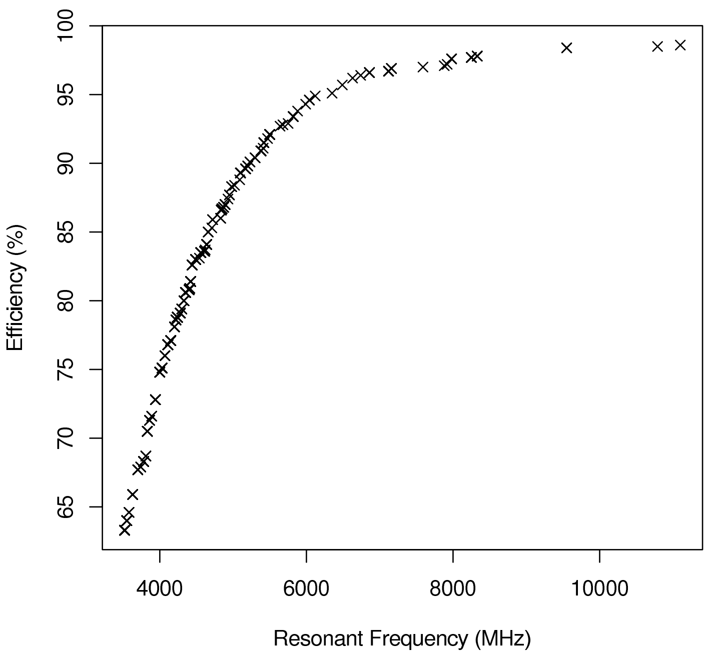


Fig. 13 ηf_0 Pareto-front for the 5×5 antenna.

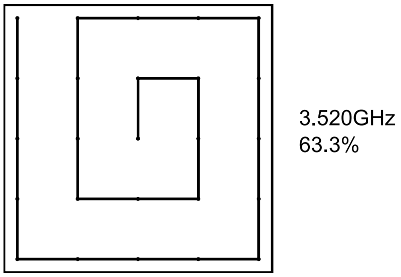


Fig. 14 Structure of the 5×5 antenna element with lowest resonant frequency, f_0.

6.2 "Low" Frequency Antenna

To design an UHF antenna able to operate at a resonant frequency in the lower, 433 MHz band it is necessary to increase the antenna size. To accomplish this with the ACO-NEC framework the only changes necessary were that track width be increased to 1 mm, the antenna half-area to 25×25 mm^2 and the bridge for the feed-point between the dipole arms to 6mm. All other details of antenna element specification and ACO algorithm remained the same as in preceding sections.

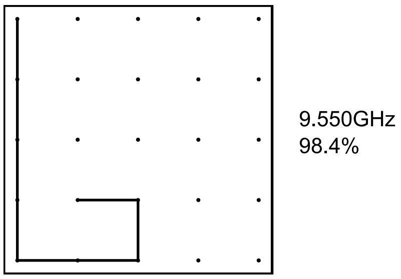


Fig. 15 Structure of the 5×5 antenna element with highest resonant frequency, f_0.

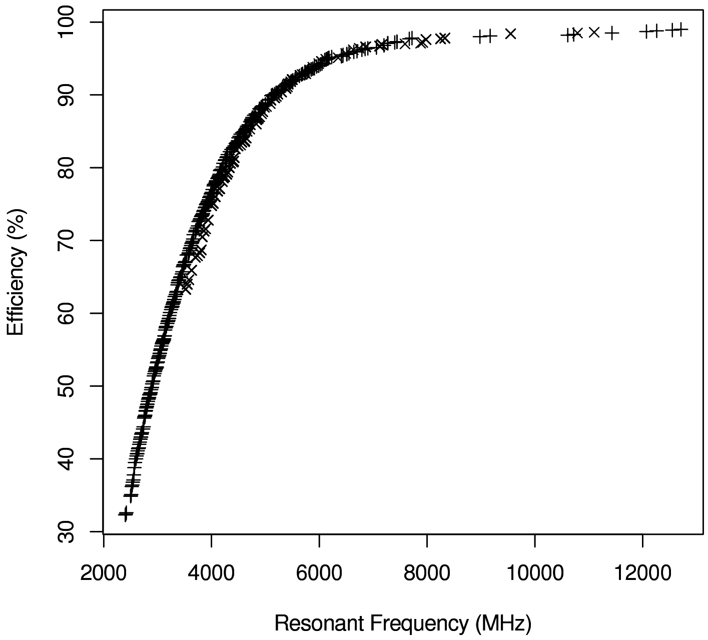


Fig. 16 ηf_0 Pareto-fronts for 5×5 (as "$\times$") and 9×9 (as "+").

Computational Experiments

Meander lines were constructed on grids ranging from 5×5 to 10×10. A computational setup similar to that of the experiment described in Section 6.1 was used. Computational details are given in Table 3. As might be expected, the behaviour of the ACO algorithm is remarkably similar between the 4×4 mm^2 and 25×25 mm^2: the ants see the same grid and know nothing of the scale of the antennas. There was a slight increase in the time taken for NEC

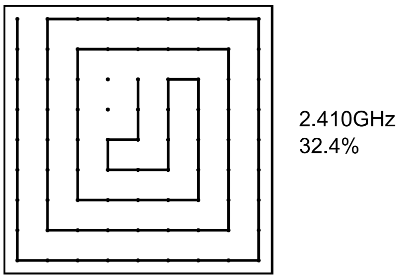


Fig. 17 Structure of the 9×9 antenna element with lowest resonant frequency, f_0.

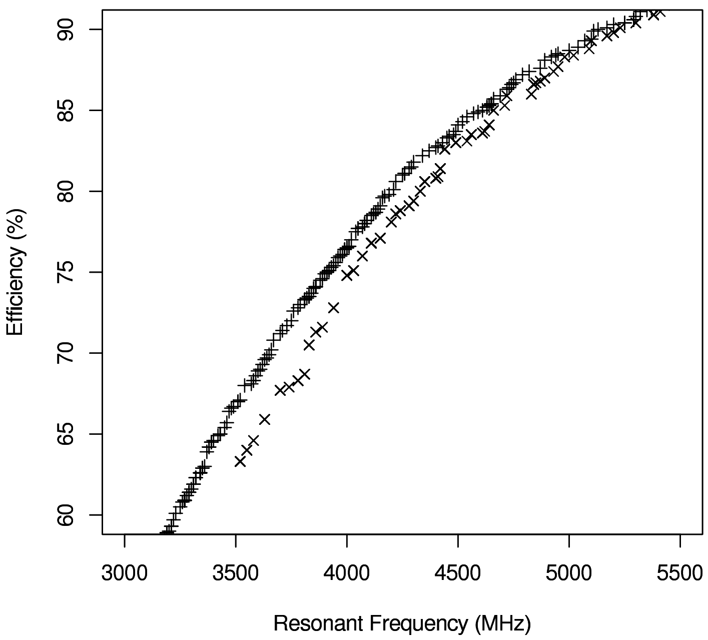


Fig. 18 ηf_0 Pareto-fronts for 5×5 (as "$\times$") and 9×9 (as "$+$") in the range 3.2 GHz $< f_0 <$ 5.5 GHz.

to evaluate the antenna elements for the 25×25 mm^2 experiment, but overall the computational times are also similar between the two experiments.

Figure 19 shows the ηf_0 Pareto-front obtained for all grid densities. The resonant frequency range was 350 MHz $< f_0 <$ 1520 MHz and efficiency range was 70.7% $< \eta <$ 99.8%. The standard operating frequency in this band is

Table 3 Computational details of 25 × 25 mm experiment. (All times given in seconds. Averages are per processor core.)

Grid Size	q_0	Average CPU time per structure	Elapsed time	Average structures computed	Total structures computed	Total structures requested
5	0.1	0.58	1290	373	2394	20286
5	0.5	0.54	1296	213	2244	17248
5	0.9	0.55	498	54	534	12128
6	0.1	0.41	4673	712	13540	39420
6	0.5	0.59	3444	376	8573	25856
6	0.9	0.71	1711	183	3009	15190
7	0.1	0.86	14097	5856	44064	59522
7	0.5	0.87	5546	734	17900	38650
7	0.9	0.96	1711	202	3009	15190
8	0.1	1.06	9658	2737	56161	63620
8	0.5	1.31	7885	1566	37387	50267
8	0.9	1.48	5150	580	13223	29743
9	0.1	1.34	17533	1885	61596	64200
9	0.5	1.53	10215	1846	48966	55585
9	0.9	1.59	7313	971	23688	32497
10	0.1	1.65	13926	2535	68372	69268
10	0.5	1.76	21140	2373	64379	65925
10	0.9	2.21	7313	1350	23688	32497

433 MHz. From the figure it is evident that optimised antenna structures have been obtained capable of operating at this frequency.

Figure 20 shows the structure of the antenna element with resonant frequency closest to this desired operating frequency (f_0 = 430 MHz, η = 83.6%). This antenna was constructed using a grid density of 10 × 10; inspection of Pareto-fronts for other grid densities showed feasible structures could also be obtained using grid densities of 9×9 and 8×8 but that the minimum resonant frequency obtainable with 7 × 7 was 460 MHz and lower grid densities had correspondingly higher minimum resonant frequencies. These results are summarised in Table 4. It should be noted that the resolution of the resonant frequency was 10 MHz.

Table 4 Performance of antennas in the 433MHz UHF band.

Grid Size	Resonant Frequency (MHz)	Maximum Efficiency%
5	570	92.8
6	510	89.5
7	460	85.7
8	430	82.8
9	430	83.5
10	430	83.6

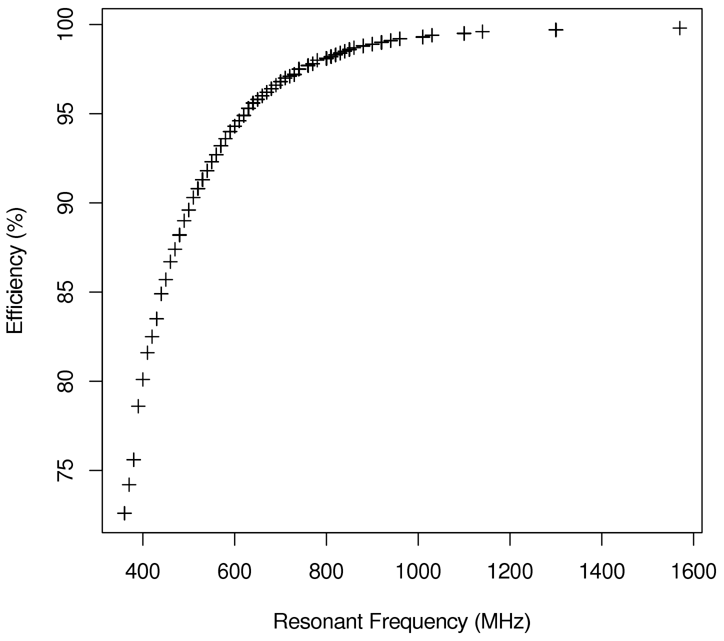


Fig. 19 ηf_0 Pareto-front for the 25×25 mm^2 antenna.

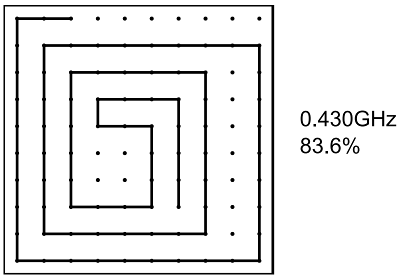


Fig. 20 Structure of the 10×10 antenna element with resonant frequency, $f_0 =$ 430 MHz.

6.3 Overall Remarks

The experiments described in this section have demonstrated that the combination of ACO with the NEC antenna analysis software is a practical and effective means of designing electrically small, meander line, RFID antennas to meet real-world design criteria. Antenna structures capable of operating with high efficiency at standard frequencies were found rapidly by an automatic process, an engineer only being required to choose a specific antenna design from several Pareto-optimal alternatives presented, according to preference.

For a standard, 433 MHz antenna designed to fit on a 25×56 mm substrate, the longest elapsed time for a single ACO run was a little under 6 hours, despite being constrained to use of heterogeneous, commodity computing resources of highly dynamic availability. While to complete the experiment described took a further 17 runs (of times ranging from under 10 minutes to a few hours) with sufficient computers available the entire investigation could have been completed in this 6 hour period, as each ACO run was independent of all others and could be performed concurrently.

It is possible that, given the insight gained from these experiments, it would be possible to obtain a feasible, useful design for a particular application by running the ACO search for only a single grid size. This would significantly reduce the computational cost of applying the design methodology in practice.

7 Concluding Remarks

The work described has demonstrated the practical application of ACO to a real problem of antenna design for RFID applications. The initial similarity of the design problem to the classical Travelling Salesman Problem, and the intuition that similar solution methods that have been successfully applied to TSP would be useful for this problem have been rewarded. Novel, feasible designs satisfying a number of design criteria have been efficiently and effectively derived by automated techniques using parallel computing resources. It should be noted that, to date, few or no attempts have been made to apply computational optimisation methods to this problem.

A range of methods have been described. Initially, a single objective optimisation was performed to design antennas with high gain. Application of a method for local refinement of the design proved capable of significantly improving antenna performance charactersitics. Finally, a multi-objective optimisation algorithm was developed to achieve simultaneous optimisation of multiple, competing objectives.

In addition to providing antenna designs fit for specific purposes, the information contained in the Pareto-optimal set of designs obtained has proved a rich source of insight for antenna design engineers. Inspired by these experiments, further work will investigate details of design such as wire width and track separation to determine what impact they have on desired performance objectives. It may also be instructive to consider alternative Multi-Objective Ant Colony Optimisation (MOACO) algorithms, such as those surveyed in Angus [1]. Beyond this, we wish to investigate other constructive heuristics with which to build meander lines as a basis of comparison to ACO.

Acknowledgements Portions of the work described in this chapter are reprinted, with permission, from Randall, M., Lewis, A., Galehdar, A. and Thiel, D., "Using Ant Colony Optimisation to Improve the Efficiency of Small Meander Line RFID Antennas", Proceedings of the Third IEEE International Conference on e-Science and Grid Computing, ©2007 IEEE, and Weis, G., Lewis, A., Randall, M., Galehdar, A. and Thiel, D., "Local Search for Ant Colony System to Improve the Efficiency of Small Meander Line RFID Antennas", in WCCI 2008 Proceedings (2008 IEEE World Congress on Computational Intelligence) ©2008 IEEE.

References

[1] Angus, D.: Multiple objective ant colony optimisation. Swarm Intelligence 3, 69–85 (2009)

[2] Burke, G., Poggio, A., Logan, J., Rockway, J.: NEC - Numerical electromagnetics code for antennas and scattering. Antennas and Propagation Society International Symposium 17, 147–150 (1979)

[3] Collette, Y., Siarry, P.: Multiobjective Optimization. Springer, Heidelberg (2003)

[4] Dorigo, M.: Optimization, learning and natural algorithms. PhD. thesis, Politecnico di Milano (1992)

[5] Dorigo, M., Di Caro, G.: The ant colony optimization meta-heuristic. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 11–32. McGraw-Hill, London (1999)

[6] Dorigo, M., Gambardella, L.: Ant Colony System: A cooperative learning approach to the trav eling salesman problem. IEEE Transactions on Evolutionary Computation 1, 53–66 (1997)

[7] Galehdar, A., Thiel, D., O'Keefe, S.: Antenna efficiency calculations for electrically small, RFID antennas. IEEE Antenna and Wireless Propagation (in press) (2007)

[8] Galehdar, A., Thiel, D., O'Keefe, S., Kingsley, S.: Efficiency variations in electrically small, meander line RFID antennas. In: Proceedings of IEEE Antenna Propagation Symposium (2007)

[9] Hayes, B.: How to avoid yourself. American Scientist 86, 314–319 (1998)

[10] Koski, J.: Defectiveness of weighting method in multicriterion optimization of structures. Communications in Applied Numerical Methods 1, 333–337 (1985)

[11] Mansfield, M.: Monte Carlo studies of polymer chain dimensions in the melt. The Journal of Chemical Physics 77(3), 1554–1559 (1982)

[12] Oberdorf, R., Ferguson, A., Jacobsen, J., Kondev, J.: Secondary structures in long compact polymers. Physical Review E 74 (2006)

[13] Randall, M., Lewis, A.: A parallel implementation of ant colony optimization. Journal of Parallel and Distributed Computing 62, 1421–1432 (2002)

[14] Randall, M., Lewis, A., Galehdar, A., Thiel, D.: Using ant colony optimisation to improve the efficiency of small meander line RFID antennas. In: 3^{rd} IEEE International e-Science and Grid Computing Conference, pp. 345–351. IEEE Computer Society, Washington (2007), http://dx.doi.org/10.1109/E-SCIENCE.2007.82

[15] Seshagiri Rao, K., Nikitin, P., Lam, S.: Antenna design for UHF RFID tags: A review and a practical application. IEEE Transactions on Antennas Propagation 53, 3870–3876 (2005)
[16] Sokal, A.: Monte carlo methods for the self avoiding walk. Monte Carlo and Molecular Dynamics Simulations in Polymer Science pp. 47–124 (1994)
[17] Sokal, A.: Monte carlo methods for the self-avoiding walk. Nuclear Physics B Proceedings Supplement 47, 172–179 (1996)
[18] Stockman, H.: Communication by means of reflected power. In: Proceedings of the Institute of Radio Engineers, pp. 1196–1204 (1948)
[19] Stützle, T.: The Max-Min Ant System and local search for combinatorial optimization problems. In: Voss, S., Martello, S., Osman, I., Roucairol, C. (eds.) Meta-heuristics: Advances and Trends in Local Search Paradigms for Optimization, pp. 313–329. Kluwer, Dordrecht (1999)
[20] Stützle, T., Hoos, H.: MAX-MIN Ant System and local search for the traveling salesman problem. In: IEEE International Conference on Evolutionary Computation, pp. 309–314. IEEE Press, Los Alamitos (1997)

The Radio Network Design Optimization Problem

Benchmarking and State-of-the-Art Solvers

Sílvio P. Mendes[1], Juan A. Gómez-Pulido[2], Miguel A. Vega-Rodríguez[2], Juan M. Sánchez-Pérez[2], Yago Sáez[3], and Pedro Isasi[3]

Abstract The fast growth and merging of communication infrastructures and services turned the planning and design of wireless networks into a very complex subject. The Radio Network Design (RND) is a NP-hard optimization problem which consists on the maximization of the coverage of a given area while minimizing the base station (BS) deployment. Solving such problems resourcefully is relevant for many fields of application and has direct impact in engineering, scientific and industrial areas. Its significance is growing due to cost dropping or profit increase allowance and can additionally be applied to several different business targets. Numerous works can be found in the literature dealing with the RND problem, although they all suffer from the same shortfall: a non-comparable efficiency. Therefore, the aim of this work is threefold: first, to offer a reliable RND benchmark reference covering a wide algorithmic spectrum, second, to offer a grand insight of accurately comparisons of efficiency, reliability and swiftness of the different employed algorithmic models and third, to disclose reproducibility details of the implemented models, including simulations of a hardware co-processing accelerator.

1 Introduction

Aimed at researchers, experts and practitioners, this chapter introduces the reader to a summarized evolution of the Radio Network Design (RND) paradoxal optimization research problem, and the employment and analysis of cutting-edge

School of Technology and Management
Polytechnic Institute of Leiria,
Leiria 2410 PORTUGAL
smendes@estg.ipleiria.pt
University of Extremadura
Caceres 10071 SPAIN
jangomez@unex.es
University Carlos III of Madrid
Leganés, 28911, SPAIN
yago.saez@uc3m.es

non-exact optimization solvers, broadly known as non-classical approaches, as opposed to the exact methods that primarily appeared from Operations Research.

RND plays a major role in various engineering, industrial and scientific applications because of the direct outcome that usually infer cost, profit or other heavy impact business performance metrics. This means that the quality of applied RND approaches have a transitive but straight attachment on industry economical plans.

On the other hand, the fast growth and merging of communication infrastructures and services turned the planning and design of wireless networks into a very complex subject. Although the attention it has deserved by the scientific community, this optimization field of research is still considerably obscure. Yet today, industry expertise is generally based on ad hoc or non-formal approaches. A plethora of scientific work has been developed around the RND optimization problem, although they all suffer from the same deficit – non-comparable efficiency. Radio network technology evolution has made this scenario recurrent due to the consecutive optimization experimental approaches that mainly consider the technological aspects of the RND problem instead of the canonical optimization formulations. As a direct consequence it is nevertheless impossible to identify the most effective formal method to tackle an RND instance optimization problem.

This chapter also gives an idea about how substantial improvements were accomplished on this field of research but clearly points out the gap at global scope research, i.e., the convergence on technology sustained research work weakened by the absence of a reference benchmark in order to evaluate algorithmic effectiveness, reliability, and swiftness. This problem is vindicated through a survey on scientific production, accounting accredited research work to point out the generalized growing problematic issue over the past 11 years. Although we have seen praiseworthy development, from an algorithmic expertise point of view, little novelty had been added since the development of the first RND GA parallel approach. This field of research has clearly been addressed to direct industry applied research. Subsequently the understanding and insight of the underlying problem and the respective formal algorithmic tackling models remain somehow unclear. In an informal manner, let's ask ourselves - what have we effectively learned after 11 years of RND research? Well, at a glance, we can deduce that the RND problem is being addressed, with ad-hoc technology dependent algorithmic implementations, but we cannot tell how well it is being addressed, i.e., from an algorithmic point of view, there are no certainties about the effectiveness of each one of the presented RND optimization proposals, albeit they all culminate in a single common concluding qualifying sentence – "…promising results…".

If we need to tackle a well known underlying problem, based on a yet-to-come radio technology which would be the formal proposal that fits the best, based on the surrounding scientific knowledge? Which would be the best approach to the reader's particular problem, or on a more global scope, what is the best approach on RND, period? These are difficult questions that blur future research and are also the main reason for the continuous booms of RND papers each time a new radio technology takes its grip on the market or industry. We've had it with GSM, UMTS, and it is currently beginning (again) with the 4th generation telecom radio standard. The main reason that slashes research according to mainstream technological intermissions is that in most cases, the developed optimization techniques rely heavily on technological constraints instead of the genesis of the underlying NP-hard problem. After 11 years of research, systematic chaos is established, hence turning it into a paradoxal problem.

The present situation is easily understandable; when we think that most of the developed works require talented skills turning them into state-of-the-art implementations, the undertaking of a reliable comparison among diverse state-of-the-art approaches, raises the stakes even more turning it subsequently into a grand challenge which has to recall and consider a wide in-depth range of algorithmic approach.

One of the key points in this grand insight is to offer a reliable RND benchmarking reference, with foundation-dissimilar approaches in order to cover a high algorithmic spectrum, consequently turning it into a valuable edifying tool for potential experimental applications regarding new or yet-to-come radio network technology. We emphasize our ranking and characterization tables in order to assist algorithmic evaluation, although further low level details are also available. Supplementary conclusions are made linking general results to parallel approaches and FPGA usage. Additionally the reader can delve right into the detailed implementation descriptions.

We combined our algorithms with advanced distributed and parallel computing platforms as the Condor high throughput computing platform [1] or BOINC in order to perform a directed, yet computationally heavy search using dedicated or non-dedicated computing resources. Previous partial experiments had been conducted [2-6], but this work scales up to a real world sized problem, in order to evaluate the effectiveness of a set of algorithms on such extent using a discretized representation. High throughput computing (HTC) pierces the time required to obtain scientific evidence, in which we could actually consider the RND optimization itself not less important, but also as a real world scientific research sidekick case study.

2 Related Work and Generalized Flaws

This section has the purpose to explain the related work or the state-of-art tightly associated to the main chapter subject. Additionally, hardware technological speedup enabler related works are introduced since they were used to side-kick our RND tackling.

2.1 RND Literature Review

Albeit some resemblances to Calégari's [7, 8] work on genetic algorithmic approaches (GA) for radio network optimization for mobile systems, developed in the mid 90's, this field of research actually focuses on the principle of minimization of resources rather than achieving the total coverage of an area. This happens in most real world problem cases, while the later scenario is fairly uncommon. Calégari GAs adopted the graph maximum independent set search method which attempts to find the largest independent set in a graph. We consider Calégari's work as the *ground-breaking work* when on the topic of RND.

Since then, many GAs have been applied with uncertain degree of success. Additional examples are [9-12], including several parallel and multi-objective implementations [13-16]. Calégari et al. [17] become known for developing his RND dominating set model (still supported on the maximum independent set search method) based on a hybrid implementation that combines a greedy algorithm with his previous GA development, presenting it in the form of a framework, formally known as STORMS (Software Tools for the Optimization of Resources in Mobile Systems). This approach had in mind some UMTS particularities. Several initiatives have been developed on the STORMS platform. Chamaret et al. [18] followed Calégari's work and tested seven different heuristics on the STORMS framework, evidently employing the maximum independent set search method.

Nevertheless several dissimilar approaches have also been identified. J. He et al. performed an unique related work [19] that consists in applying a pattern search algorithm called DIviding RECTangles (DIRECT) proposed by Joneset et al. [20]. The particularity of this work is that their algorithmic approach has been connected to a parallel 3D radio propagation ray tracing modeler running on a 200-node Beowulf cluster of workstations. A high focus has been made on the 3D ray tracing propagation model, based on geometrical optics when computing BS site power levels.

Isolated analytical and heuristic proposals have also been detected in this domain. Vasquez et al. proposed a Tabu-Based heuristic approach for antenna positioning [21] using the quintuplet BS compound (site, antenna, tilt, azimuth and power).

Elkamchouchi et al. [22] developed work based on a Particle Swarm Optimization approach (PSO) and included morphological data in their internal representation matrix and based their generic omni-directional wave on a 56dBm antenna.

Finally, RND directed research work has been detected where a demand-based criterion had primarily been taken into account, i.e., predicting traffic density. This kind of work, while out-scoping our main subject, is relevant because of some proposed novelties at algorithmic level. Tutschku proposes a *Set Cover Base Station Positioning Algorithm* (SCBPA) [23] applied on the *Maximal Coverage Location Problem* (MCLP) [24] with heavy restrictions on predicted traffic density. SCBPA is a simple Greedy-based heuristic. Ibbetson et al. propose two simple heuristics based on excess traffic re-distribution of BS [25] and Fritch et al. propose an approach on self-organizing sensory neurons implemented via Simulated Annealing [26].

2.2 RND Related Work Concluding Notes

The large amount of related work clearly showed us that there is a cycling research interlude each time a new radio-based technology emerges. Such research in its turn typically endeavors to adapt previous algorithms without any overriding consideration of research related to previous generation-technology. Each inspected work tackles a specific RND problem lying on imperative specific technology-dependent features and all use a myriad of optimization approaches. Every work additionally concludes how well and promising their achieved results are, but indeed it is very unlikely that all of the previous works achieve optimal results.

Due to this fact, after each intermission many researchers are erroneously guided and this is noticeable in Nebro's work [27] where after comparing only four algorithms deducts as being using the best cutting-edge algorithm that science has to offer, which is far from true as demonstrated in Section 7.

2.3 Hardware Technological Speedup Enablers

Field Programmable Gate Arrays (FPGAs) have been used in several fields of genetic computing. Donninger et al. presents the Brutus chess playing system [28]. Brutus combines a hardware and software engine, with the hardware part being an FPGA. The FPGA is used to compute the time critical part of the tree-based Chess algorithm search near the leaves. Additionally, Brutus uses the MPI library to parallelize processes over a small cluster of machines. According to its authors, Brutus is one of the top performer chess computer system [29].

Eklund [30] proposed an FPGA-based for the diffusion model of genetic algorithms. Specifically, the model relies on the implementation of a specially-tailored and relatively simple CPU, adapted to the problem being tackled. The simplicity of the CPU means that it requires a small amount of the FPGA gates and hence many replicas of the CPU can be set in a single FPGA. This way, large scale parallelism can be achieved.

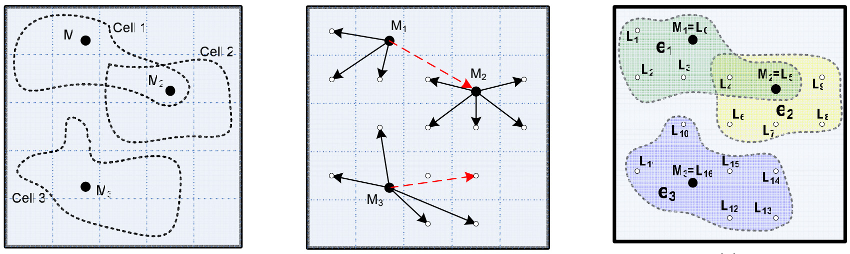
Although graphical processing units (GPU) are primarily devoted to graphics, recent models boost an amazing performance, several times higher than CPUs, all of this at affordable prices (close to the \$200 barrier). Moreover, due to the inherent parallelism of their internal organization, GPUs achieve higher benefits from Moore's law than CPUs, yielding a performance annual growth of 2, against 1.5 for CPUs. However, conversely to CPUs, the computing performance of GPUs is not directly exploitable for general-purpose computation. Indeed, porting existing applications is non-trivial [31], even for evolutionary computing which normally fits the parallel paradigm. Nonetheless, successful examples exist. For instance, Fok et al. have tackled evolutionary programming (EP) in a GPU, running EP mutation, selection and fitness computation with a speedup up to 5 [32]. Langdon and Banzhaf [33] propose a SIMD interpreter for genetic programming over a NVIDIA GPU card, achieving a seven time speedup relatively to a CPU-based execution. The recent advent of some frameworks, like AMD/ATI's "Close To Metal" (CTM) and the higher level NVIDIA's Compute Unified Device Architecture (CUDA) have eased the programmability of GPU [34]. This way, it is expectable that more and more developers will exploit the performance advantages of GPUs in EA.

3 RND Problem Formulation Model

The RND optimization problem comprehends the maximization of the coverage of a given geographical area while minimizing the Base Station (BS), hence, being intrinsically multi-objective. A BS transmitter is a radio signal transmitting device that irradiates any type of wave model and the part of the area that is covered by a BS is called a *cell.* If two or more BS transmitters are close to each other, their cells can overlap, and the locations inside these areas might have different degrees of coverage (for example, one location can be under the influence of two BS transmitters while another can be inside the cell of only one transmitter; in this case the second location has a lower level of intensity of the received signal).

We propose a model that is based on a hypergraph $H = (V, E)$ (a set of graphs), where E represents the hyperedges (edges which can contain any subset vertices of a hypergraph) and $V = M \cup L$, where M is the set of all possible BS locations, and L is the set of all potentially covered locations. Our hypergraph H is a *set system* composed by directed monomial graphs $p(x) = x^{E^{'}}$ in a way that a central vertex is achieved in each one of them. Each central vertex represents an irradiating source (*M*') and hyperedges are arcs in the form of *E*'=(x,y), directed from x to y (in this case any number of vertices can belong to *E*' within the (x,y) boundary). Figure 1 shows how the current model can be extracted from a discrete surface.

A fitness function f is required to evaluate the quality of a BS set *H*'. The fitness is described by the ratio of the square of the cover rate and the amount of BS transmitters used (1). It has also been reported as being widely adopted in the telecom industry [8], hence its usage in our experiences.



(a) (b) (c)

Fig. 1. *Three potential BS M1, M2, M3 and the associated cells (a) on a discrete surface. A graph set, whose arcs link transmitters to the location they cover, where the dotted lines represent directed hyperedges and vertex colouring has been considered for the irradiation sources (b). The Hypergraph representation is very analogous to its discretized surface counterpart (c).*

$$f(x) = \frac{CoverRate\ (x)^2}{NumberTran\ smittersUs\ ed(x)} \tag{1}$$

where

$$CoverRate(x) = 100 \cdot \frac{Neighbours(M', E')}{Neighbours(M, E)} \tag{2}$$

The problem we consider recalls the *Unicost Set Covering Problem* (USCP), which is a known NP-hard combinatorial optimization problem [35]. The RND problem differs however from the USCP in that the target is to select a subset of BS that ensures a good coverage of a given area, and not to ensure a total coverage. This emphasizes the principle of minimization of resources rather than achieving the total coverage of an area, since in most real world problem cases, these later scenarios are uncommon. On the other hand, the RND NP-Hardness is maintained in magnitude and is as tightly related to the field of mathematical combinatory as USCP.

4 RND Formal Assessment Specification

Prior to the introduction of our set of cutting-edge algorithms to tackle the RND problems some theoretical considerations are taken into account on how an accurate and reliable comparison can be achieved regarding the pluralism and heterogeneity of the implementation of such algorithms.

Our comparison methodology relies on the canonical RND problem formulation and is governed by two main directives: technology-independency and a normalized comparison criterion. The technology independency is achieved by disregarding any of the additional technological constraints that would be thrown into the problem, as for instance part of the BS quintuple properties definition (Antenna, Power, Azimuth and Tilt), path loss models, bandwidth zone prediction, etc. Instead we considered a theoretical isotropic radiating model, which is mainly used as a reference radiating model. Since RND is a well known NP-hard problem, technological constraints would only raise the combinatorial complexity (eventually by turning the search space more rugged) while the problem's essence would stay untouched. The normalized comparison criterion (or benchmark) is based on an Asymptotic Fitness Evaluation Effort Metric (AFEEM) since real-world applications will spend most of their computing effort on the evaluation of real wave based solutions instead of the algorithm per se. This means that the problem itself, besides being NP-hard is aggravated due to the intrinsic nature of the radio wave propagation that requires even more intensive computation. Thus, beside having a non-tractable rugged search space, the way it is explored (*f(x)*) requires even more computing power.

The RND Formal Assessment Specification (RND/FAS) describes the accurate formal process that is used when comparing diversified implementations that are aspirant for RND optimization. From an algorithmic point of view, the fitness function (1) is redefined according to the previous mathematical notation of the problem formulation, and therefore are equivalent. Equation (3) shows the accurate definition, where *M’* represents a candidate solution and *MSectors* represent the total sectors of the considered surface, (previously mentioned as grid) although our theoretical model disregards the shape of the surface.

$$f(M') = \frac{\left(100 \cdot \frac{|M'_L|}{MSectors}\right)^2}{is} \tag{3}$$

where

$M' \subseteq M$

$is = |M'|$, i.e., solution size

$M' = \{M_1, \ldots, M_{is}\}$

$M'_L = \{M_{L1}, \ldots, M_{Ln}\}$, where n is *Max(|L|)*

$|M'_L| = \{|M_{L1}|, \ldots, |M_{Ln}|\}$

4.1 Asymptotic Fitness Evaluation Effort Metric

The Asymptotic Fitness Evaluation Effort Metric (AFEEM) has been conceived to normalize the comparison of heterogeneous RND run-time environment discrepancies, including hardware and software issues (also known as benchmark routines for system clock calibration). The theoretical algorithmic RND wall clock run-time *WT* is defined in (4).

$$WT = \sum^{st} t\left(\sum_{j=1}^{is} h(e_j)\right) + t\left(\sum_{k=1}^{is} f(e_k^{GWM} \cdot Pwr)\right) \tag{4}$$

$M' = \{e_1, \ldots, e_n\}, \forall e = E' \wedge \forall e \in M'$

where *t* represents the concrete partial WT for a given instance, *h* represents an abstract algorithmic heuristic procedure, *f* the fitness function evaluator and *st* a given stopping criterion. *GWM* defines an abstract generalized wave model and *Pwr* defines the power of the BS (the remaining of the quintuple properties of the BS are disregarded).

After conducting profiling experiences, the *WT* using an Omni-directional isotropic reference radiation *IRR* gathers the result shown in (5).

$$\lim_{M \to IRR^C} WT(M') >= 90\% \cdot \sum cp \tag{5}$$

With C = 1

Where $\sum cp$ represents the total required computing power and *C* denotes a proximity complexity factor of the considered wave model *Q* and its method of employment *PLM* (Radial, Ray-Tracing, etc…) for each BS. Although we disregard most of the BS quintuple properties, the power *Pwr* constitutes a massive restrictive feature on any real or theoretical wave model, hence it's generalized consideration. The definition of the complexity factor C is given in (6).

$$C = (Q_{BS} + PLM_{BS}) \cdot Pwr \tag{6}$$

When replacing *IRR* for a real wave model *RWM*, hence *RWM* = *IRR*^{C} we obtain the trend shown in (9).

$$\lim_{M \to RWM} WT(M') = 100\% \cdot \sum cp \tag{7}$$

In this case $RWM^{C} = \infty$ and the 100% $\sum cp$ is never reached. Although no experiences had been made in order to induce (7), J. He [19] reveal their RND algorithmic approach connected to a parallel 3D radio propagation ray tracing modeler running on a 200-node Beowulf cluster of workstations, providing evidence about the computing power burden.

Equation (7) allows us to create a clock calibration Asymptotical Fitness Evaluation based Effort Metric (AFEEM) that effectively replaces the WT measuring, allowing the disregarding of hardware run-time platforms, mainly the processor(s) frequency or architecture (like parallelization through core replication). Software issues are also minimized whenever using highly optimized code compilers (like gcc) or intermediate compilers allowed on some interpreting enterprise run-time platforms (like J2EE or .NET).

Since $\sum cp$ and *WT* are highly correlated we can asymptotically deduce (8).

$$\lim_{M \to RWM} WT(M') = 100 \Rightarrow \lim_{M \to RWM} h(M') = 0 \tag{8}$$

4.2 Coarse Grained AFEEM

Population-based approaches usually rely on light heuristics, intrinsically defined by their breeding operators. In these cases a whole *M'* will be evaluated at once. Hence the *AFEEM* can be defined theoretically as presented in (9):

$$1 \cdot AFEEM_{CG} = \sum_{i=1}^{is} R_{h}^{WT} + \left(\sum_{j=1}^{is} f(e_{j} \cdot GWM \cdot Pwr) \right) \tag{9}$$

where R_h^{WT} represents the approximate *cp* of the equivalent *WT* for *h().* Since $RWM^C = \infty$ and $R = \sum cp - \sum_{j=1}^{is} f(e_j \cdot GWM \cdot Pwr)$. Using (8) a simplification can be made as shown in (10):

$$1 \cdot AFEEM_{CG} = 0 + \left(\sum_{j=1}^{is} f(e_j \cdot GWM \cdot Pwr) \right) \tag{10}$$

Taking *GWM* = *IRR* as the unitary reference value, we finally attain the ending simplification as presented in (11).

$$AFEEM_{CG} = 0 + \sum f(e) = 0 + 1 = 1 \tag{11}$$

Informally definition (11) means that the reference model sums 1 *AFEEM* whenever computing a *RWM* based *M'* evaluation. This is true for any *RWM*, because absolute measurements require the definition of the complexity factor C (6). Since we aim at comparison, the relative performance will be proportional on heterogeneous execution environments where the same $IRR^C \Leftrightarrow RWM$ is employed, yielding the result shown by (11).

4.3 Fine Grained AFEEM

Algorithms relying on heavy-based heuristics and/or incremental fitness evaluations do not always compute a complete *M'* at a given time. In these cases, the elements that compose *M'* are typically used to partially evaluate the solution. Equation (12) describes the theoretical *AFEEM* definition for a partially evaluated *M'*.

$$1 \cdot AFEEM_{FG} = \left(\frac{\sum_{i=1}^{is} R_t^{WT} + \left(\sum_{j=1}^{ts} f(e_j \cdot GWM \cdot Pwr) \right)}{is} \right) \cdot ne \tag{12}$$

Where

$ts = |U| - |E^T|$, i.e, transformation size

$E^T = \{e_i, \ldots, e_n\} \setminus \overline{\{e_j, \ldots, e_{tn}\}}$

tn represents the number of partially affected E' and *ne* represents the individual compound elements that have to be evaluated when *ts* = 1

As with the coarse grained definition, the same simplifications can be made, including the replacement of the *M'* transformation size *ts* for a single evaluation of a compound element of *M'* (13,14).

$$AFEEM_{FG} = \frac{0 + f(e \cdot GWM \cdot Pwr)}{is} \cdot 1 = \frac{f(e)}{is} \tag{13}$$

$$AFEEM_{FG} = \frac{1}{is} \tag{14}$$

Informally definition (14) means that the reference model sums $\frac{1}{is}$ *AFEEM* for each *RWM* based compound element *e* evaluation, recalling that $M' = \{e_1, \ldots, e_n\}, \forall e = E' \wedge \forall e \in M'$

A single algorithm can use both *AFEEM* profiling in a single run, depending on the type of evaluations employed, which means that $AFEEM_{FG}$ and $AFEEM_{CG}$ are a compatible additive metric that can be compared.

5 Optimization Algorithms

In this section we describe the optimization algorithms employed throughout this work including parameter characterization, for the sake of reproducibility.

5.1 Population-Based Incremental Learning

Population-Based Incremental Learning (PBIL) is a method that combines the genetic algorithms with competitive learning (typical in artificial neural networks) for function optimization [36, 37]. PBIL is an extension to the EGA (Equilibrium Genetic Algorithm) achieved through the re-examination of the performance of the EGA in terms of competitive learning.

PBIL attempts to create a probability vector from which samples can be drawn to produce the next generation's population. The algorithm general process is described in algorithm 1.

As we can see, the necessary parameters for PBIL are: the population size (*NS*, number of samples/individuals to produce per generation), the probability of mutation occurring in each position of the probability vector (*MUT_P*), the amount for mutation to affect the probability vector (*MUT_A*) and the learning rate (*LR*).

After initializing the probability vector *P* (each position equal to 0.5), the *NS* samples (individuals in the population) are generated. Each sample vector must be generated according to probabilities in *P*. Furthermore, each sample vector is also evaluated using the fitness function. Then, we look for the best sample *max*. This *max* sample is used in order to update the probability vector *P*, position by position, using the learning rate *LR*.

Finally, we have to mutate the probability vector *P*, position by position, using the mutation probability *MUT_P* and the mutation amount *MUT_A*.

Although PBIL has been used in very diverse optimization problems ([38-41] are some recent examples), surprisingly it has not been used in many telecommunication studies (only some few cases exist [42-45]). In fact, to the best of our knowledge, our proposal represents the first time that PBIL is employed for solving the RND problem, and this is another important contribution of our work. Preliminary experiments were carried out to find the best set of parameter values which are shown in table 1.

Algorithm 1. PBIL procedural Pseudo-Code

```
    P ← InitProbVector(each position $P_i$ = 0.5)
    while NumGenerations do
       # Generate Samples
          while NS do
              $sample_i$ ← GenerateSampleAccordingP()
              $evaluation_i$ ← Evaluate($sample_i$)
          end while
          # Find Best Sample
          max ← FindSampleWithMaximumEvaluation()
          # Update Probability Vector
          while LengthProbVectorP do
              $P_i$ ← $P_i$ * (1.0 - LR) + $max_1$ * (LR)
          end while
          # Mutate Probability Vector
          while LengthProbVectorP do
              if (random (0,1] < MUT_P) them
                  $P_i$ ← $P_i$ * (1.0 – MUT_A) +
                  random (0.0 or 1.0) * (MUT_A)
              end if
          end while
    end while
return max
```

Table 1. PBIL parameter table

Parameter	Value
NS	135
MUT_P	0.02
MUT_A	0.05
LR	0.10
Strategy	Elitist

5.2 Differential Evolution

Differential Evolution (DE) is an algorithm created by Ken Price and Rainer Storn [46]. Since 1994, DE has been used for many optimization problems with satisfactory results [47-49].

DE is a very simple population-based stochastic function minimizer, which can be categorized into a class of *floating-point encoded, evolutionary algorithms*. It is currently used in a wide range of optimization problems, including multi-objective optimization [50]. Generally, the function to be optimized, F, is done by the mean of optimizing the values of its parameters, where X denotes a vector composed of n_{param} objective function parameters. As with all population-based evolutionary optimization algorithms, DE handles a population of solutions, instead of a single solution for the optimization of a domain dependant problem. Population P of generation G contains n_{pop} solution vectors, each one, usually known as an individual of the population. Consequently, each vector represents a potential solution for the optimization problem.

At any time, a population P of generation G' contains n_{pop} individuals, each one containing n_{param} parameters (usually referred as *chromosomes*). In order to establish a starting point for optimum seeking, the population $P^{(0)}$ (initial population) must be initialized. This is usually done by seeding $P^{(0)}$ with random values that are within a given boundary constraints

The population reproduction scheme of DE is different from other evolutionary algorithms. From the first generation forward, the population of the following generation $P^{(G+1)}$ is created in the following way on basis of the current population $P^{(G)}$. First, a temporary individual (usually referred as trial) that can possibly populate the subsequent generation, $P'^{(G+1)}$, is generated as shown in (15).

$$x_{i,j}^{'(G+1)} = \begin{cases} x_{C_i,j}^{(G)} + F \cdot (x_{A_i,j}^{(G)} - x_{B_i,j}^{(G)}) & \text{if } r_{i,j} \le Cr \\ x_{C_i,j}^{(G)} \end{cases} \tag{15}$$

Where

$i = 1,\ldots n_{pop}, j = 1,\ldots,n_{param}$

$A = 1,\ldots n_{pop}, B = 1,\ldots,n_{pop}, C = 1,\ldots,n_{pop}, A_i \ne B_i \ne C_i \ne i$

$Cr \in [0,1], F \in [0,2], r \in [0,1[$

A, B and C are three randomly chosen indexes referring three individuals of the population.

F, Cr and n_{pop} are DE control parameters that remain constant during the search process. n_{pop} represents the population size, F is a real valued factor in range [0.0, 2.0] that controls the amplification of differential variations, and Cr is a real valued crossover factor in range [0.0,1.0] controlling the probability to choose the mutated value for x instead of its current value.

The generational scheme of DE also differs from other evolutionary algorithms. Accordingly, each computed trial vector (generally known as a *donor* vector) is compared with the target vector. The one with the lower value of cost function (or maximal fitness) $f_{\cos t}(X)$ will remain in the population of the next generation. The procedural pseudo-code is described in algorithm 2.

Algorithm 2. Differential Evolution procedural Pseudo-Code

```
  # Initialize population
  $P^{(0)}$ ← Initialize($P^{(G)}$)
  # Evaluate the fitness for each individual in the population
  Evaluate($P^{(0)}$($X_i$))
  Repeat until stopping criterion is met
      $X_A$ ← SelectRandomIndividual($P^{(G)}$)
      $X_B$ ← SelectRandomIndividual($P^{(G)}$)
      $X_{TARGET}$ ← SelectRandomIndividual($P^{(G)}$)
      Offspring ← $X^{(G)}_{TARGET_{i,j}} + F \times (X^{(G)}_{A_{i,j}} - X^{(G)}_{B_{i,j}})$ if $r_{i,j \leq Cr}$
      Evaluate(offspring)
      If offspring better than $X_{TARGET}$ then
          Replace $X_{TARGET}$ with offspring
  End Repeat
Return bestIndividual($X_i$,$P^{(G)}$)
```

DE design issues take into account the differential evolution fast convergence, proven in previous works [51], and its canonical self adapting differential mutation operator.

Additionally, we developed a differential mutation operator called *Nearest Point Differential Mutation* (NPDM) which uses DE differential mutation scheme and enforce the RND hard constraints. Before fitness computation of the trial vector, each gene is checked to see if the location is an available BS location (since differential mutation will create non legitimate alleles). If not, it is replaced with the nearest available location (using the Euclidian distance) not yet in the offspring. Preliminary experiments were carried out to find the best set of parameter values which are shown in table 2.

In its canonical form, the DE algorithm is only capable of handling continuous variables. The NPDM operator needs thus to enforce both integer representation

Table 2. DE parameter table

Parameter	**Value**
Cr	0.3
Npop	135
F	0.2

and the location constraints, al-though, the donor vector will use underlying temporary floating point values. According to Lampinen and Zelinka [52], the handling of integer and discrete variables in DE can be done rather easily, by truncating a real value to an integer, just before the fitness function evaluation. In-depth details about DE experiences can be found in Mendes, Gómez-Pulido, Vega-Rodríguez, Pereira and Pérez [5].

5.3 GRASP

The Greedy Randomized Adaptive Search Procedure (GRASP) is a recognized meta-heuristic that has been used on the solving of many combinatorial optimization problems in a successful way [53-61].

According to general literature, GRASP is an iterative process, where each iteration consists of two phases: construction and local search. The construction phase builds an initial solution, while the second phase explores the search space based on the result of the previous phase, hoping to find a better solution. The best computed solution after all GRASP iterations is considered as the final solution. Algorithm 3 depicts the pseudo-code for a GRASP procedure.

Algorithm 3. GRASP procedural Pseudo-Code

```
for k = 1 until Max_Iterations do
    Solution ← GreedyRandomizedConstruction(Seed)
    Solution' ← LocalSearch(Solution)
    BestSolution ← UpdateBestSolution(Solution',BestSolution)
end for
return BestSolution
```

The GRASP meta-heuristic is commonly composed of two main parameters: the number of GRASP iterations Max_Iterations and the initial Seed for pseudo-random numeric generation.

A solution is usually represented as a set of elements. The construction phase starts from an empty set and iteratively adds elements to it until reaching a feasible solution. This is achieved by means of a restricted candidate list (RCL) that integrates all existing elements sorted in function of their problem specific dependant myopic greedy evaluator. At each step of the construction phase the RCL will only be composed of elements that haven't been selected to be included in the initial solution. Furthermore, each time an element is added the cost or fitness of the evolving solution is updated. Usually the selected candidates are those that induce the smallest increment cost, which represent the greedy component of GRASP. A usual complement is to randomly choose, from the RCL, the next element to be added to the solution, which in turn represents the probabilistic component of GRASP. This allows the building of different feasible solutions, at each GRASP iteration.

The solutions returned by the construction phase are not guaranteed to be locally optimal considering neighborhood concepts. The search phase attempts to improve each initial construction by means of a local search algorithm that iteratively replaces the current solution by a better one.

Furthermore, the construction phase plays an important part, since good starting solutions are desirable. There are two basic strategies employed on exploring a solution's neighborhood:

1. *Best-improvement*: all neighborhoods are evaluated and the current solution is replaced by the best neighbor.
2. *First-improvement*: the current solution is replaced when finding the first better neighbor solution.

According to Resende and Ribeiro [62], in most of the cases, when applying both strategies, they mutually achieve the same quality in their final solution, but generally the *first-improvement* strategy takes a lower computational effort. They also observe that it is more frequent to attain premature convergence to a non-global local optimum when using *best-improvement* instead of *first-improvement*.

Further details, formal definitions and GRASP extensions can be found in [53, 56, 61-64], which also includes extensive analysis of GRASP meta-heuristics based on many applications.

In this work, the RND GRASP based approach was developed and is summarily described as follows.

- *Shrinking-RCL* (GRASP SRCL) – is GRASP implementation that uses a tunable greedy local search algorithm. The search procedure uses a canonical RCL greedy mechanism for selecting a new solution iteratively. Each new solution is computed by exchanging an element for any other that fits better, if such one is available. A *RCLSize* control parameter is also used to define the greediness of the LS and is based on a regular RCL mechanism. This implementation uses a continuously shrinking RCL, turning the search deterministic when *sizeof*(RCL) = 1, and ending it when *sizeof*(RCL) = 0. During the iterative search procedure the RCL shrinks when the *loopsize* parameter reaches zero, ending its execution when the RCL is empty. Preliminary experiments were carried out to find the best set of parameter values which are shown in table 3.

Table 3. GRASP Shrinking-RCL parameter table

Parameter	**Value**
RCL (Construction Phase)	20
Greedy *RCLSize*	5
Loopsize (LS Phase)	30

5.4 Variable Neighborhood Search

Variable neighborhood search (VNS) is a modern meta-heuristic introduced by Mladenovic and Hansen [65], based on systematic changes of the neighborhood search space to solve optimization problems. Its main strategy is based on the employment of more than one neighborhood structure during the search. Its main dynamics focus on the change of the neighborhood structure in a systematic way as the search progresses. This is one of the most recent meta-heuristics developed for problem solving in an easier way. It is acknowledged as being one of the very well-known local search methods [66, 67], getting more awareness day-by-day, because of its ease of use and accomplishments in solving combinatorial optimization problems as the one currently dealing with [68-79].

VNS is a simple and effective search procedure that proceeds to a systematic change of neighborhood. A common VNS implementation builds an initial solution, $x \in S$, where S is the whole set of search space, controlling it through a two-level nested loop in which the core one alters and explores via two main functions named *shake* and *local search* (see procedural pseudo-code in Algorithm 4). The outer loop works as an energizer, reiterating the inner loop, while the inner loop carries the key search. Local search looks for an enhanced solution within the local neighborhood, whilst shake diversifies the solution by changing it randomly to another local neighborhood. The inner loop iterates as long as the solutions keep improving, where an integer control parameter k defines the length of the loop, hence defining the number of shifting neighborhood structures. Once an inner loop is completed, the outer loop re-iterates until the predetermined termination condition is satisfied. Since the set complementariness of neighborhood functions is the key idea behind VNS, the neighborhood structure and the heuristic functions should be carefully chosen to achieve an efficient VNS implementation.

Theoretically speaking, intensification is achieved by the local search while the shaking of the neighborhood structure acts as a diversification mechanism, raising its probabilities to avoid non-global optima.

In order to develop an effective VNS algorithm, two kinds of neighborhood functions are required, $N_k^s(x)$ and $N_l^{LS}(x)$ each yielding a particular neighborhood structure association, where $N_k^s(x)$ and $N_l^{LS}(x)$ denote neighborhood functions for shake and local search functions, respectively. It is usually referred [65-67] that multiple neighborhood structures may be used for each function (*shake* and *local search*) consenting them to achieve different views of the search landscape and allowing the shaking phase to generate new starting solutions that lie near other local optima regions. For that reason, the indexes, k and l, are to be used for shake and local search functions, respectively, in order to ease switching from one neighborhood to another.

Algorithm 4. VNS procedural Pseudo-Code

```
1.  Initialization: Find an initial solution $x$.
2.  Repeat the following steps until the stopping condition is met:
        (a) Shake Procedure: Generate at random a starting solution
            $x' \in N_k^s(x)$.
        (b) Local Search: Apply a local search from the starting solu-
            tion $x'$ using the base neighborhood structure $N_l^{LS}(x)$ until
            a local minimum $x'' \in N_l^{LS}(x)$ is found.
        (c) Improve or not: If $x''$ is better than $x$, do $x \leftarrow x''$.
Return $x$
```

If the local search uses greedy strategy, then at Step *2(b)* (Algorithm 4) an iterative procedure tests the entire base moves returning the best neighboring solution until a local minimum is obtained. The shake procedure selects a random solution from the global search space.

There are many variants of variable neighborhood search such as *variable neighborhood decomposition search* (VNDS) [80] and *skewed variable neighborhood search* (SVNS) [67]. Given the flexibility of the technique, other variants of this algorithm can be employed [66].

Thousands of experiments have been executed in order to access several distinct VNS implementations, allowing the study on the upsides, downsides and pitfalls observed through exhaustive theoretical analysis and empirical testing. The two main representative VNS-variants that will be used in our analysis are briefly described as follows:

- *VNS* – Basic canonical VNS implementation with local search using a greedy LS (GLS) within a *gn* neighborhood and the shaking operator using a RCL Entropy Operator (RCLEO). RCLEO replaces the canonical VNS shaking procedure and has the purpose of changing the direction of the search space, at global search scope. Since a solution is represented as a set of elements, this operator works by means of a RCL that integrates all existing elements sorted in function of RND problem specific dependant myopic greedy evaluation *S*. RCLs are the core apparatus in GRASP implementations [53-61]. Each time the shaking operator is needed, the RCL will only be composed of elements that haven't been selected to be included in *x'*. Furthermore, each time an element is changed, the fitness of the evolving solution is updated. The selected candidates are those that induce the smallest increment cost, which represent the greedy component of the RCLEO. Also when triggered, the new entropic element is randomly chosen from the RCL, which in turn represents the probabilistic component of the RCLEO. This allows the building of different feasible solutions, at the end of shaking requests. The *ep* parameter defines the pool size of the RCL selectable elements $e \in E$, according to their *S* valuation (16). The *np* parameter defines the minimum number of required element changes in order to commit the shaking operation. Parameter values are shown in table 4.

$$S = | E_i \setminus E_j |, x_i \wedge x_j \in SearchSpace \tag{16}$$

Table 4. VNS parameter table

Parameter	Value
GLS Neighborhood *gn*	1.5*30 (max. isotropic wave radius)
RCL Size *ep*	5

- *GRASP VNS* – Implementation that allows dynamic changes on neighborhood range *nr* when executing the local search procedure, increasing it when no better neighborhood selections are found and decreasing it while better solutions are continuously found. This is an implementation that uses GRASP metaheuristic [53-61] to power the global search, replacing the shaking operator. The LS procedure is the GLS. Preliminary experiments were carried out to find the best set of parameter values which are shown in table 5.

Table 5. GRASPVNS parameter table

Parameter	**Value**
GLS Neighborhood *gn*	1.5*30 (max. isotropic wave radius)
RCL Size *ep*	5
Neighborhood Step *nr*	15
Initial Neighborhood	30 (max. isotropic wave radius)

5.5 Clustered Genetic Algorithm

A Genetic Algorithm is a population-based technique which uses a set of genetic operators (selection, crossover and mutation) to evolve a solution in a problem. The solution is represented as population of individuals, and the individuals with higher fitness values have higher probabilities to survive the selection.

The individual representation is one of the most important issues. In the canonical GA each chromosome is usually represented by a bit string, where each position represents a bitwise value. For this problem, we propose using data mining techniques in order to determine transmitter clusters and only allow one active transmitter in each cluster. This representation makes the search space smaller which is extremely useful. In the reference domain, the binary search space is $2^{1000} \approx 10^{301}$ and the new search space using 70 clusters is $16^{70} \approx 10^{84}$. In this proposal, one individual has the same number of genes than clusters. A gene is a list of transmitters in the cluster and a number with the active transmitter. Only one BS can be working in each cluster. If there is a BS which is inactive then it is selected by -1.

The first step in the Clustered Genetic algorithm (CGA) is to determine the clusters. For this purpose the WEKA implementation of the k-means algorithm with the employment of Euclidian distance was used [81]. Once the clusters are found, the population is randomly generated and evaluated for the first time. In each successive generation, part of the population is selected to breed a new generation. Tournament selection is used to select which individuals evolve to the next generation. The selection pressure has been adjusted by changing the tournament size in several experiments. Uniform crossover selects one gene of each parent alternatively and each child receives 50% of genetic information of each parent. Mutation occurs according to a user-definable mutation probability swapping the gene value. Algorithm 5 depicts the pseudo-code for the CGA procedure.

Algorithm 5. CGA procedural Pseudo-Code

```
# Find Clusters and generate initial solutions
$C_1$ ← findClustersInMap(Simple_KMeans)
$P_0$ ← generateInitialSolution($C_1$);
evaluation($P_0$);
while not stopCondition() do
   # Evolve best individuals
      P′ ← selection($P_0$);
      P″ ← crossover(P′);
      $P_{n+1}$ ← mutation(P″);
      evaluation($P_{n+1}$);
end while
# Find neighbor solutions
    2-OPT($P_n$)
```

After the stop condition is met, a variant of the 2-OPT heuristic is carried out, since it may redefine the solution at a low cost. The variant of the 2-OPT heuristic, also known as the pairwise interchange heuristic [82], has been selected as a LS method. Basically, the 2-OPT technique follows a procedure that searches on all the neighbors of the solution looking for better solutions. Briefly explained, the 2-opt consist of swapping the active antennas given by the CGA with their neighbors and checking if the final solution improves. Those small permutations between antennas can lead to slightly better solutions. To avoid high computational costs, the LS is limited only to the neighbors which are closer to each one of the antennas. Preliminary experiments were carried out to find the best set of parameter values which are shown in table 6.

Table 6. Clustered GA Parameters

Parameter	Value
Clusters	70%
Population Size	100
Selection Type	Tournament
Tournament Size	7%
Crossover Type	Uniform
Crossover Size	90%
Mutation Type	Cluster
Mutation Size	0.048%
Elitism Size	1

5.6 Clustered Chromosome Appearance Probability Matrix

The Chromosome Appearance Probability Matrix method (CCPM) is a modified GA in order to deal with micro populations. In PBIL algorithms, the recombination operator is replaced by a probability vector for each variable, and sampling this vector implies the study of the selections made by the algorithm until that moment. This concept, applied to Interactive Evolutionary Computation (IEC) can be done in order to speed up the evolution in regards to the user needs. This was the key motivation for developing this new method based on the GA. Basically, it consists of a GA that uses a probability matrix which drives the mutation operator

Algorithm 6. CCPM procedural Pseudo-Code

```
# Find Clusters and generate initial solutions
C1 ← findClustersInMap(Simple_KMeans)
P0 ← generateInitialSolution(C1);
InitializeStatistics(M1);
evaluation(P0);
while not stopCondition() do
   # Evolve best individuals
      P' ← selection(P0);
   # Update Probability Matrix with the selected
      updateProbabilityMatrix(M1, α)
      P" ← crossover(P');
   # Mutation sampling the Probability Matrix
      P"' ← orientedMutation(P", M1);
      Pn+1 ← cloneRemover(P"');
      evaluation(Pn+1);
end while
# Find neighbor solutions
   2-OPT(Pn)
```

towards the solution speeding up the convergence during the first generations. Algorithm 6 depicts the pseudo-code for the CCPM procedure.

The codification is the same as the explained for the CGA and the steps of the proposed algorithm are very similar. All steps are explained in detail in [83, 84]. The main modifications for the proposed algorithm are the evaluation, selection and mutation operators. In addition, a new operator that removes identical individuals and a 2-OPT search has been included. Preliminary experiments were carried out to find the best set of parameter values which are shown in table 7.

Table 7. CCPM Parameters

Parameter	Value
Clusters	70%
Population Size	100
Selection Type	Tournament
Tournament Size	20%
Crossover Type	Uniform
Crossover Size	80%
Mutation Type	Matrix Oriented
Mutation Size	0.4%
Elitism Size	1

5.7 Clustered Memetic Algorithm

The memetic algorithm (MA) is a combination of LS techniques and EAs. It is based on the concept of *meme* introduced by Dawkins [85]. The key idea of a *meme* is that an individual can change its genetic code during its live improving the evolution process. To simulate this concept of *meme* it includes an LS in the reproduction operators (crossover and mutation). For many domains the MAs enhance the performance of the GAs. In contrast, LS methods are less generic and difficult to customize.

The Clustered Memetic Algorithm (CME) representation uses the same codification as the ones described before (GAs). The crossover operator is the same as the one used in the GA, but an LS is done in order to find the best possible crossover.

Ideally the LS should calculate all possible crossover combinations and choose the best, but this means too many evaluations per crossover. Therefore, the developed LS randomly selects a predefined percentage of individuals and then finds the best crossover for those individuals. At this point, when the crossover is done, both parents are marked in order to avoid identical crossovers within the same iteration. The best possible crossover is guaranteed for the selected individuals and the offspring proceeds to the next step.

The mutation operator is the same as the one used in the GA plus an LS. In this case, LS is performed for trying to find the best possible mutation. Ideally, all genes must be changed in order to find the best possible mutation. As there are too many possible combinations, instead, the operator first calculates the number of mutations for each individual and then it makes a LS limited to blocks of 2 genes. This measure reduces computational costs of evaluating all possible mutation combinations. Algorithm 7 depicts the pseudo-code for the CME procedure while the best set of parameter values is shown in table 8.

Algorithm 7. CME procedural Pseudo-Code

```
# Find Clusters and generate initial solutions
$C_1$ ← findClustersInMap(Simple_KMeans)
$P_0$ ← generateInitialSolution($C_1$);
evaluation($P_0$);
while not stopCondition() do
   # Evolve best individuals
      P' ← BestSelection&Crossover($P_0$,%PCT);
      $P_{n+1}$ ← BestMutation(P',2);
      evaluation($P_{n+1}$);
end while
# Find neighbor solutions
   2-OPT($P_n$)
```

Table 8. CME Parameters

Parameter	Value
Clusters	70%
Population Size	100
Selection Type	Tournament
Tournament Size	7%
Crossover Type	Uniform
Crossover Size	90%
Mutation Type	LS
Mutation Size	0.07%
Elitism Size	1

5.8 Hybrid and Multi-start Variants

5.8.1 Fixed Neighborhood Tabu (MS FNS)

MS FNS is a hybrid LS procedure based on the greedy VNS and tabu techniques, mainly through the prevention of previously visited solutions. The global search space is managed by a multi-start mechanism. The best set of parameter values for MS is shown in table 9.

Table 9. Fixed Neighborhood Tabu parameter table

Parameter	**Value**
GLS Neighborhood *gn*	1.5*30 (max. isotropic wave radius)
RCL Size *ep*	5
Tabu tenure	5
MSC re-initialization *me*	10,000 Fitness Evaluations

In contrast to the previous algorithm, the multi-start versions presented next are all derived from the previously presented algorithms. Although each one of the former search procedures have unique and extendable techniques for avoiding local optima, several of our experiments gave us an insight that some pitfalls could not be avoided using a single modeled technique algorithm (or even employing multiple techniques). One of the considered options was to create multi-start versions of some of the promising algorithms that occasionally got trapped in local optima. Good results and high deviation was the substantiation that the algorithm could indeed be very effective but once in a while, also very deceptive (see Section 7.2).

5.8.2 Reversed Unleashed Neighborhood Search (RUFNS)

RUFNS is a triple hybrid searcher, originally based on the GRASP meta-heuristic combined with the VNS neighborhood operations and a tabu propagation technique, mainly through the prevention of previously visited solutions. The LS operations are interleaved with tabu techniques. This GRASP version delivers heavy based heuristics while executing the LS phase (opposing to our light weighted SRCL_GRASP and GRASP_VNS local search procedures). Preliminary experiments were carried out to find the best set of parameter values which are shown in table 10.

Table 10. Reversed Unleashed Neighborhood Search Parameter table

Parameter	**Value**
GLS Neighborhood *gn*	1.5*30 (max. isotropic wave radius)
RCL Size *ep*	5
Tabu propagation tenure	1

5.8.3 Multi Start VNS (MS_VNS)

MS VNS is a Multi-start version of VNS. This version replaces the shaking operator by a MS mechanism. Experiments were carried out to find the best set of parameter values which are shown in table 11.

Table 11. Multi Start VNS Parameter table

Parameter	Value
GLS Neighborhood *gn*	1.5*30 (max. isotropic wave radius)
RCL Size *ep*	5
MSC re-initialization *me*	10,000 Fitness Evaluations

5.8.4 Innate Immune VNS (IIVNS)

In the IIVNS the GLS local search procedure remains unchanged while population elements rely on a Simple Innate Acquired Immune Response System (SIAIRS) and a multi-start mechanism has been implemented when stagnation or premature convergence is detected during run-time. The foundation of the SIAIRS method is the recognition that the structure-related dynamics of any system, i.e., the inference, circular, interlocking relationships among its elements are often just as important in determining its potential behavior as the evaluation of the individual structures themselves this case the system dynamics inference has been considered with the aim of increasing IIVNS robustness when dealing with growing constraint rules.

Most of the algorithmic immunology aspects have been subjugated by the idea based on clonal selection, which accounts for the features of the adaptive immune response. Our current proposal does not rely on the adaptive immune responsive system. Instead we resorted to a SIAIRS which compute static innate affinity levels, based on the whole search space (representing the organism).

Before creating an initial population, one has to initialize immunity affinity on all elements that compose the search space. This requires a specific measurement based on the dynamic capacity of each element. The dynamic capacity of an element is the property that defines its odds in moving itself in the search space when manipulated by the LS operator. For the RND problem we defined this range by means of the sum of every predetermined BTS position in a *IN* neighborhood, using an Euclidean distance function. Then each predetermined BTS position is labeled according to its total value and finally an immunity amplifying factor $IF \in [0.1, \infty[$ is applied on every existing immunity labels in the search space. The result is called the Immunity Affinity *IA* and represents the immunity levels for each element in the search space. Each element needs to accommodate its immune base level according to (17).

$$x_{i_{immunity}} = IF \cdot \begin{pmatrix} Max(|Neighbors(x', IN)|) \\ -|Neighbors(x_i, IN)|) \end{pmatrix}, \ , x_i \in U, \forall x' \in U \tag{17}$$

For each attempt on processing such element, its immunity level decreases. While the immunity remains above 0 the operators have no effect on it. If an immune element is perturbed (by the RCLEO) or processed (by the GLS), it resets its immunity level immediately according to (20). High immunity levels decrease search convergence speed, hence the importance on adequate balancing of the *IF* factor.

If the search space is continuous and has no constraints then this component can be disregarded.

This approach delivers a double outer-diversity echelon: i) by its shaking operator and ii) its multi-start uttermost mechanism that relies on the initialization method. While the GRASP VNS approach emphasizes re-initialization (hence being denoted as GRASP), this one is classified as a multi-start variant due to its controllable initialization method. Experiments were carried out to find the best set of parameter values which are shown in table 12.

Table 12. IIVNS Parameters

Parameter	**Value**
GNO Neighborhood *gn*	1.5*30 (max. isotropic wave radius)
RCL Size *ep*	5
RCLEO commit size *np*	1
SIAIRS Affinity Neighborhood *IN*	2*30 (max. isotropic wave radius)
SIAIRS Amplifying Factor *IF*	1
MSC re-initialization *me*	50,000 Fitness Evaluations

6 Experimental Environment

RND experiments were tackled in a multi-perspective convergent approach. We’ve defined a credible RND asymptotic-based benchmark (Section 4), delved into a broad specter of solvers (including many bio-inspired), and employed advanced parallel computing techniques and hardware co-processing sustainability. Analysis, include solver meta-optimization and a formal extension proposal of the assessment axes usually employed in algorithmic characterization. Finally and for the first time we present a relative performance ranking reference for RND solvers (see Section 8), including a plethora of additional considerations regarding the problem instance dimensioning based on real empirical normalized run-time quality distributions.

Our implementations were partially built on a proprietary framework called RND3voSDK, which was used to mingle, blend, combine and fuse many bio-inspired techniques; starting from simple conventional local search procedures to cutting-edge meta-heuristics. Further deviations include extreme handling through high performance parallel and distributed implementations relying on desktop and grid computing as well as assistance through dedicated reconfigurable hardware co-processing based on FPGA devices.

6.1 Problem Instance

Although several distinct problem instances have been used, the results shown in section 7.2 are the most relevant. Data was drawn from the problem instance described as follows. Results are fully coherent with the remaining problem instance results from previous experiments.

A real world-sized problem instance, defined upon the geographical layout of the city of Malaga (Spain), has been used to solidify testing and present the numerical results of the algorithm performances. This instance, named *Malaga1K*, represents an urban area of $27.2km^2$. The terrain has been modeled using a 450*300 grid, where each point represents a surface of approximately 15*15*m*. This fine grained discretization enables us to achieve highly accurate results. A dataset containing 1,000 candidate sites for the BSs, and their corresponding coordinates on the grid, is used. The dataset can be found at the website [86]. The cell model for BS coverage, as explained in Section 3, is a omni-directional isotropic model, with a radius of approximately one half kilometer (30 grid points). In this scenario, according to the instance parameter definition, the maximum coverage that can be attained is 95.522%. There are two major potential uncovered areas: the sea and the mountains.

6.2 Heterogeneous Development Environments

Although the approaches studied in this paper aim to reduce the computational effort, the search space is still considerable, and thus we also resorted to high throughput and grid computing. Additionally some techniques combining software and hardware have been used in order to accelerate the computations. Besides being computationally demanding, the algorithms can easily be divided into independent tasks. Both these characteristics make these classes of algorithms particularly suited for execution over Condor and BOINC controlled resources. Indeed, they were instrumental for conducting our fine-tuning control parameter experiments, having run over some thousands of moderately long executions, each taking between three to five hours to complete, depending on the machine's speed. Next some of them are briefly described.

6.2.1 CONDOR Desktop Computing

The executions of the DE, GRASP, VNS, Hybrids and MS variant experiments described in this paper were carried out through the Condor high throughput computing framework [1]. Specifically, the Condor framework permits to harvest computing resources that would otherwise be left idle, allowing users with access to the Condor system to submit batches of independent tasks. These tasks are then scheduled by the Condor master over the available computing resources. If a task does not complete in the assigned machine – for instance, the remote machine is taken back for interactive usage or the machine is simply turned off – the execution lease times out after a given time interval and Condor automatically reschedules the task to another machine. All of this is practically transparent to the application programmer, with application submitters only providing the unchanged application binary (this needs to be a console application), a specially tailored submit file (this is a simple text file holding instructions to the Condor system, such as the required environments, for example, Windows or Linux, or what are the command-line arguments to be passed for the application), and the required input files.

Since the algorithms are computationally intensive and can easily be split into independent tasks, an easy to use high throughput computing system like Condor is an important tool for running such class of algorithms, and in fact was instrumental for conducting a huge share of our experiments.

At the academic institution where all experiments were performed, a Linux machine fulfils the role of the Condor server, while the Condor client is installed on over 170 Windows XP machines that are distributed through 10 classrooms (each classroom holds 17 machines). Four additional client machines (the fastest ones) belong to an advanced laboratory. The pool of client machines ranges from P4@2.4 GHz to Core 2 Duo 6600@2.40 GHz. Each type of machine was classified according to the NBench benchmark [87] integer and floating-point performance indexes. The NBench's indexes are used to assess relative performance among the monitored machines, since the same benchmark binary was used throughout the machines. The fastest machines – Core 2 Duo 6600@2.40 – are roughly 70% faster than the slowest ones (P4@2.4 GHz). Furthermore, the Windows machines are primarily assigned for teaching activities (essentially to support classes), and are also used by students for their practical assignments and other e-activities (e-mail, web, etc.). Therefore, the machines are primarily devoted to interactive usage, with Condor tasks being scheduled to a machine only when no interactive user is logged on. Additionally, whenever an interactive login occurs at a machine that is running a Condor task, the task is suspended, and after 10 minutes, if the interactive usage persists, the task is evicted and rescheduled to another machine. Again, this conservative configuration was adopted to prioritize interactive users over Condor's tasks, although it provokes a high churn rate of tasks, since machines are frequently used for interactive usage, resulting in a vast percentage of interrupted executions. This is aggravated by the fact that the classrooms are heavily used (they are open 20 hours on weekdays, and 13 hours on Saturdays, only closing on Sundays).

6.2.2 BOINC Desktop Grid Computing

BOINC (Berkeley Open Infrastructure for Network Computing) [88], [89] is a system for “volunteer computing” and “desktop grid computing”. Volunteer computing uses computers volunteered by the general public to do distributed scientific computing. We have used the middleware system BOINC in order to perform many different executions of the PBIL algorithm in parallel. In this way, we can do a deep survey about which are the best parameters and combinations for solving the RND problem.

Our experiences have demonstrated us the utility of using BOINC. We have added 100 computers of our University to the volunteer computers joined to the RND@home project [90], allowing us a very important computational power for our purposes. The used computers were distributed along the campus, with approximately 20 PCs in each laboratory. Our BOINC clients had different characteristics: different processors (Pentium IV and AMD), with various frequencies (from 1.8 GHz up to 3.2 GHz), and under diverse operating systems (GNU/LinEx [91], other GNU/Linux and Windows XP).

We planned a total of 25280 different experiments. We foresaw that a total of 32454 hours (3.71 years) would be necessary in order to perform all these experiments. Using BOINC, we have concluded all these experiments in 6 months. Furthermore, we have to highlight that during the month of August all the BOINC clients were available 24 hours per day (in this month the University is closed to academic activities), but this is not true for the rest of months. The computers were used by the students in our University, reducing their availability.

Comparatively to the Condor approach, the BOINC middleware requires not only a specifically tailored application (the application needs to use the BOINC client API), but also a whole server-side infrastructure (used (1) to distribute tasks and collect results and (2) to give feedback to resource donors through forums and contribution ranks). Moreover, in order to participate in the computation, resource donors need to be attracted to the project and explicitly register their machines to the project. On the contrary, the Condor approach is mostly transparent, since only rather simple submit files need to be prepared, with practically no changes required for the applications itself. However, a BOINC-based project has the potential to reach several thousands of resource donors, where all expenses (computer acquisition and maintenance, network bandwidth, etc.), apart the server-side ones, are supported by the volunteers, while Condor is limited to single-geographical sites (mostly local area environments) and thus has a much lower potential for scalability.

6.2.3 Reconfigurable Computing

Reconfiguration of circuitry at runtime to suit the application at hand has created a promising paradigm of computing that blurs traditional frontiers between software and hardware. This powerful computing paradigm, named reconfigurable computing (RC) [92], is based on the use of programmable logic devices, mainly field programmable gate arrays (FPGAs) incorporated in board-level systems. FPGAs have the benefits of hardware speed and software flexibility, being a good alternative for many real scientific and engineering applications [93, 94].

The interest of a hardware solution based on FPGAs is to determine if it is profitable to run an evolutionary algorithm accelerating some of its calculations. Since the biggest resource consumption comes from the arithmetic computation of the fitness (see section III.a), we have designed and implemented an arithmetic co-processor to relieve the main processor from this task, introducing the largest possible degree of parallelism. This is the main advantage of the hardware exploitation in order to increase the efficiency in comparison with software solutions. As an example of the promising results, on a 724x724 terrain with the same isotropic omni-directional coverage of radius 30, the FPGA solution (with a 13 parallel processor architecture on a Virtex4 xc4vlx200 FPGA device running at 48 MHz) outperformed a computer-based solution (Intel Core2Duo 2GHz PC) by 1.145% (yielding an effective speedup of 11.45). Further details, including architecture design and additional results can be found in Gómez-Pulido, Vega-Rodríguez, Pérez and Mendes [95].

Although several works [4, 9-11, 27, 96, 97] have addressed the RND optimization problem, the role of EAs has not been able to fold in the telecom industry, mainly due to the high computational efforts demanded when applying real wave models. To cope with this technological constraint, we designed an FPGA hardware device prototype that acts as a dedicated RND co-processor, enabling the usage of real wave models on software-based RND algorithms.

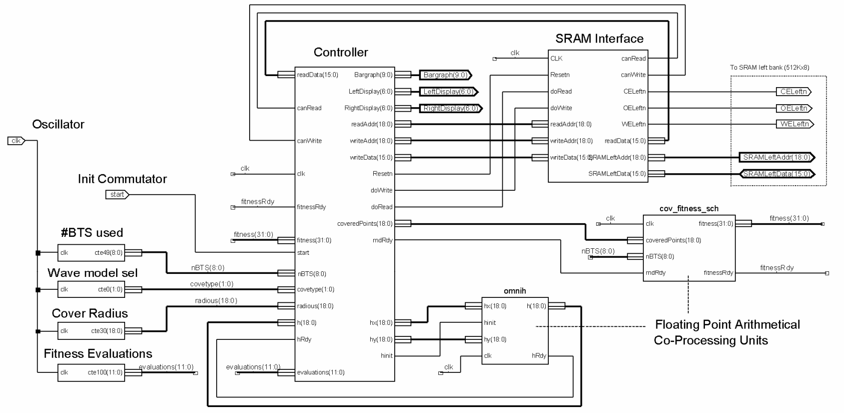
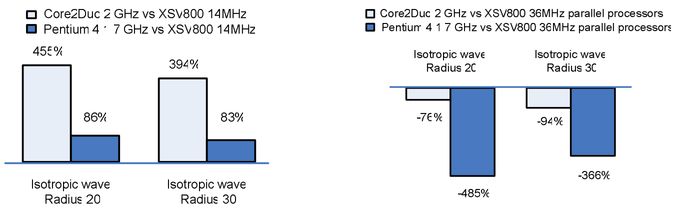


Fig. 2. Prototype Architecture designed to slot in 2 wave models

This prototype was exclusively designed to perform the wave coverage computing, thus assisting the software algorithm. Its design considers the same isotropic omni-directional wave model used in all algorithms, allowing a direct comparison. Fig. 2 shows the experimental architecture of the co-processor, identifying its main parts. The prototype uses an SRAM interface for memory access where the terrain matrix is loaded at initialization. The controller is its main component and computes the wave coverage related math. The floating-point arithmetical co-processing units relieve the main controller on these operations.



(a) (b)

Fig. 3. – 14 MHz & 36 MHz FPGA testing performance comparison

Fig. 3a depicts a big advantage (455% faster) for the software coded solution running on a 2-GHz PCore2Duo when compared to an obsolete 14 MHz Xilinx Virtex 800 FPGA [98] on a Xess XSV800 board [99]. A subtle detail can be observed in both plots: an increase in the arithmetical operations (obtained by comparing the 2 radius values), decreases the software coded solution advantage, an indicator that the FPGA-based approach is more suitable for handling complex wave model related to arithmetic calculations.

After the increase of the FPGA clock frequency and the replication of its basic architecture allowing parallelization, we used a synthesis and simulation tool that allowed us to predict performance at 36 MHz clock frequency operation with 4 parallel processors on the same obsolete device. Fig. 3b depicts this comparison, giving a clear advantage to the FPGA implementation.

Our last simulation targeted a 48 MHz Xilinx Virtex4 xc4vlx100-11ff1513 that is able to embed 13 parallel processors. It yielded a performance outgrow of 4751% when compared to the 2 GHz Core2Duo, an irrefutable proof that this technology is currently an enabler for real EAs software-based RND design tools capable of managing real wave models. Further optimizations can be achieved, as for instance, eliminating the external SRAM accesses.

7 Empirical Result Analysis

In this section we present and describe the results and the experiments performed with the different optimization algorithms described in Section 5.

7.1 Report Planning Method

Our experiments promote thoughtful [100], well-planned, and algorithmic extensive testing, full disclosure of experimental conditions, including the integrity and reproducibility of reported results. The most relevant results we currently present are based on the following measures:

- Effectiveness - Defines the fitness-based quality of the results.
- Computational Effort - Speed of computation is obviously a key factor. In this work the computational effort is based on the FEEM definition as shown in Section 3.1
- Algorithmic reliability - Defines a degree of confidence for a given algorithm to yield good results, according to its average effectiveness.

Additionally all results were derived from statistical experimental design techniques aiming at the reduction of variability and reliability within the results and promoting a comprehensive report of the results.

For each of the proposed algorithms, 30 independent runs have been conducted with a stopping criterion of 5,000,000 fitness evaluations each. All presented results rely on the statistical values yielded by each one of the 30 runs.

7.2 Algorithmic Comparison

In this problem instance the optimal value is unknown. Fig. 9 depicts the average fitness achieved per algorithm. This measure is based on the average convergence point *P'* obtained through the run-time distributions. There seem to be two comprehensible (fitness-based) subsets of algorithms denoted by the divisor in Figure 4.

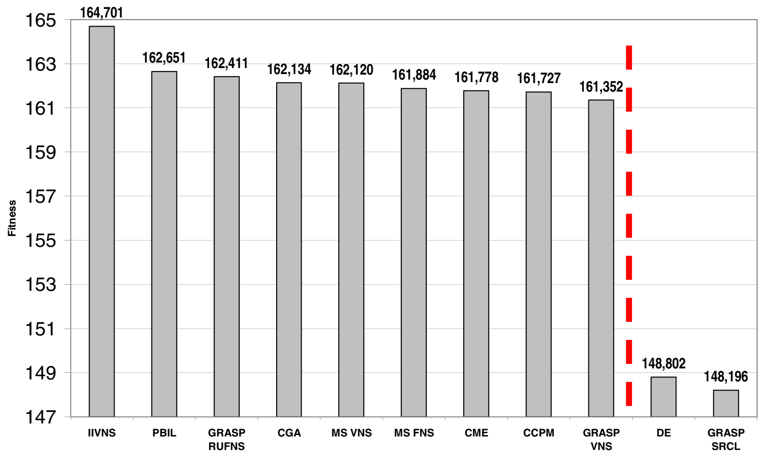


Fig. 4. Average effectiveness

The convergence point *P'* is defined by the AFEEM that have elapsed until the best result has been found. Figure 5 gives an overview of the average convergence point *P'* obtained through the *AvgSeries* $Avg(\sum_{i=1}^{30} AFEEM(P_i'))$ run-time distributions. It is

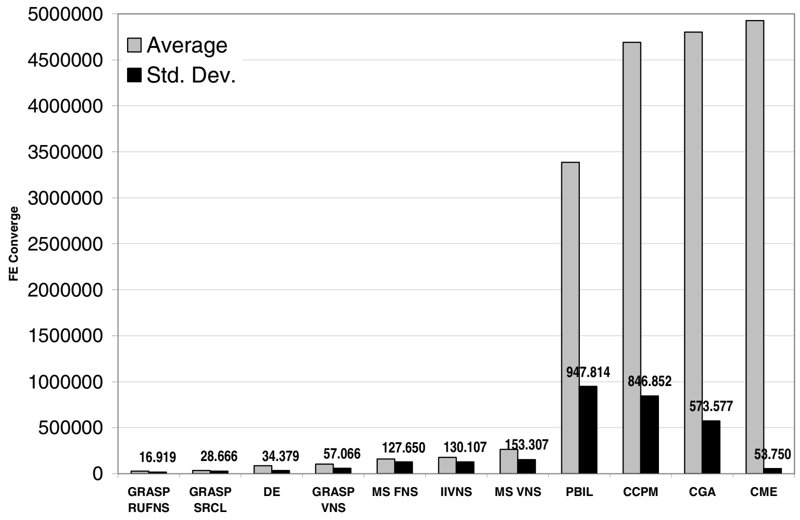


Fig. 5. Algorithmic AFEEM computational effort on convergence point P'

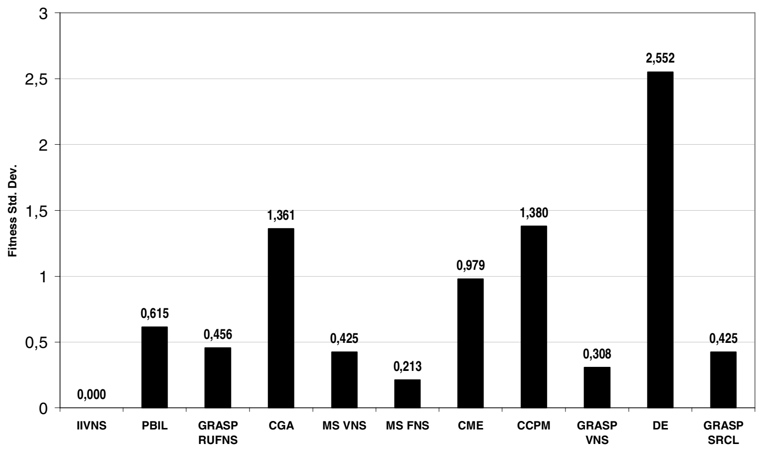


Fig. 6. Algorithmic reliability

possible to state that in a general manner, non population-based approaches have better computational effort measures, with the exception of DE.

The reliability of an algorithm refers to the extent of confidence for a given algorithm to achieve good results in any execution, tightly related to its average effectiveness. A commonly used measure is the standard deviation of the fitness of the average convergence point *P'*.

Fig. 6 shows, for each algorithm, the standard deviation on *P'*. The less reliable of the algorithms is DE, with a standard deviation of 1.72% (2.552) of the average fitness. We also observe that All GRASP and multi-start algorithms have a standard deviation under the 0.5 boundary. There is another interesting detail: the IIVNS algorithm, besides presenting itself as the most effective algorithm in the set, has a standard deviation of 0. This algorithm thus proves to achieve the maximal result in 100% of conducted experiences. Nothing can be deduced about its optimality, but its standard deviation entrusts a good interval of confidence at this matter.

Globally we can observe that the GRASP approaches are very fast but are also very deceptive upon local optima trapping. GRASP's Light LSs are unable to explore the search space in an effective way, although better results are noticed when the LS procedures incorporate heavier heuristics or additional flow mechanisms (as the GRASP RUFNS). The combinatorial complexity of real-world RND instances are not satisfactory for GRASP approaches that utterly emphasize the optimization itself, although these approaches can be used for interactive CAD tools since they deliver reasonable quality in a very short amount of time.

None of our population-based approaches suffered from premature convergence (except the DE stagnation phenomena [101]) since these bio-inspired models intrinsically use their control parameters to avoid such backdrops, but they usually also get to be slower than most of the contestants. On the other hand these approaches are easily powered-up through parallelism.

The IIVNS variant, besides delivering high quality results is a reasonably fast optimizer, standing very near to the GRASP echelon performance.

8 Characterization and RNDBench Quick Reference

This section delivers summarized information related to the run-time environment description, algorithmic characterization and our RND benchmark proposal. This information targets at quick referencing, when the reader is already familiar with the remaining subjects.

8.1 Heterogeneous Environments Characterization

Table 13. Heterogeneous experimental run-time environment specification table

Algo-rithm	Lang.	Run-time Platform (a)	Hardware	Execution Time (b)	Remarks
DE	C# (ECMA)	Windows + .Net Framework / Linux + Mono Framework	pool ranging from PIII 800 Mhz to Core2Duo 2.4 GHz	32m (P4 3.0GHz ref.)	CONDOR used with a 180 machine heterogon-ous hardware pool.
PBIL	C++ (ISO)	Windows + .Net Framework / GNU Linux or LinEx + KDe-velop	pool ranging from 1.8 Ghz to 3.2 GHz	2h30m	BOINC used with a 100 machine heterogeneous hardware pool.
CGA	Java	Windows 2003 r2 Server X64 + JRE 1.4	Dual Core AMD-Opteron Processor 2212 2.00GhZ, 4.00 GB RAM	4h-8h	Execution time is not accurate because several experiments were running in the same machine simultaneously.
CCPM	Java	Windows 2003 r2 Server X64 + JRE 1.4	Dual Core AMD-Opteron Processor 2212 2.00GhZ, 4.00 GB RAM	5h-9h	Execution time is not accurate because several experiments were running in the same machine simultaneously.
CME	Java	Windows 2003 r2 Server X64 + JRE 1.4	Dual Core AMD-Opteron Processor 2212 2.00GhZ, 4.00 GB RAM	5h-9h	Execution time is not accurate because several experiments were running in the same machine simultaneously.
GRASP RUFNS	C# (ECMA)	Windows + .Net Framework / Linux + Mono Framework	pool ranging from PIII 800 Mhz to Core2Duo 2.4 GHz	5h (P4 3.0GHz ref.)	CONDOR used with a 180 machine heterogon-ous hardware pool.
GRASP SRCL	C# (ECMA)	Windows + .Net Framework / Linux + Mono Framework	pool ranging from PIII 800 Mhz to Core2Duo 2.4 GHz	6h50m (P4 3.0GHz ref.)	CONDOR used with a 180 machine heterogon-ous hardware pool.
GRASP VNS	C# (ECMA)	Windows + .Net Framework / Linux + Mono Framework	pool ranging from PIII 800 Mhz to Core2Duo 2.4 GHz	1h30m (P4 3.0GHz ref.)	CONDOR used with a 180 machine heterogon-ous hardware pool.
IIVNS	C# (ECMA)	Windows + .Net Framework / Linux + Mono Framework	pool ranging from PIII 800 Mhz to Core2Duo 2.4 GHz	3h30m (P4 3.0GHz ref.)	CONDOR used with a 180 machine heterogon-ous hardware pool.
MS FNS	C# (ECMA)	Windows + .Net Framework / Linux + Mono Framework	pool ranging from PIII 800 Mhz to Core2Duo 2.4 GHz	4h20m (P4 3.0GHz ref.)	CONDOR used with a 180 machine heterogon-ous hardware pool.
MS VNS	C# (ECMA)	Windows + .Net Framework / Linux + Mono Framework	pool ranging from PIII 800 Mhz to Core2Duo 2.4 GHz	3h40m (P4 3.0GHz ref.)	CONDOR used with a 180 machine heterogon-ous hardware pool.

a) Includes OS
b) Approximate values

8.2 Algorithmic Characterization

Table 14. Algorithmic characterization table

Name	Trajec-tory	Population	Memory	Bio-Inspiration	Hybridization
DE	-	+	∃	+	-
PBIL	-	+	∃	+	-
CGA	-	+	∃	+	+
CCPM	-	+	-	+	+
CME	-	+	-	+	+
GRASP RUFNS	-	-	∃	-	+
GRASP SRCL	-	-	∃	-	-
GRASP VNS	∃	-	∃	-	+
IIVNS	∃	-	∃	∃	+
MS FNS	∃	-	+	-	-
MS VNS	∃	-	∃	-	-

+: feature present, ∃ :partially present, -: not present

8.3 RNDBench Reference

The RNDBench benchmark proposal stand upon four basic assessment axes tightly related to the overall performance indexes previously described in Section 7. The RNDBench ranking results from the relative normalized [0,1] weighted sum of its axes, defined as follows:

- EFFECT: normalized relative values obtained from the algorithm's effectiveness.
- CEFFORT/2: 50% weighted normalized relative values obtained from the algorithm's computing effort.
- REL: normalized relative values obtained from the algorithm's reliability.
- VAR/2: 50% weighted normalized relative values obtained from the algorithm's run-time variability.

Table 15 shows the RNDBench results based on the previous factors.

Table 15. RNDBench ranking table

Name	EFFECT	CEFFORT/2	REL	VAR/2	RNDBench
IIVNS	1,000	0,485	1,000	0,461	**2,946**
MS FNS	0,829	0,487	0,917	0,461	**2,694**
GRASP VNS	0,797	0,493	0,879	0,486	**2,655**
MS VNS	0,844	0,476	0,833	0,453	**2,606**
GRASP RUFNS	0,861	0,069	0,821	0,500	**2,252**
PBIL	0,876	0,158	0,759	0,176	**1,969**
CME	0,823	0,000	0,616	0,487	**1,927**
GRASP SRCL	0,000	0,499	0,833	0,496	**1,829**
CGA	0,844	0,013	0,467	0,306	**1,630**
CCPM	0,820	0,024	0,459	0,211	**1,515**
DE	0,037	0,494	0,000	0,494	**1,025**

i) Dashed lines delineate quartile boundaries

9 Conclusions and Future Works

In this work we have presented a multifaceted comparison of a wide range of algorithms. The fourteen different algorithms applied to solve the RND problem have followed two main principles: technology-independence and a normalized comparison through the definition of clock calibration routines. We have stated that the best results are yielded by the IIVNS variant, surpassing the results of the remaining contestants. Through this work, the foundations for a credible RND benchmark are proposed, although particular and specific additional problem instances are required. As future development we fold our research line in two main directions: a) the inclusion of multi-objective optimization methods in order to enlarge our optimization approach support base and b) the creation of additional extreme instances with landscape simulation features, including path loss models and bandwidth demand zones.

References

[1] Thain, D., Tannenbaum, T., Livny, M.: Distributed computing in practice: The Condor experience. Concurrency and Computation Practice and Experience 17, 323–356 (2005)

[2] Mendes, S.P., Gómez-Pulido, J.A., Vega-Rodríguez, M.A., Sánchez-Pérez, J.M.: A differential based algorithm to optimize the radio network design problem. In: Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing, p. 119 (2006)

[3] Vega-Rodríguez, M.A., Gómez-Pulido, J.A., Alba, E., Vega-Pérez, D., Mendes, S., Molina, G.: Different evolutionary approaches for selecting the optimal number and locations of omnidirectional BTS in a radio network. In: Proceedings of the 11th International Conference on Computer Aided Systems Theory (2007)

[4] Vega-Rodríguez, M.A., Gómez-Pulido, J.A., Alba, E., Vega-Pérez, D., Mendes, S.P., Molina, G.: Using omnidirectional BTS and different evolutionary approaches to solve the RND problem. In: Moreno Díaz, R., Pichler, F., Quesada Arencibia, A. (eds.) EUROCAST 2007. LNCS, vol. 4739, pp. 853–860. Springer, Heidelberg (2007)

[5] Mendes, S.P., Gómez-Pulido, J.A., Vega-Rodríguez, M.A., Pereira, A.M., Pérez, J.M.: Fast wide area network design optimisation using differential evolution. In: Proceedings of the International Conference on Advanced Engineering Computing and Applications in Sciences, pp. 3–10 (2007)

[6] Mendes, S.P., Domingues, P., Pereira, D., Vale, R., Gomez-Pulido, J.A., Silva, L.M., Vega-Rodríguez, M.A., Sánchez-Pérez, J.M.: Omni-directional RND optimisation using differential evolution: In-depth analysis via high throughput computing. In: Proceedings of EPIA (2007)

[7] Calegari, P., Guidec, F., Kuonen, P.: Combinatorial optimization algorithms for radio network planning. Journal of Theoretical Computer Science 263, 235–265 (2001)

[8] Calegari, P., Guidec, F., Kuonen, P., Kobler, D.: Parallel island-based genetic algorithm for radio network design. Journal of Parallel and Distributed Computing 47, 86–90 (1997)

[9] Alba, E.: Evolutionary algorithms for optimal placement of antennae in radio network design. In: Proceedings of the International Parallel and Distributed Processing Symposium (2004)

[10] Alba, E., Almeida, F., Blesa, M., Cotta, C., Díaz, M., Dorta, I., Gabarró, J., León, C., Luque, G., Petit, J., Rodríguez, C., Rojas, A., Xhafa, F.: Efficient parallel LAN/WAN algorithms for optimization: The MALLBA project. Parallel Computing 32, 415–440 (2006)

[11] Alba, E., Chicano, F.: On the behaviour of parallel genetic algorithms for optimal placement of antennae in telecommunications. International Journal of Foundations of Computer Science 16, 86–90 (2005)

[12] Celli, G., Costamagna, E., Fanni, A.: Genetic algorithms for telecommunication network optimization. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, pp. 1227–1232 (1995)

[13] Meunier, H., Talbi, E.G., Reininger, P.: A multiobjective genetic algorithm for radio network optimization. In: Proceedings of the Congress on Evolutionary Computation, pp. 317–324 (2000)

[14] Watanabe, S., Hiroyasu, T., Miki, M.: Parallel evolutionary multi-criterion optimization for mobile telecommunication networks optimization. In: Proceedings of Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems Conference, pp. 167–172 (2001)

[15] Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 416–423 (1993)

[16] Talbi, E.G., Cahon, S., Melab, N.: Designing cellular networks using a parallel hybrid metaheuristic on the computational grid. Computer Communications 30, 698–713 (2007)

[17] Calégari, P., Guidec, F., Kuonen, P., Chamaret, B., Ubéda, S., Josselin, S., Wagner, D., Pizarosso, M.: Radio network planning with combinatorial optimization algorithms. Theoretical Computer Science 263, 235–265 (2001)

[18] Chamaret, B., Josselin, S., Kuonen, P., Pizarroso, M., Salas-Manzanedo, B., Ubeda, S., Wagner, D.: Radio network optimization with maximum independent set search. In: Proceedings of the 47th IEEE Vehicular Technology Conference, pp. 770–774 (1997)

[19] He, J., Verstak, A., Watson, L., Rappaport, T., Anderson, C., Ramakrishnan, N., Shaffer, C., Tranter, W., Bae, K., Jiang, J.: Global optimization of transmitter placement in wireless communication systems. In: Proceedings of the High Performance Computing Symposium, pp. 328–333 (2002)

[20] Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. Journal of Optimization Theory and Applications 79, 157–181 (1993)

[21] Vasquez, M., Hao, J.K.: A heuristic approach for antenna positioning in cellular networks. Journal of Heuristics 7, 443–472 (2001)

[22] Elkamchouchi, H.M., Elragal, H.M., Makar, M.A.: Cellular radio network planning using particle swarm optimization. In: Proceedings of the Radio Science Conference, pp. 1–8 (2007)

[23] Tutschku, K.: Demand-based radio network planning of cellular mobile communication systems. University of Wurzburg Research Report Series, 177 (1997)

[24] Church, R.L., ReVelle, C.: The maximal covering location problem. Regional Science 30, 101–118 (1974)

[25] Ibbetson, L.J., Lopes, J.B.: An automatic base site placement algorithm. In: Proceedings of the 47th IEEE Vehicular Technology Conference, pp. 760–764 (1997)

[26] Fritsch, T., Hanshans, S.: An integrated approach to cellular mobile communication planning using traffic data prestructured by a self-organizing feature map. In: Proceedings of the EEE International Conference on Neural Networks, pp. 822D–822I (1993)

[27] Nebro, A.J., Alba, E., Molina, G., Chicano, F., Luna, F., Durillo, J.J.: Optimal antenna placement using a new multi-objective CHC algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 876–883 (2007)

[28] Mendes, S., Domingues, P., Vale, R., Pereira, D., Gomez-Pulido, J.A., Silva, L.M., Vega-Rodríguez, M.A., Sánchez-Pérez, J.M.: Omni-directional RND optimisation using differential evolution: In-depth analysis via high throughput computing. In: Proceedings of the Portuguese Conference on Artificial Intelligence (2007)

[29] Donninger, C., Kure, A., Lorenz, U.: Parallel Brutus: the first distributed, FPGA accelerated chess program. In: Proceedings of the 18th International Parallel and Distributed Processing Symposium, p. 44 (2004)

[30] Eklund, S.E.: Time series forecasting using massively parallel genetic programming. In: Proceedings of the 17th International Parallel and Distributed Processing Symposium, p. 143.1 (2003)

[31] Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A.E., Purcell, T.J.: A survey of general-purpose computation on graphics hardware. Computer Graphics Forum 26, 80–113 (2007)

[32] Fok, K., Wong, T., Man-Leung, J.: Evolutionary computing on consumer graphics hardware. IEEE Intelligent Systems 22, 69–78 (2007)

[33] Langdon, W.B., Banzhaf, W.: A SIMD interpreter for genetic programming on GPU graphics cards. In: Proceedings of the European Genetic Programming Conference (2008)

[34] NVIDIA Corporation, Cuda Zone (2008), `http://www.nvidia.com/object/cuda_home.html`

[35] Garey, M., Johnson, D.: Computers and intractability: A guide to the theory of NP-completeness. Freeman and Co., New York (1979)

[36] Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMUCS, Carnegie Mellon University, pp. 94–163 (1994)

[37] Baluja, S., Caruana, R.: Removing the genetics from the standard genetic algorithm. In: Proceedings of the Twelfth International Conference on Machine Learning (1995)

[38] Yang, S.Y., Ho, S.L., Ni, G.Z., Machado, J.M., Wong, K.F.: A new implementation of population based incremental learning method for optimizations in electromagnetics. IEEE Transactions on Magnetics 43, 1601–1604 (2007)

[39] Bureerat, S., Sriworamas, K.: Population-based incremental learning for multiobjective optimisation. Soft Computing in Industrial Applications, Advances in Soft Computing 39, 223–232 (2007)

[40] Hong, Y., Kwong, S., Chang, Y., Ren, Q.: Clustering ensembles guided unsupervised feature selection using population based incremental learning algorithm. Pattern Recognition 41(9), 2742–2756 (2009)

[41] Jelodar, M.S., Fakhraie, S.M., Ahmadabadi, M.N.: A new approach for training of artificial neural networks using population based incremental learning (PBIL). In: Proceedings of the International Conference on Computational Intelligence, pp. 165–168 (2004)

[42] Chiang, F., Braun, R.: Towards a management paradigm with a constrained benchmark for autonomic communications. In: Wang, Y., Cheung, Y.-m., Liu, H. (eds.) CIS 2006. LNCS, vol. 4456, pp. 250–258. Springer, Heidelberg (2007)

[43] Domínguez-González, D., Chaves-González, J.M., Vega-Rodríguez, M.A., Gómez-Pulido, J.A., Sánchez-Pérez, J.M.: Using PBIL for solving a real-world frequency assignment problem in GSM networks. New Trends in Artificial Intelligence, 207–218 (2007)

[44] Papadimitriou, G.I., Obaidat, M.S., Pomportsis, A.S.: On the use of population-based incremental learning in the medium access control of broadcast communication systems. In: Proceedings of the 10th IEEE International Conference on Electronics, Circuits and Systems, pp. 1260–1263 (2003)

[45] Kendall, R., Braun, R.: Digital communication filter design by stochastic optimization. In: Workshop on the Applications of Radio Science (2002)

[46] Price, K., Storn, R.: Differential evolution – a simple evolution strategy for fast optimisation. Dr. Dobb's Journal 22, 18–24 (1997)

[47] Price, K., Storn, R.: Web site of DE (2006), `http://www.ICSI.Berkeley.edu/~storn/code.html` (accessed July 1, 2006)

[48] Joshi, R., Sanderson, A.: Minimal representation multisensor fusion using differential evolution. In: Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, p. 266 (1997)

[49] Vasan, A., Raju, K.: Optimal reservoir operation using differential evolution. In: Proceedings of International Conference on Hydraulic Engineering: Research and Practice (2004)

[50] Abbass, H.A., Sarker, R.: The Pareto differential evolution algorithm. International Journal on Artificial Intelligence Tools 11, 531–552 (2002)

[51] Storn, R., Price, K.: A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, The University of California, Berkley (1995)
[52] Lampinen, J., Zelinka, I.: Mixed variable non-linear optimization by differential evolution. In: Proceedings of the 2nd International Prediction Conference, pp. 45–55 (1999)
[53] Aragão, M.P., Ribeiro, C.C., Uchoa, E., Werneck, R.F.: Hybrid local search for the Steiner problem in graphs. In: Proceedings of the 4th Metaheuristics International Conference (2001)
[54] Festa, P., Resende, M.G.: An annotated bibliography of GRASP. Technical Report TD-5WYSEW, AT&T Labs Research (2004)
[55] Mavridou, T., Pardalos, P.M., Pitsoulis, L.S., Resende, M.G.: A GRASP for the biquadratic assignment problem. European Journal of Operational Research 105, 613–621 (1998)
[56] Martins, S.L., Resende, M.G., Ribeiro, C.C., Pardalos, P.M.: A parallel GRASP for the Steiner tree problem in graphs using a hybrid local search strategy. Journal of Global Optimization 17, 267–283 (2000)
[57] Pardalos, P.M., Qian, T., Resende, M.G.: A greedy randomized adaptive search procedure for the feedback vertex set problem. Journal of Combinatorial Optimization 2, 399–412 (1999)
[58] Rosseti, R., Aragão, M.P., Ribeiro, C.C., Uchoa, E., Werneck, R.F.: New benchmarck instances for the Steiner problem in graphs. In: Proceedings of the 4th Metaheuristics International Conference (2001)
[59] Resende, M.G.: Computing approximate solutions of the maximum covering problem using GRASP. Journal of Heuristics 4, 161–171 (1998)
[60] Resende, M.G., Ribeiro, C.C.: A GRASP for graph planarization. Journal of Heuristics 4, 171–181 (1998)
[61] Ribeiro, C.C., Uchoa, E., Werneck, R.F.: A hybrid GRASP with perturbations for the Steiner problem in graphs. INFORMS Journal on Computing 14, 228–246 (2002)
[62] Resende, M.G., Ribeiro, C.C.: Greedy randomized adaptive search procedures. Technical Report TD-53RSJY, AT&T Labs Research (2002)
[63] Feo, T.A., Resende, M.G.: A probabilistic heuristic for a computationally difficult set covering problem. Operation Research Letters 8, 67–71 (1989)
[64] Feo, T.A., Resende, M.G.: Greedy randomized adaptive search procedures. Journal of Global Optimization 6, 109–133 (1995)
[65] Mladenovic, N., Hansen, P.: Variable neighbourhood search. Computers and Operations Research 24, 1097–1100 (1997)
[66] Hansen, P., Mladenovic, N.: Variable neighbourhood search: Principles and applications. European Journal of Operational Research 130, 449–467 (2001)
[67] Hansen, P., Mladenovic, N.: A Tutorial on variable neighborhood search. Technical Report - GERAD and Mathematical Institute, SANU, Belgrade (2003)
[68] Fleszar, K., Hindi, K.S.: New heuristics for one-dimensional bin-packing. Computers and Operations Research 29, 821–839 (2002)
[69] Hansen, P., Mladenovic, N., Dragan, U.: Variable neighborhood search for the maximum clique. Discrete Applied Mathematics 145, 117–125 (2004)
[70] Liberti, L., Draži, M.: Variable neighbourhood search for the global optimization of constrained NLPs. In: Proceedings of Global Optimization, pp. 1–5 (2005)

[71] Burke, E.K., Cowling, P., Keuthen, R.: Implementation report: Variable neighbourhood search. Technical Report - University of Nottingham, NG8 1BB (2000)
[72] Avanthay, C., Hertz, A., Zufferey, N.: A variable neighborhood search for graph coloring. European Journal of Operational Research 151, 379–388 (2003)
[73] Galinier, P., Hao, J.K.: Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization 3, 379–397 (1999)
[74] Caporossi, G., Hansen, P.: Variable neighborhood search for extremal graphs: Three ways to automate finding conjectures. Discrete Mathematics 276, 81–94 (2004)
[75] Pérez, J.A., Moreno-Vega, J.M., Martín, I.R.: Variable neighbourhood tabu search and its application to the median cycle problem. European Journal of Operational Research 151, 365–378 (2003)
[76] Fleszar, K., Hindi, K.H.: Solving the resource-constrained project scheduling problem by a variable neighbourhood search. European Journal of Operational Research 155, 402–413 (2004)
[77] Burke, E.K., Causmaecker, P.D., Petrovic, S., Berghe, G.V.: Variable neighbourhood search for nurse rostering problems. In: Resende, M.G., Sousa, J.P. (eds.) Metaheuristics: Computer Decision-Making, pp. 153–172. Kluwer, Norwell (2003)
[78] Pérez, J.A., Mladenovic, N., Batista, B.M., Amo, I.J.: Variable Neighbourhood Search. Springer, New York (2006)
[79] Lusa, A., Potts, C.N.: A variable neighbourhood search algorithm for the constrained task allocation problem. Journal of the Operational Research Society 59(6), 812–822 (2007)
[80] Lejeune, M.A.: A variable neighborhood decomposition search method for supply chain management planning problems. European Journal of Operational Research 175(2), 959–976 (2006)
[81] Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
[82] Buffa, E.S., Armour, G.C., Vollmann, T.E.: Allocating facilities with CRAFT. Harvard Business Review, 136–158 (1964)
[83] Saez, Y., Isasi, P., Segovia, J., Hernandez, J.C.: Reference chromosome to overcome user fatigue in IEC. New Generation Computing 23, 129–142 (2005)
[84] Saez, Y., Isasi, P., Segovia, J.: Interactive evolutionary computation algorithms applied to solve Rastrigin test functions. In: Proceedings of the 4th IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology, pp. 682–691 (2005)
[85] Dawkins, R.: The Selfish Gene. Oxford University Press, New York (1976)
[86] Gómez-Pulido, J.: Web site of Net-Centric Optimization (OPLINK:UNEX) (2006), http://oplink.unex.es/rnd (accessed April 1, 2008)
[87] Mayer, U.: NBenchProject (2007), `http://www.tux.org/~mayer/linux/` (accessed December 1, 2007)
[88] Anderson, D.P.: BOINC (2007), `http://boinc.berkeley.edu` (accessed June 1, 2008)
[89] Anderson, D.P.: BOINC: A system for public-Resource computing and storage. In: Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, pp. 4–10 (2004)
[90] RND@home (2008), `http://arcoboinc.unex.es/`rnd (accessed March 1, 2008)
[91] LinEx (2008), `http://www.linex.org` (accessed March 1, 2008)

[92] Buell, D., El-Ghazawi, T., Gaj, K., Kindratenko, V.: High-performance reconfigurable computting. Computer 40, 23–27 (2007)
[93] Vega-Rodríguez, M.A., Sánchez-Pérez, J.M., Gómez-Pulido, J.A.: Guest editors' introduction - special issue on FPGAs: Applications and designs. Microprocessors and Microsystems 28, 193–196 (2004)
[94] Hsiao, J.M., Tsai, C.J.: Analysis of an SOC architecture for MPEG reconfigurable video coding framework. In: Proceedings of the IEEE International Symposium on Circuits and Systems, pp. 761–764 (2007)
[95] Gómez-Pulido, J.A., Vega-Rodríguez, M.A., Pérez, J.M., Mendes, S.P.: Diseño y prototipado de un processador para el cálculo de la cobertura en el diseño de redes de radiocomunicaciones. In: VII Jornadas de Computación Reconfigurable y Aplicaciones (2007)
[96] Alba, E., Molina, G., Chicano, F.: Optimal placement of antennae using metaheuristics. In: Boyanov, T., Dimova, S., Georgiev, K., Nikolov, G. (eds.) NMA 2006. LNCS, vol. 4310, pp. 214–222. Springer, Heidelberg (2007)
[97] Alba, E., Cotta, C., Chicano, F., Nebro, A.J.: Parallel evolutionary algorithms in telecommunications: Two case studies. In: Proceedings of Congreso Argentino de Ciencias de la Computación (2002)
[98] Xilinx Inc., Xilinx (2008), `http://www.xilinx.com` (accessed March 1, 2008)
[99] Xess Corporation (200) Xess, `http://www.xess.com` (accessed March 1, 2008)
[100] Barr, R.S., Golden, B.L., Kelly, J.P., Resende, M.G., Stewart, W.: Designing and reporting on computational experiments with heuristic methods. Journal of Heuristics 1, 9–32 (1996)
[101] Lampinen, J., Zelinka, I.: On stagnation of the differential evolution algorithm. In: Proceedings of the 6th International Mendel Conference on Soft Computing, pp. 76–83 (2000)

Strategies for Decentralised Balancing Power

Andreas Kamper[1] and Anke Eßer[2]

Abstract There are many different approaches to central load management in power supply systems, such as direct load control or price signals to control production and consumption. Despite these measures there will always be an imbalance between production and consumption, i.e. due to fluctuating resource availabilities and unforeseen changes in consumption. As CO_2 emissions and sustainable electricity production have entered the focus of attention in politics and industries, ecologically advantageous concepts avoiding inefficiencies in power supply are strongly promoted. In this article, a self-organising approach to small devices such as freezers or washing machines as well as Combined Heat and Power plants (CHP) is presented, which aims at avoiding imbalances in the power network. While each device has its own constraints and specific task (e.g. provide heat or wash clothes), most of them have a limited degree of freedom in their schedules. A P2P approach with an Evolutionary Algorithm in combination with a local search is used to identify suitable partners to cover their production or consumption and thus to adjust the load in a way to minimise network imbalances.

Karlsruher Institute of Technology
Institute AIFB
76128 Karlsruhe
Germany
`aka@aifb.uni-karlsruhe.de`
Karlsruher Institute of Technology
Institute IIP
76128 Karlsruhe
Germany
`Anke.Esser@wiwi.uni-karlsruhe.de`

1 Introduction

Due to the increasing cost of energy carriers, new approaches to a more efficient use of the available resources are gaining importance. In particular, strategies are sought to efficiently manage the permanently occurring imbalances between production and consumption. These imbalances are, for instance, caused by inevitable differences between the expected energy usage and the actual consumption, or through fluctuating resource availabilities. They cause malfunctions on or damages to electrical appliances.

To balance consumption and production, an effective but expensive system of so-called balancing power markets has been installed. Imbalances occurring at lower voltage levels (i.e. in balancing groups) are summed up and balanced on the highest voltage level. The balancing group on lower voltage levels is charged the incurring costs. The amount of balancing power needed is provided by special power plants. The capacity restrictions for power plants participating in balancing power markets are rather high. They can only be met for larger power plants. In the last few years several balancing power pools were created, in which smaller power plants and larger responsive loads pool to jointly reach the critical size in order to participate in balancing power provision. Smaller loads like home appliances cannot be pooled efficiently. However, even these small devices can be used to reduce imbalances and the balancing costs on lower voltage levels resulting from this.

In the following sections a decentralised P2P approach to pool small home appliances and privately owned Combined Heat and Power (CHP) plants is introduced. Thus, they can be used to reduce the amount of balancing power needed in balancing groups. The devices and CHP plants belong to private households and have to meet their owners demands. Only the remaining degree of freedom is used to balance the consumption and production. For the optimisation, an Evolutionary Algorithm in combination with a local search is used to group the devices so that existing imbalances are reduced. This decentralised approach has to be integrated into the existing system of balancing power, and the corresponding restrictions must be met.

Therefore, sections 2 and 3 will first give a brief introduction of the German power supply system, balancing power as well as the German power markets and balancing power markets. In section 4, existing approaches to decentralised balancing pools on a larger scale are shown. In section 5, a new pooling approach with the Evolutionary Algorithm is introduced. The chapter closes with a conclusion.

2 The German Power Supply System

Today, electricity is used in almost all processes of industrial production as well as for most household appliances. With about 20% it is the third most important final energy carrier in Germany [12]. To supply end consumers with electricity, complex network structures have been built. These so-called electricity supply systems consist of power sources (power plants) and power sinks (consumers) as well as power lines to transport the electricity from power plants to consumers. In the following, the German power supply system is described.

2.1 Power Generation in Germany

The driving force in power supply systems is power demand. In 2005, the total power demand to be met in Germany amounted to 518 GWh. It is characterised by cyclical variations[1] and short-term stochastic fluctuations. Since it is not economically justifiable to store large amounts of electricity, power has to be produced at the time it is consumed. This characteristic of power supply affects the structure of power generation capacities as well as the organisation of the power markets on wholesale level.

Traditionally, power generation in Germany is based on fossil fuels and uranium. In 2005, a net electrical capacity of 124 GW was installed. Thereof 76 GW were conventional thermal power plants, 20 GW nuclear stations, 18 GW windmills, and 8 GW hydro power stations [10]. Due to the nuclear phase out as well as the replacement of old conventional thermal plants, 40-50 GW of electrical capacity have to be built until 2020 [15]. Assuming a continuousness of today's energy and climate policy, it is most likely that the share of RES fuelled plants as well as Combined Heat and Power (CHP) plants will increase. Regarding large conventional power stations, gas power plants are most likely to be built [21].

2.2 Power Markets

On wholesale level, power supply companies arrange schedules to the satisfaction of the anticipated power demand of their customers up to one day in advance. The required power is obtained via bilateral trading[2] or on power exchanges. In bilateral trades, the design characteristics of the contracts, such as price, volume, or place and time of delivery, are negotiated between the two contracting parties. By contrast, at power exchanges standardised contracts are traded, guaranteeing a higher degree of transparency. In Germany, just under 20% of the daily demand is traded on power exchanges [25], while 80% are traded bilaterally. In addition, power trading is used for speculation or to hedge against the risks of energy supply, such as price fluctuations or unforeseen changes in the demand structure. Based on the time lags between closing day and delivery, principally three markets for scheduled energy can be distinguished. On spot markets, power contracts with (physical) fulfilment on the subsequent day are traded. Contracts whose fulfilments lie weeks or months ahead are traded future markets. Power markets for short-dated power trading are called

[1] Power demand follows seasonal, weekly, and daily patterns [28].

[2] Bilateral trading is also called Over-The-Counter (OTC) trading.

intra-day markets. In Germany they are realized on the European Energy Exchange (EEX).

In 2008, there were more than 900 energy suppliers supplying end-customers with electricity in Germany. They comprise distribution companies of power producers, pure redistributors, and niche players, such as green electricity suppliers.

2.3 Power Transmission and Distribution in Germany

2.3.1 Organisation of Power System Networks

Power transmission and distribution networks conduce to transfer electric energy from power plants to the end consumer. The German transmission network, which comprises the voltage levels of 220 kV and 380 kV, is part of the UCTE[3]-network that links up the national networks in continental Europe. It is divided into four control areas operated by E.ON Netz GmbH, Vattenfall Europe Transmission GmbH, RWE Transportnetz Strom GmbH, and EnBW Transportnetze AG. The distribution networks, which comprise the voltage levels of 110 kV to 230 V, are operated by so-called distribution system operators. In contrast to only four TSO, there are more than 900 distribution system operators in Germany.

The TSO are responsible for the construction, maintenance, and operation of the transmission system infrastructure. In particular, both the system reliability and the quality of supply have to be guaranteed. Concerning system reliability, the number of outages has to be minimised. Regarding quality of supply, above all, a constant network frequency is required. The nominal value of network frequency in the UCTE network is 50 Hz. Deviations of the network frequency from the nominal value cause malfunctions or damages on electrical appliances. Frequency deviations are mainly caused by variations in power production and consumption. If less electrical energy is fed into the grid than withdrawn on the consumer side, the frequency drops below the nominal value of 50 Hz. If more electricity is supplied than withdrawn, the network frequency rises above the nominal value.

2.3.2 Balancing of Power System Networks

Due to demand variations, forecasting errors or unforeseen events, demand and supply are virtually always in disequilibrium. Therefore, extra generating capacity for balancing power production and consumption on very short notice is needed. Extra generation capacity, which is made available by increasing the load of already operating power plants, is called spinning reserve. By contrast, power plants, such as pumped storage plants which have to be brought online to supply balancing power, are called non-spinning reserve. Germany distinguishes between three types of

[3] Union for the Co-ordination and Transmission of Electricity.

balancing (or control) power (cp. Figure 1): primary control, secondary control, and minutes reserve. Primary control arises from the self-adjusting reaction of power plants (active contribution) and frequency dependent loads (passive contributions) to frequency deviations. In the UCTE network the Transmission System Operators (TSO) are collectively responsible for the provision of primary control. In total, a spinning reserve of 3000 MW has to be held back, which is distributed among the TSO proportionately to their share in total power production. The UCTE defines secondary control as the *"instructed action of particular generating sets linked to a control loop in a control area, to move the overall system (frequency and interchange) deviation of the control area toward zero following the delivery of primary control in response to a sudden variation in production or consumption."* [9] Like primary control, it is activated automatically. Minutes reserve (or tertiary control) is activated after 15 minutes at the latest in case of plant losses or major forecasting errors. Unlike primary and secondary control, it is not automatically delivered but instructed by the TSO.

The amounts of secondary control and minutes reserve to be held in a control area are determined by the TSO. The sum of secondary control and minutes reserve must be at least equal to the capacity of the largest power plant in the control area. After one hour, secondary control and minute reserve are replaced by the hour (or replacement) reserve. It is not held ready by the TSO but by the various balancing groups (BG). The term balancing group relates to groups of power suppliers and consumers in liberalised markets which are pooled, in particular, for accounting purposes.

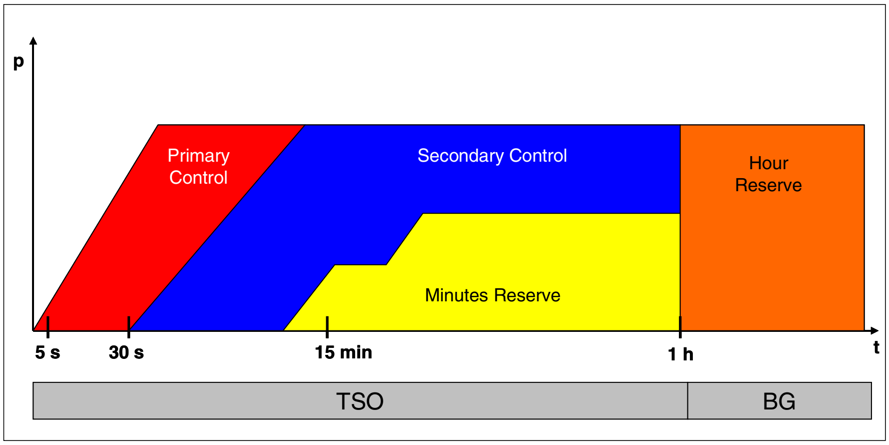


Fig. 1 Types of balancing power in Germany

The TSO procure their balancing power by tendering. In Germany, there is a separate tendering procedure for each type of balancing power. However, since 1.12.2007 the four German TSOs have to maintain a common tendering platform for primary control and secondary control. For minutes reserve, it has already been in operation since 1.12.2006 (www.regelleistung.net). To participate in the tendering procedures, power plant operators have to pre-qualify. In particular, technical requirements regarding the activation speed and availability of the units have to be fulfilled. The billing of balancing power is regulated in para. 8 StromNZV[4] [26]. While the balancing power is provided by the TSO, it is needed to countervail balance variations of balancing groups. Hence, it is cleared at the expenses of the balancing groups. Within a control area, excesses and deficits of the balancing groups occur at the same time and even, to some extent, outside the control area. The remaining difference is balanced by the TSO. For settling, a uniform price for positive (or negative) balancing power is determined for each quarter of an hour. Balancing groups with a power surplus are paid this price for variations, while balancing groups with a power deficit are charged this price [26]. Hence, deviations from the announced schedule can be very costly for the balancing groups. Thus, reducing those variations by means of decentralised balancing seems a promising way of reducing the costs of power supply.

2.4 Future Developments

The increasing use of RES and CHP will lead to a further decentralisation of the power supply system in Germany. If the national targets for power generation from RES are to be met, the RES-based power production will have to exceed 240 TWh per year in 2020. Therefore, 67 GW of renewable capacity would have to be installed. In particular, wind energy would have to contribute with 25 GW of onshore and 16 GW of offshore capacity to reach the target. In addition to wind energy, the whole potential of biogenic energy carriers and geothermal energy would need to be realised [23].

Today, the share of CHP in total power production amounts to approximately 10%. Until 2020, the federal government wants to increase that share to 25%. Considering an economic electrical potential of more than 350 TWh/a this seems accomplishable [32] [7]. The realisation of this potential would lead to a further decentralisation of the energy systems. For even though almost 70% of the potential relates to district heating CHP, compared to large central power stations they can still be regarded as local units.

The ongoing decentralisation of energy systems will influence the development of transmission as well as the development of distribution networks. On the transmission level, especially the concentration of wind power in the north, far away from large consumption centres, will necessitate an extension of transmission capacity, in particular in north-south direction [5].[5] Regarding distribution networks, the connection of a multitude of distributed generation capacities demands a more active distribution grid management. Currently distribution networks in decentralised

[4] Stromnetzzugangsverordnung - Power Network Access Ordinance.

[5] A further reason necessitating the reinforcement of the German transmission system are the increasing inter-regional power transfers [10].

energy systems are semi-active networks, in which Decentralised Energy Management (DEM), load shedding and the disconnection of RES units is possible to some extent. In the future more and more so-called microgrids are most likely to emerge. These active network systems allow a more efficient, secure, and reliable regional energy supply. In addition, they are to enable bi-directional power exchanges as well as islanding and autonomous operation [6]. In addition, assuming that innovative Information and Communication Technologies (ICT) will be installed, the integration of consumers into power markets is possible. To establish such microgrids, the net infrastructure of distribution grids has to be significantly reinforced. Above all, control and protection technology has to be added. In addition, qualifications to allow for the collection and analysis of network data and thus, to allow for an efficient decentralised energy management, have to be set up. That way, the preconditions for the decentralised provision of balancing power would also be set.

3 Approaches to Decentralised Balancing

It is often claimed that responsive loads and distributed generation units should be enabled to provide balancing power. The model used mainly for the decentralised provision of balancing power are balancing power pools. In general, such power pools consist of small power plants and responsive loads. By pooling, they reach the critical size to participate in the balancing power market.[6] Distributed generation facilities and loads are particularly suitable to provide secondary control and minutes reserve [30].

The main precondition for the formation of a balancing power pool is that all participating units are information-technologically integrated. In particular, a bidirectional communication should be possible. Participating power plants have to guarantee short start-up times. In addition, no further restrictions such as heat demands should apply. With regard to responsive loads, only (industrial) processes allowing interruptions of up to 60 minutes are deployable. Therefore, particularly devices with storage possibilities, such as cooling devices, seem most suitable.

In addition, Virtual Power Plants (VPP) are discussed in relation to decentralised balancing of power system networks. They can be defined as an interactive, centrally controlled network of distributed generation units [3]. In its composition they are adapted to the load profile. By means of continuous optimisation and control, its production can be adapted to the aims of the operator of the VPP. For example, peak load demands or schedule variations can be reduced. Thus, the VPP can be used to reduce the balancing power demand of balancing groups. In contrast to balancing pools, the aim of the integration of the units is not to reach the critical size to act on balancing power markets but to attain the means that reduce the amount of balancing power which is needed by the balancing group.

[6] The pooling of the capacities is necessary due to the capacity requirements of 15 MW in Germany.

All distributed generation units forming a VPP have to be information- technological integrated. In addition, they should at least be installed in the same control area, best, in the area of the same balancing group. Regarding the ownership, either ownership of an energy supply company or private ownership is possible. If the generation unit(s) are privately owned, but controlled by the power supply company (or balancing group manager), incentive or remuneration schemes for the participation in the VPP have to be developed. Among others, profit sharing approaches or incentives like more beneficial conditions for gas supply seem promising.

3.1 Existing Balancing Power Pools and Virtual Power Plants

To date, there are only few examples of balancing power pools in Europe. The most prominent German power pool is the "Virtuelle Regelkraftwerk" of Steag[7]. The Steag balancing power pool is open for power plants and responsive loads that fulfill the following requirements [24]:

- the capacity made available has to be at least 1 MW,
- hourly availability,
- the temporal availability has to be at least 95%,
- the response time has to be less than 7 minutes, and
- the reserve power has to be available for at least 4 hours.

In 2008, the Steag balancing power pool had a size of 400 MW. A similar balancing pool is to be realised by a German engineering consultancy. It will be designed to operate in the control area of E.ON. In this power pool, units of a size of more than 100 kW will be able to participate [17].

Moreover, the Norwegian TSO Statnett[8] runs a market for balancing energy, called Regulation Capacity Option Markte (RCOM)[9]. Its main purpose is to guarantee sufficient operating reserve in winter peak load times [3].

Concerning VPPs, so far, a varying number of research projects have been undertaken in Germany [1]. In most of the projects, the main research focus is the optimisation of power generation and network utilisation. However, most of these VPP merely work on laboratory scale. The only economically operating VPP is the VK Unna, which is operated by the utilities of Unna [13]. It integrates five CHP plants, one wind park, one gas turbine, and one run-on-river hydro power station. In total, it covers 14% of the power demand within the balancing group of the utilities of Unna. It is controlled by a central control station, in which the operation modes of the power plants participating in the VPP are optimised every quarter of an hour. So far, principally positive results have been achieved. Most notably, schedule variations could be reduced to 2%. In addition, the purchasing portfolio could

[7] Since September 2007, Steag belongs to Evonik Industries.

[8] www.statnett.no

[9] Norwegian: Regulerkraftopsjonsmarkedet (RKOM).

be optimised, in particular the share of proprietary generation was increased. Moreover, the maximum peak load could be reduced by 5 MW and transmission losses were minimised.

3.2 *Available Potential for Decentralised System Balancing in Germany*

The available potential for decentralised balancing power provision depends on the availability of responsive loads as well as of small or micro generation units, such as CHP plants. However, due to data availability problems, only rough estimations for some industries and households can be made.

For some industries, the provision of positive secondary control and minutes reserve can be economically beneficial. In particular, cooling technologies as well as air conditioning systems can be deployed. In addition, industrial CHP plants can be used. According to Auer et al. [2], the total demand of minutes reserve could be met with only 60% of all air conditioning systems in German trades and industries.[10] At the University of Karlsruhe, the potential for selected sectors to provide secondary control and minutes reserve has been determined. The results are summarised in table 3.2. According to this, especially in the food industry, a high potential for the provision of balancing power exist. This is above all due to the high number of cooling or heating devices.

Table 1 Potentials for the provision of balancing power in selected industries [20]

	Secondary control	**Minutes reserve**
Metalworking industry	1011 MW	(-)
Plastics industry	(-)	2,358 MW
Food industry	2,739 MW	10,600 MW

Like industries, households and small businesses can provide balancing power by means of responsive loads. Condition precedent is that the operation of the corresponding household appliances and electrical devices is either dispensable or relocatable. This applies to the household appliances listed in table 2. In total, these appliances account for 54% of the total power demand of German households. Considering those appliances, Auer et al. calculated an average load shifting potential of 15,000 MW in summer and of 20,000 in winter [2] [18]. Similar results can be deduced from the results obtained in a simulation study conducted at the University of Karlsruhe [8].

In addition, micro power generation units installed in households could participate in decentralised balancing. Concerning conventional units, only combustion engines (CHP units and emergency generators), steam engines, micro gas turbines,

[10] It was assumed a realistic availability of 90% [2] [18].

Table 2 Power demand of German household appliances in 2000 [29]

Appliance	Yearly Consumption [bn kWh]	Proportion [%]
Hot water generation	15.0	13
Refrigerator	11.6	11
Freezer	11.2	10
Illumination	7.7	8
Washing machine	3.6	4
Dish washer	3.5	3
Dryer	3.2	3
Electric heating devices	2.0	2
Total	57.8	54

and Polymerelektroytmembran Fuel Cells (PEFC) fulfil the technical requirements of balancing power provision. However, due to the small size of the units, a large number of units has to be pooled to meet the prequalification criteria.

Today, only about 60 MW of micro-CHPs [31] and only about 30 MW of micro gas turbines [14] are installed in Germany. Stirling engines as well as fuel cells are available only on laboratory scale. Thus, at most one balancing power pool could be created with the existing units. Considering future potentials, EWI et al. estimated that until 2030, the total potential for micro fuel cells in households' power and heat supply will add up to between 3 GW and 15 GW [11]. However, they assume that the capacity actually constructed will be in the range of 3 GW.

To subsume a pooling of distributed generation and/or household units to act on the German balancing power market does not seem very promising. Therefore, other approaches, such as virtual power plants for decentralised balancing, should be considered.

One problem of virtual power plants is that a central control unit is needed, and the controlled power plants have to reserve at least a fraction of their maximum power for the VPP. Operating a central control unit can be very expensive, especially if there are many small power plants or sheddable loads. Private CHP plants and home appliances cannot be controlled efficiently by a central unit because these devices have to fulfil specific tasks for their owners and consume and produce energy at will. In section 4 a decentralised approach to these devices is shown which does not need a central control unit.

4 Decentralised Balancing Power Pools

In this chapter, a new strategy is introduced to show how pools of combined heat and power plants and home appliances like washing machines and refrigerators can be used to reduce the amount of balancing power. In comparison to the balancing power pools, consisting of larger power plants presented in section 3.1, the devices and CHP plants in this pool cannot be used to form a new balancing power pool. First, the number of appliances needed to meet the requirements of 30MW to provide secondary balancing power or 1 MW by one appliance to take part in a balancing pool is much too large. Furthermore, the appliances cannot be used exclusively to provide balancing power because they have different restrictions (i.e. temperature for a refrigerator) which are more important. Whereas a balancing power pool will only be used if there is an imbalance in the power grid, these small devices will consume or produce energy according to their restrictions. Nevertheless, even these small appliances and CHP plants can help to increase the efficiency by reducing the need for balancing power.

In order to use a pool of devices in a similar way as one big power plant, the restrictions of the devices have to be hidden inside the pool. The device pool has to act as one big unit which can produce or consume energy as needed. If a device has to produce or consume energy because of its restrictions, the other devices should consume or produce the equivalent so that the energy usage of the whole pool is in balance. The devices in the pool have to find suitable partners to cover their energy needs. The remaining degree of freedom can then be used to reduce imbalances in the network.

The main idea is to group some devices in the pool so that their resulting load curve matches the energy usage of other devices. This can be done by rescheduling the devices or by adjusting their load. The resulting optimisation problem is a variation of the NP-complete knapsack problem [16] [19] [4]. The knapsack problem has been studied for a long time in operations research, management science and computer science. In the classical 0/1 knapsack problem a set of items has to be chosen to get the highest profit. Every item has a profit p_ij and a weight w_ij. All chosen items have to be put in a knapsack with capacity c. The problem is to select a subset of items whose total weight does not exceed the knapsack capacity c and whose total profit is the maximum. The formal description of the problem is shown in the following formula.

$$P = max \sum_{i=1}^{n} p_i * x_i \tag{1}$$

subject to

$$\sum_{i=1}^{n} w_i * x_i \leq c, x_i \in [0,1], i \in 1,...,n \tag{2}$$

$$x_i = \begin{cases} 1 & item \quad \text{i} \quad has \quad been \quad chosen \\ 0 & else \end{cases} \tag{3}$$

x_i	item i
w_i	weight of item i
p_i	profit of item i
n	number of items

To cover the energy usage of devices in the pool, a set of devices is needed that is able to produce or consume the energy. Normally the energy usage will last for several minutes so the devices must cover it for the same time. In contrast to the items in the classical knapsack problem, the devices have 2 dimensions (load and runtime) and the capacity of the knapsack is the needed energy over time. The goal is to fill the knapsack as good as possible so that the remaining deviation is minimised. In the classical knapsack problem, this would mean that the profit of an item is the same as the weight. The maximum profit is then the same as the capacity of the knapsack. Another difference to the classical problem is that the pieces do not have a fixed size. This means the load of all devices is not fixed but start time, runtime and energy usage can be adjusted. It is even possible to overload the knapsack by scheduling more devices than necessary. In literature, there are different approaches on how to deal with overloading. In many papers a penalty is added to the profit in order to avoid overloading [27] [22]. In this scenario, it does not matter whethter the deviation is positive or negative. Therefore overloading means that the resulting deviation increases again and because of this no extra penalty is needed.

The formal description of the problem is shown in the next equation.

$$I = min \left| \sum_{t=t_s}^{t_e} \left\{ P_c(t) + \sum_{i \in A} P_i(t) \right\} \right| \tag{4}$$

c	device c that needs cover
$P_c(t)$	energy usage of device c in time slot t
$P_i(t)$	consumption or production of device *i* in time slot t
t_s	start time of the imbalance
t_e	end time of the imbalance
I	remaining Imbalance

In equation 4, the sum of all covering devices in every time slot is added to the energy usage of the load of device *c*. The energy usage of device *c* and of all other devices can change in every time slot. As described above, it does not matter if the cover is too small or too large. Only if I is 0, the cover fits perfectly and production and consumption are in balance. In contrast to many synthetic knapsack problems, most of the time there is not only one optimal solution. The possibility to resize the schedules leads to many possible optimal solutions. Because of this the optimisation problem gets much easier and this reduces the resources needed to find an optimal solution.

All devices in the pool are responsible for finding suitable partners to cover their energy usage. In this decentralised approach all devices form a Peer-To-Peer (P2P) network where every node can communicate with all other nodes. The details of the P2P network are not shown here, but it is implied that a P2P network is available. Due to privacy reasons, it is not possible to send the characteristics of all devices to a central server, which can identify suitable partners for all appliances. Furthermore, much computational resource would be needed to solve the resulting problem with all the constraints of all devices and power plants.

In this approach the identification and the optimisation are done by the households using the existing infrastructure. This can be done by each device or by a central unit inside each household. One of the main objectives of this approach is to keep the technical requirements so low that it is inexpensive enough to equip every device with a very small control unit or that programmable routers can be used for this task. Routers are advantageous in that they already have access to the internet. It would also be possible to use a normal computer (2GHz single core) but it would not be efficient to run a computer whose energy use is higher than the consumption or production of the controlled devices and power plants. To take part in such a pool the devices must be able to express their energy needs and to describe their degree of freedom regarding their energy usage.

This appliance pool is not designed to completely replace the normal balancing power plants but to reduce the amount of energy needed by balancing groups. As stated in section 2.3, balancing groups have to pay for all imbalances between their forecast and the actual consumption. The balancing groups are therefore interested in a fast adjustment of production and consumption.

Not all home appliances can be used for this pool because they do not have a degree of freedom in their energy usage. Only devices which are able to shift their energy usage at least for a couple of minutes can be used. For instance, multimedia appliances cannot be rescheduled because the impact on the household would be too large. The following list enumerates the devices which can be used in this approach:

- Air conditioner
- Refrigerator
- Freezer
- Elec. Heating
- Dishwasher
- Washing machine
- Dryer

The energy usage of these appliances can normally be shifted for a couple of minutes without any problems.

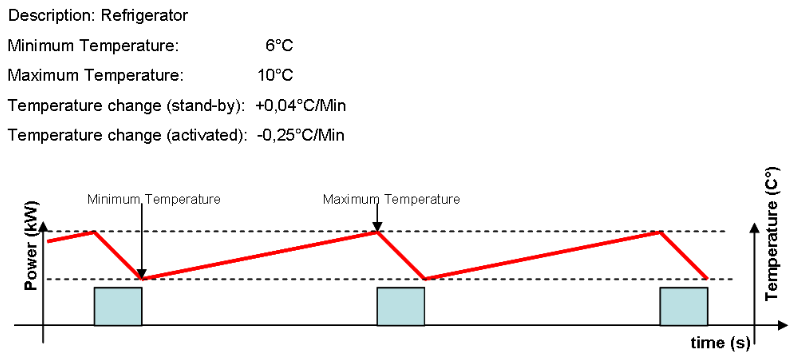
The households that own the devices are neither at a disadvantage nor at an advantage if their appliances are influenced by external signals. Furthermore, to integrate appliances into such an appliance pool the household has to pay for the internet connection and the control units. Therefore, the household must be paid a small participation fee. If the pool is managed by the balancing group manager, it can be used to reduce the cost of balancing power. The balancing group manger divides a fraction of these savings between all participating households and keeps the rest. If the pool is managed by someone else, this manger has to provide an incentive. Without an incentive, households will not take part in such a system. In contrast to secondary balancing power, no kW hour rate is needed because the appliances and CHP plants must run to fulfil their restrictions. Only the activation time has changed and not the overall energy usage so that usually no additional charge has to be paid. In the proceeding subsections the data format used to describe the degree of freedom of the appliance, the optimisation inside the pool to hide the restrictions and the usage of the pool to reduce imbalances is presented.

4.1 Data Modelling

One of the main problems is how the appliances and CHP plants can exchange information on their energy usage. This includes the energy needed or produced, the exact time and the duration. Based on these values it is possible to modify the load curve considering the restrictions and constraints. Note that for CHP plants the production is only a negative consumption and vice versa. In the following, it will only be distinguished between devices that consume electricity and power plants that produce electrical energy if the difference is important. Below, refrigerators are the main example used to explain the data modelling. The usage of the data model is identical for all other appliances.

The energy usage of the appliances depends on the internal status (i.e. temperatures for refrigerators) and the tasks it has to perform (i.e. dry clothes). The task is normally specified by the owner with certain restrictions (i.e. finish drying until 6pm). To determine the status value of an appliance or a plant it needs the ability to measure some parameters and check its restrictions. Based on these values it is possible to predict the status in the future. For a refrigerator this means that it must know the actual temperature, the minimum or maximum temperature specified by the owner and the temperature change per minute if turned on or off. If this data is not known in advance by the manufacturer or depends on the current situation, it can be easily measured automatically. With this data, the time the refrigerator has to be turned on again can be calculated easily. The specified data like temperature changes and the prediction can be influenced by external effects. These effects could be, for instance, that the refrigerator is opened by the owner, weather conditions or the filling of the fridge. Due to this, the prediction has to be adjusted, if necessary.

Figure 2 shows the condition of a refrigerator. The solid line marks the temperature over time and the dashed lines mark the limits (right scale). The blocks show the time, length and energy usage of a device (left scale). At the top, all relevant parameters are shown. As you can see, the refrigerator is always turned on and of when reaching the limits.


Description: Refrigerator

Minimum Temperature:	6°C
Maximum Temperature:	10°C
Temperature change (stand-by):	+0,04°C/Min
Temperature change (activated):	-0,25°C/Min

Fig. 2 Parameters of a refrigerator

With this data, a control unit is able to predict the point of time when the refrigerator must be turned on at the latest. Regarding the maximum and minimum temperature, a refrigerator is rather free in its energy usage. Normally a refrigerator starts when the temperature reaches the maximum temperature and stops at the minimum temperature. The energy consumption is fixed by the manufacturer and depends only on the refrigerator itself, so to change the temperature permanently a refrigerator can only increase its activated time. Other, more complex devices have many more constraints like

- minimum and maximum runtime
- minimum and maximum rest period
- minimum and maximum power consumption
- minimum and maximum power production (for CHP plants).
- latest time to finish a job
- earliest start time
- heat profile of a household (for CHP plants).

The degree of freedom in the energy usage is limited by these constraints. In figure 3 the status of a refrigerator is shown again. This time the schedules are not automatically created when the temperature reaches the limits but rather free, but with respect to the constraints.

As can be seen with the schedule shown in figure 3, a) the refrigerator will exceed the maximum temperature. In order to keep the limits it is necessary to activate the refrigerator again before reaching the limit. With regard to the rest of the schedule there are several points of time where the refrigerator can be activated. Figure b) shows a new schedule. The new activation time starts at t_1 and must at least last until t_2 and at the most until t_3. A shorter or longer activation time will exceed the limits, as shown in the figure. These values show the degree of freedom of an appliance regarding a fixed start time. For other start times it is necessary to calculate all values again. This works as well for CHP plants which produce heat for the household and electricity. Not only the activation of a device or a CHP plant can be used in the pool but also the deactivation. If a device stops consuming energy, it is the same as if a power plant produces this amount. This means that a refrigerator can produce energy by going into stand-by mode, and a CHP plant can consume energy by lowering its production.

The calculation of all possible and correct schedules is rather complex concerning all the constraints mentioned above. Another problem is the privacy of the data. With the specification and the constraints of an appliance it is possible to get confidential information on the household (e.g., is anybody home during the day, which devices are available and other private information). To cope with this problem, the transferred data to describe the degree of freedom of every appliance is restricted to non-confidential information which cannot be used to gather private information on the household.

Only the following values are considered to express the degree of freedom of an appliance for a specific time:

- maximum activation time

- minimum activation time
- minimum energy consumption/production
- maximum energy consumption/production

With this simple representation it is possible to express many possible schedule changes. Due to the complex constraints, it is not possible to express all valid schedule changes in this format.Note that this format can also be used by a household to represent not only one appliance but the data of all controllable devices in a household combined. This reduces the communication overhead and hides appliance details from the outside.

To express the degree of freedom for a certain period, these values have to be computed for every possible start time. Within a timeframe of 60 minutes and a minute-based slot size there are 60 possible start times and, for this reason, 60x4=240 values for each production and consumption. This data format is much smaller than the complete description of the device including the complete schedule for the next couple of hours. Because of the restrictions, which must be complied with, the complete schedule, instead of only parts, would be necessary (cf. figure 3). With this simple data format all appliances and CHP plants can exchange their data without violating the privacy of the households. The calculation of the needed values is rather simple and does not require much computational power. The calculation can be done by control units like intelligent routers inside the households. In the next subsection this data is used to optimise the appliance pool.

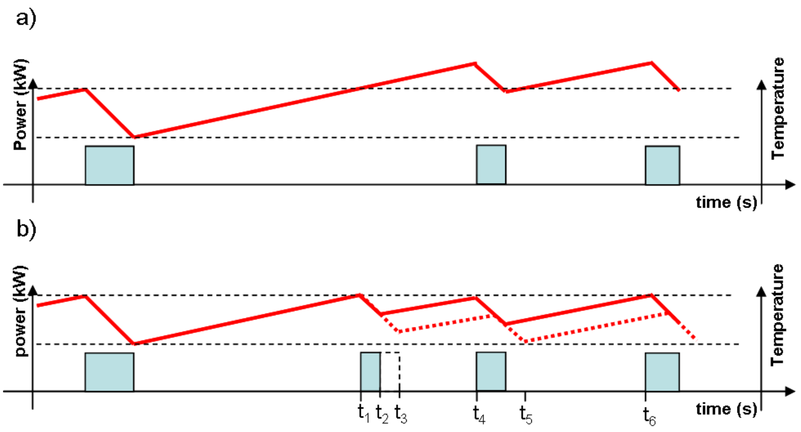


Fig. 3 Possible schedules for a refrigerator

4.2 *Optimisation*

The main objective is to create a decentralised appliance pool similar to the balancing pools in section 3.1. Due to the restrictions, the appliances and CHP plants will consume and produce power at will. Production and consumption must be in balance in order to use the pool to reduce the amount of balancing power in a balancing group. To create a balance between consumption and production in the device pool, all appliances and CHP plants must find suitable partners to cover their energy usage. This means that energy consumers must find one or more CHP plants that can produce the necessary amount or other appliances that can be turned off. In a first step a pool is created which is able to satisfy its power demands without external power plants and loads. In the following subsection, this pool is used to reduce the amount of balancing power needed.

With the data format introduced in the last paragraph, all devices are able to express their degree of freedom regarding the energy usage. Furthermore, a device is able to predict its own energy usage in the future. If a device foresees that it has to be turned on in the next couple of minutes, it has to ask its neighbours to cover the energy consumption or production. The device contacts its P2P neighbours over the internet and asks for their description of the degree of freedom for the period in which it wants to consume or produce power. The number of neighbours needed depends on the ratio between the own energy needs and the possible consumption/production of the neighbours. If the neighbourhood is too small, there might not be enough other devices to find suitable partners, and this will lead to an imbalance in the pool. If the neighbourhood is too large, the communication overhead will increase, just like the needed computation time to identify and select the best set of partners. Due to this fact, every device has to adjust the size of the P2P neighbourhood regarding the quality of the solution and the time needed to find a good solution. If the quality of the optimisation is not sufficient, the neighbourhood can be increased. If it takes too much time for the optimisation, the number of neighbours can be decreased.

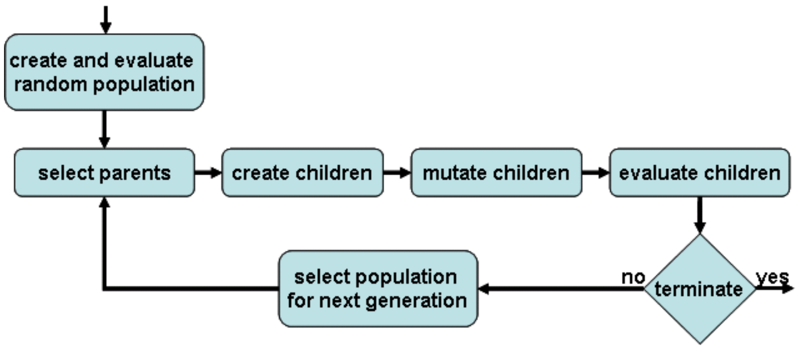


Fig. 4 General structure of Evolutionary Algorithms

When asking other devices for cover, the stated period has to be slightly bigger than the activation time. This allows covering devices to start a little bit earlier or run longer in order to be used for cover at all. An early start time or a longer runtime will lead to an imbalance but this might be the only possibility to get an acceptable cover. To cover the energy use, CHP plants can either increase the production or other devices can reduce their load. If a CHP plant needs to produce power because the heat is needed by the household, it is possible to decrease the production or to increase the consumption in the pool. How many appliances or CHP plants are needed depends on the amount of power needed and the consumption/production of the neighbours.

As described before the resulting optimisation problem is a variation of the well-known knapsack problem. Normally the problem size is small enough to be solved even by full enumeration on today's standard computers. Since it is not efficient to run a computer with a power consumption of more than 200 Watts for load balancing purposes in every household, an Evolutionary Algorithm is used to speed up the calculation of good solutions. Evolutionary Algorithms have proven to be able to solve such problems and there is a lot of research in this field. The Evolutionary Algorithm is used to select suitable partners and to create the corresponding schedules. In an Evolutionary Algorithm, different possible solutions are recombined and slightly changed in order to find better solutions.

In an Evolutionary Algorithm, different solutions are called individuals and an individual normally consists of several parts, the so-called genes. For example, the genes could be different parameters in a function. The individuals are rated with regard to the fitness function, which represents the problem and allows the comparison of individuals. The general structure of Evolutionary Algorithms is shown in figure 4.

In the first step a population of random individuals is created and all individuals are evaluated with regard to the given fitness function. Then several parents are selected and used to create new individuals from the genes of the parents. The genes of the children are mutated randomly to create new genes in the population. After the mutation step the children are evaluated and the new population is selected with regard to the fitness. If the resulting fitness is not good enough, the algorithm starts again for a new generation. Besides fitness, other conditions (e.g. computing power or time) may limit the number of generations.

Here the Evolutionary Algorithm should select the partner and create the schedules with minimum resources. Based on the format from the last paragraph, the Evolutionary Algorithm has all the information from the neighbours that is needed to find a good cover.

Because every device has to calculate its degree of freedom for itself, and it can do this in advance, the resulting problem is much smaller than when one device has to compute the degree of freedom regarding the restrictions for all neighbours and then select suitable partners. The Evolutionary Algorithm has only to decide if a neighbouring device should be used at all. It has to find the best start time, runtime, and consumption/production of electricity for all those devices. Every device can only be used once, otherwise it could exceed the restrictions of the appliance (cf.

subsection 4.1). The genes of an individual in the Evolutionary Algorithm contain the schedule for one device, which consists of start time, runtime and energy usage.

In the first tests a standard Evolutionary Algorithm is used. The population size is 15, and a (15 + 2) reproduction scheme is used, i.e. in every generation, 2 offspring are generated and compete for survival with the individuals from the old population. Crossover type is a simple one-point crossover. Every schedule (consisting of start time, runtime and load) for a device is one gene. The crossover takes a random number of genes from one parent and the rest from the other. During the mutation step the start time, runtime and the load can be changed. The start time is randomly shifted and the runtime randomly increased or decreased. Since start time and runtime are integer values the range of possible values is rather small. The load is a real value with a much wider range. The mutation of the load is Gaussian and the step size is fixed with respect to the minimum and maximum load of the device.

The quality of the solution depends on the neighbours in the P2P network, the distribution of energy consumers and producers in the district and their energy needs during the day.

As main test scenario, a virtual district based on real data and various statistics [8] with more than 1000 households is used. This model has been used before to test the impact of price signals on the load management of households and small businesses. There are 1960 refrigerators or freezers and 84 combined heat and power plants in the district which are now used to form a decentralised balancing power pool. The energy consumption of the refrigerators and freezers is almost the same as the production of all combined heat and power plants. Because of this difference, there will always remain an imbalance in the pool. The district is simulated for 1440 minutes (1 day). Every device has to find cover for its energy demand over the course of the day. This leads to more than 20,000 runs of the Evolutionary Algorithm per simulation.

In the first test without any optimisation the resulting imbalance in the pool is almost 11% of the complete production and consumption. This imbalance is not only caused by the uncoordinated energy usage of the devices but also by the difference between overall consumption and production mentioned before. This difference between production and consumption cannot be avoided and is responsible for about one sixth of the resulting imbalance. The minimal possible imbalance is therefore greater than about 2%. Regarding the different constraints of the devices and the overall energy demand, the minimal imbalance has to be even greater. The goal is to get as close as possible to this minimal imbalance.

Table 3 Remaining imbalance in the device pool (EA only)

	Size of neighbourhood							
	10	40	70	100	130	160	190	220
Remaining imbalance (%)	10.75	9.52	8.47	8.15	7.91	7.84	7.41	7.27
Average cover (%)	21.69	37.64	40.94	45.50	49.07	51.89	54.2	56.31

Using the Evolutionary Algorithm to select suitable Partners and to create the needed schedules reduces the overall imbalance to 7.22%. Every entry in the following table is the average of at least 10 simulation runs. Every simulation configuration was tested with different parameter settings and the best results are shown in table 3. The size of the neighbourhood is fixed but normally not all neighbours are able to provide cover. Therefore, the resulting number of possible schedules can be much smaller.

It can be seen that the resulting imbalance can be reduced by increasing the size of the neighbourhood. The second row shows the average cover a device gets when looking for suitable partners. Depending on the size of the neighbourhood and the restrictions of the devices in the neighbourhood, it often happens that there is not one suitable partner and the resulting cover is 0%. This problem reduces the overall performance in the simulation but is not a direct problem of the Evolutionary Algorithm. Additionally, devices and combined heat and power plants sometimes ask for a relatively small amount of energy. If, for example the minimum production of a CHP is much greater than the needed cover the CHP plant cannot be scheduled and the resulting cover is again 0%.

The analysis of the data showed that the Evolutionary Algorithm performs well if it must not adjust the load of the devices. The algorithm finds the integer values for start time and runtime quickly but is not able to adjust the real values for the load that fast. There are two problems adjusting the load of devices. The load has a bigger range (e.g. a CHP plant can produce between 1000 and 5500 Watts) and the range is not steady (0 and from minimum production to maximum production). Increasing the number of generations does not work as well as increasing the neighbourhood. In a bigger neighbourhood there are more devices with different load profiles form which the algorithm can select the best subset. Even if the neighbourhood contains more than 100 devices, not more than 30 are normally used in the best solution. A bigger neighbourhood increases the resources (computation power, memory, bandwidth) needed to find a good solution and therefore cannot be done by small control units. Because of that, a local search is integrated into the Evolutionary Algorithm that should help the algorithm find the local optimum for every individual. The changed structure of the Evolutionary Algorithm is shown in figure 5.

The local search is very simple but also very efficient with regard to this problem. For every planned device, the local search tries to find a better position just by adjusting the load and moving the schedule. After the recombination and the mutation step of the Evolutionary Algorithm, the local search is started and the resulting local optimum is inserted into the population as the new child. The population then consists only of local optima and that speeds up the convergence to a global optima. The local search performs the following tests for each device in an individual and accepts the solution if the cover improves:

- increasing or decreasing the energy usage
- earlier or later start point
- longer or shorter runtime
- later start point and a reduced runtime

- earlier start point and increased runtime

As long as there is a significant improvement all tests start again. The third and fourth tests look very similar to the first two but they cover a situation which the first two cannot detect. Figure 6 shows the first 4 situations. In the figures, the dashed rectangle symbolises the energy usage that is needed to cover a device, and the grey rectangle represents the schedule of a neighbouring device. In figure a) and b) it is obvious that the cover can be improved by simply moving to an earlier start point or to a later one. In figure b) and c) the cover improves by increasing or decreasing the runtime of the device. In figure e) none of the first two tests will lead to an improvement. Only the combination of a later start point and a reduced runtime leads to optimal improvement. In the last case, e) a combination of earlier start point and increased runtime is needed for an optimal cover. Because of this all five tests are necessary to find a local optimum based on an individual of the Evolutionary Algorithm.

It would be possible to skip tests c) and d) by changing the data format from start point, runtime, and consumption/production to start point, end point, and consumption/production. The resulting tests would then be:

- earlier or later start point
- earlier or later end time
- increasing or decreasing the energy usage

The selected format has some programming-related advantages and was therefore selected for implementation. All these tests are performed with respect to the constraints specified by the devices. If the power usage exceeds the lower limit specified by the device, that device is removed from the list. In the next generation, the Evolutionary Algorithm can schedule this device again, if needed.

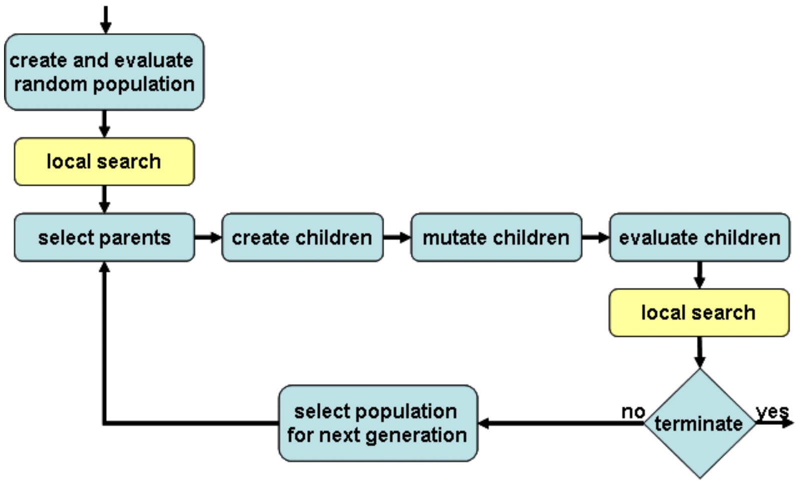


Fig. 5 Structure of the Evolutionary Algorithm with local search

In a last optimisation step, the number of devices is reduced. CHP plants and some appliances have the highest efficiency if used with full load. The efficiency decreases if they run blow their maximum load. Due to this fact, most of the devices should run with full load. The local search removes randomly single devices from the best solution of the Evolutionary Algorithm and tries, by employing the tests described above, to fill the gap. This optimisation step could also be included in the normal local search or it could be added to the Evolutionary Algorithm to assure that all solutions in the population have the minimal devices needed. The results prove that it is sufficient to run this optimisation step only once at the end and, by doing so, save the computational resources.

Table 4 Remaining imbalance in the device pool (EA with local search)

	Size of neighbourhood							
	10	40	70	100	130	160	190	220
Remaining imbalance (%)	9.89	7.92	6.19	5.22	4.84	4.83	5.06	5.59
Average cover (%)	42.52	67.1	75.38	79.27	80.36	80.58	80.58	80.44

In table 4 the results of the combined algorithm are shown. The performance of the Evolutionary Algorithm in combination with a local search is significantly better than the results shown in table 3. The local search is very expensive regarding the computational cost for one generation of the Evolutionary Algorithm. The calculation of the fitness, which is the most expensive part, has to be done much more often than in the standard algorithm. Nevertheless, the combined algorithm is much faster because the size of the neighbourhood can be reduced. The reduced number of neighbours accelerates the optimisation again. Furthermore, the number of

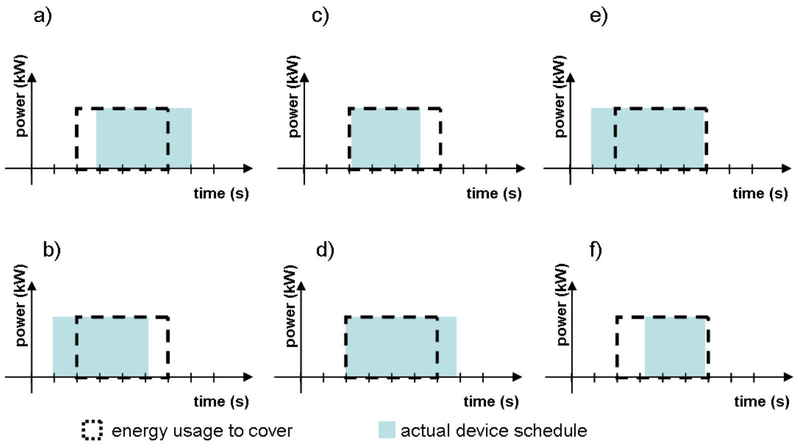


Fig. 6 Tests of the local search

generations can be decreased from more than 1200 to about 160, saving additional resources.

The Evolutionary Algorithm is now used for the exploration of the search space, and the local search is used for the exploitation in a specific region. The Evolutionary Algorithm selects the set of devices that should be used and the local search is responsible for moving and resizing the schedule so that the given set of devices fits best. Although the performance of the local search in finding a local optima is much higher than the performance of the Evolutionary Algorithm, the local search is not able to move to a new search space or to select a new set of devices. For example, only the Evolutionary Algorithm is able to switch the start time of devices or to adjust the runtime of more than one device at the same time.

With this combination, the imbalance in the pool can be significantly reduced without the need for any central control. The resulting pool is able to hide its restrictions to the outside and the remaining imbalance could be reduced by more than 55%. The remaining degree of freedom in the pool can now be used to reduce imbalances from outside the pool. In the next subsection, this pool is used to reduce the necessary balancing power.

Figure 7 depicts a screenshot of the Graphical User Interface of the simulation tool. The diagram contains the load curve and the status of a CHP plant which needs cover for its energy production. The production of the CHP plant is shown above the x-axis and below are the appliances which cover the energy usage. As can be seen, the production of the CHP plant is perfectly covered. Every small block symbolises a different appliance or CHP plant. To cover a CHP plant, many different appliances are needed which can satisfy their energy demand by covering the CHP plant.

The results show that the algorithm does not need many resources and can be implemented on much smaller devices with only a fraction of the computational power. The decentralised approach, where all devices determine their degree of freedom for themselves, and then these descriptions are used for the optimisation, speeds up the algorithm. Therefore, with a forecast time of 2 to 5 minutes and a small DSL internet connection there is enough time to communicate with the neighbours and to find a good solution.

In table 4 the results are shown if the algorithm does not stop after a number of generations but after a specific time (here 100 and 1000 ms). The results show that even in this short period the algorithm can find a good cover. The best size for the neighbourhood is between 70 and 130.

Table 5 Remaining external imbalance in percent (EA with local search)

	Size of neighbourhood					
Optimisation time	10	40	70	100	130	160
100ms	9.95	7.73	6.25	5.63	5.48	6.02
1000ms	9.88	6.39	5.12	5.52	5.90	6.58

If the cover is not good and there is enough time, the device can increase or change the neighbourhood and start the optimisation process once more or just take the best solution. The missing cover will result in an imbalance which must be dealt with by obtaining balancing power.

4.3 Providing Balancing Power

The last section introduces a method to create an appliance pool which causes only small imbalances in the power grid and tries to satisfy its own energy needs. This pool can now be used in the same way as the balancing power pools in section 3.1.

The remaining degree of freedom of the devices in the pool can be used to provide balancing power. The method is similar to the search of cover of a single device shown in the last paragraph. Every time imbalance of a specific size occurs in a balancing group, the balancing group manager can simply submit a request to the pool to cover a specific amount of the imbalance for a fixed period. If the pool is

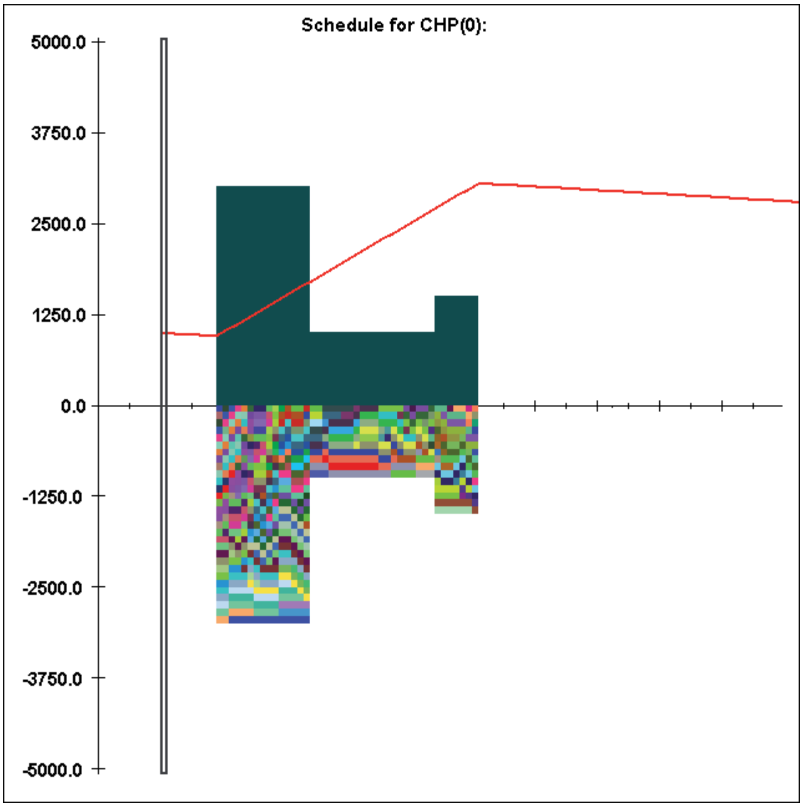


Fig. 7 Cover of a Combined Heat and Power plant

able to cover this amount, it is not necessary to obtain balancing power from other expensive power plants.

The balancing group manager is able to control the power output in the grid without controlling all appliances and CHP plants and without knowing details about the devices in the pool. This allows the balancing group manager to adjust to new situations caused by renewable energy sources and reduces the expensive imbalances in his balancing group.

Another advantage of the decentralised pool is that it is not necessary for the balancing group manager to provide a central computer which handles all imbalances; rather, it is possible to use as many control units as needed. At every connection between different balancing groups, the current flow is measured to determine the schedule in the balancing group. Because the current flow is known in advance for every connection, it is possible to determine which balancing group is responsible for the imbalance, and react accordingly. Every connection point that is able to measure the current flow can be amended with a control unit similar to those in the households.

The balancing group manager can even feign an imbalance to control the power usage in the area. This can be used to either balance the unforeseen output of renewable energy sources or to cope with a mismatch of current consumers and combined heat and power plants. If the pool only consists of energy consumers, it is not possible for the pool to produce its own power consumption. Due to this fact, the pool will always produce a loss of energy in the balancing group. To solve this problem the balancing group manager can feign a power surplus in the grid so that the pool has enough imbalance to satisfy the energy needs of the pool. To do so, the manager has to make a higher forecast for the balancing group and make sure that normal power plants provide the required energy. The fake imbalances work as additional CHP plants for the devices in the pool, which they try to cover as described above. This strategy also works in pools consisting only of CHP plants. In such a situation, the balancing group manger has to lower its forecast so that the CHP plants can be turned on to produce heat for the households whenever necessary.

The distributed balancing power pool has been tested with an external synthetic imbalance.The results of the tests are shown in table 6. After a fixed period, an imbalance was created with a fixed load and duration. The pool was then asked to cover the imbalance as good as possible. After that imbalance, the pool had a

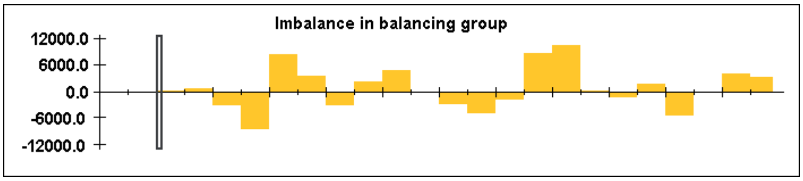


Fig. 8 Imbalances in a virtual district

Table 6 Remaining external imbalance in percent with and without optimisation (values in brackets)

	load of imbalance in kW					
duration	10	20	30	40	50	60
30 min	5,29 (12,70)	6,07 (15,48)	9,24 (19,80)	14,75 (27,08)	20,18 (33,68)	25,93 (41,66)
60 min	5,25 (12,85)	6,15 (14,04)	8,39 (17,94)	11,86 (23,14)	16,16 (28,35)	20,72 (32,96)
90 min	5,35 (12,70)	5,92 (14,02)	7,69 (16,08)	10,09 (20,68)	14,15 (24,46)	17,77 (27,81)

couple of minutes to organise again until the next imbalance (now with the negative load) was created. The size of the neighbourhood for the optimisation of the external imbalance was set to 160 and the size of the neighbourhood for the devices was set to 100. The results show that the overall imbalance can be reduced by 40 to 60%.

Figure 8 shows an example of random imbalances in the simulation. The imbalance changes over time and can be positive and negative. The scale on the left shows the deviation in Watts. The balancing group manager has to pay for the imbalances because he has to obtain balancing power from outside to balance his group. By installing the decentralised appliance pool, the deviation can be reduced. The first diagram of figure 9 indicates the resulting imbalances. The reduction of deviations is significant. Whereas in the example in figure 8 the largest external imbalance is almost 12,000 Watts, the largest deviation in figure 9 is 55 Watts at most. The imbalance inside the pool is not shown in the figure.

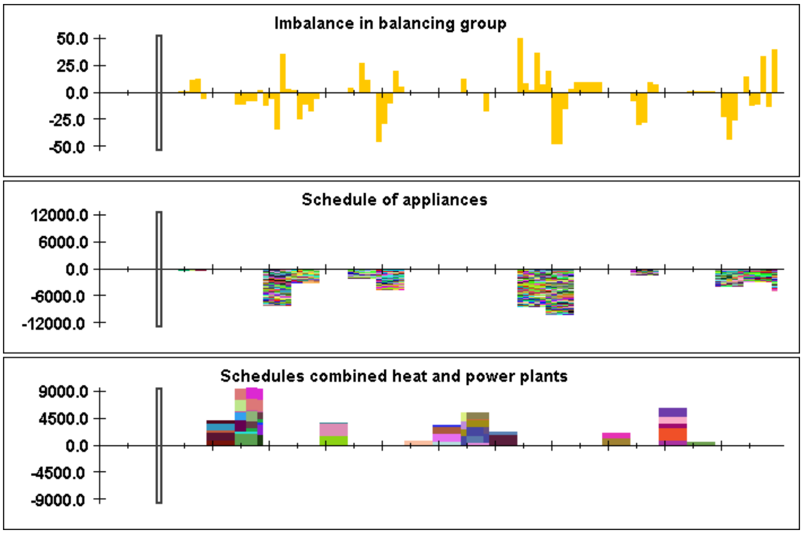


Fig. 9 Reaction of the decentralised balancing pool

The second and third diagram of figure 9 show the resulting schedule for the home appliances and CHP plants. Many home appliances and CHP plants are needed to cover the imbalance.

With the new approach and the combination of the Evolutionary Algorithm with the local search the remaining imbalance could be reduced by up to 60%. These results show that even small devices with many restrictions can be used to provide balancing power.

5 Conclusion

In this chapter, a new approach was presented that shows how small home appliances like washing machines or refrigerators and privately owned combined heat and power plants can be used to reduce the balancing power. Like existing balancing power pools and Virtual Power Plants, where smaller power plants and larger sheddable loads are pooled to meet the requirements to provide balancing power or act as one big power plant, these home appliances are also pooled together. Although these self-organising appliance pools are not large enough to meet the requirements to provide balancing power, they can still be used by balancing groups to reduce the amount of balancing power needed.

Due to restrictions of the home appliances, these devices cannot be used exclusively due to restrictions and specific tasks from the owner. A self-organising approach concerning how the pool can hide the restrictions of the appliances from the outside was presented in this chapter. The devices in the pool have to find suitable partners inside the pool which replace them if they have to fulfil specific tasks and other partners to cover their energy demands so that consumption and production are always in balance.

Because it is not efficient to run a computer in every household, the resources necessary to find these partners must be kept very low so that small control units or even DSL Routers are able to perform the optimisation. A data model for all devices to express their energy needs, their restrictions and degree of freedom was outlined. For the optimisation an Evolutionary Algorithm in combination with a local search is used to find partners and create the corresponding schedules. The resulting pool acts as though it has no restrictions at all, and the remaining degree of freedom can be used to reduce imbalances in a balancing group. Therefore, the balancing group manager can use the pool as though he needed cover for one of his devices to reduce the deviation in the balancing group.

References

[1] Arndt, U., Roon, S.V., Wagner, U.: Virtuelle Kraftwerke: Theorie oder Realität? BWK 58(6), 52–57 (2006)

[2] Auer, H., Huber, C., Stadler, M., Obersteiner, C., Ragwitz, M., Klobasa, M.: Modellierung von Kraftwerksbetrieb und Regelenergiebedarf bei verstärkter Einspeisung von Windenergie in verschiedene Energiesysteme unter Berücksichtigung des Lastmanagements, endbericht edn. TU Wien (2005)

[3] Auer, H., Haas, R., Faber, T., Weißensteiner, L., Obersteiner, C., Fuchs, E., Heher, A., Höhne, U., Molnar, P., Kastner, S.: Faire Wettbewerbsbedingungen für Virtuelle Kraftwerke - Projektbericht im Rahmen der Programmlinie Energiesysteme der Zukunft. Bundesministerium für Verkehr, Innovation und Technologie, Wien (2006)

[4] Chu, P.C., Beasley, J.E.: A genetic algorithm for the multidimensional knapsack problem. Journal of Heuristics 4(1), 63–86 (1998), http://dx.doi.org/10.1023/A:1009642405419

[5] dena: Zusammenfassung der wesentlichen Ergebnisse der Studie, Energiewirtschaftliche Planung für die Netzintegration von Windenergie in Deutschland an Land und Offshore bis zum Jahr 2020 (dena-Netzstudie) durch die Projektsteuergruppe (2005)

[6] Driesen, J., Katiraei, F.: Design for Distributed Energy Sources. IEEE power & energy magazine, 30–40 (May/June 2008)

[7] Eikmeier, B., Schulz, W., Krewitt, W., Nast, M.: Nationales Potential für hocheffiziente Kraft-Wärme-Kopplung. Euro Heat & Power 35(6), 2–10 (2006)

[8] Eßer, A., Kamper, A., Franke, M., Möst, D., Rentz, O.: Scheduling of electrical household appliances with price signals. In: Waldmann, K.H., Stocker, U.M. (eds.) Operations Research Proceedings 2006. Springer, Heidelberg (2006)

[9] ETSO, Current State of Trading Tertiary Reserves Across Borders in Europe. ETSO (2005)

[10] EUROSTAT Energy - yearly statistics 2005 (2007)

[11] EWI, Progno, Die Entwicklung der Energiemärkte bis zum Jahr 2030 - Energiereport IVB (2005)

[12] EWI, Prognos, Energiereport IV - Die Entwicklung der Energiemärkte bis zum Jahr, Oldenbourg Industrieverlag (2005)

[13] Forster, J.: Mehr als nur Regelenergie - Ein Jahr virtuelles Kraftwerk der Stadtwerke Unna. Energie Spektrum 20(4), 16–17 (2005)

[14] hessenEnergie, Mikrogasturbinen im Markt der Kraft-Wärme-Kopplung (2004)

[15] Hille, M., Pfaffenberger, W.: Power Generation in Germany: How to Close the Gap in Gemeration Capacity in the Context of a Liberalized Energy Market. In: 3rd Conference on Applied Infrastructure Research Network Economics: Financing, Regulation and Capacity Allocation in Infrastructure Sectors, pp. 953–972 (2004)

[16] Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack problems. Springer, Heidelberg (2004)

[17] Ingenierubüro, H.K.: Das virtuelle Regelkraftwerk (2008)

[18] Kirby, B.: Spinning Reserve From Responsive Loads. Oak Ridge National Laboratory, Oak Ridge (2003)

[19] Kumar, R., Banerjee, N.: Analysis of a multiobjective evolutionary algorithm on the 0-1 knapsack problem. Theor. Comput. Sci. 358(1), 104–120 (2006)

[20] Marquardt, F.: Analyse industrieller Bereitstellungspotentiale an Netzregelenergie (Diplomarbeit). Universität Karlsruhe, Karlsruhe (2007)

[21] Möst, D., Genoese, M., Eßer, A., Rentz, O.: European electricity and emission market modeling - the design of emission allocation plans and its effects on investment planning. In: IEEE Proceedings of the 5th Conference on the Euroepean electricity market (EEM) (2008)

[22] Olsen, A.: Penalty functions and the knapsack problem. In: Proceedings of the First IEEE Conference on Evolutionary Computation, 1994 IEEE World Congress on Computational Intelligence, vol. 2, pp. 554–558 (1994)

[23] Rosen, J.: The future role of renewable energy sources in European electricity supply: A model-based analysis for the EU-15. Universitätverlag Karlsruhe, Karlsruhe (2007)
[24] Rothweiler, H.: Marktgesetz oder Manipulation. Energiespektrum (11), 24–25 (2005)
[25] der Markttransparenz in Stromhandel"' SBLAV, Verbesserung der Markttransparenz auch auf europäischer Ebene / Strompreisbildung an der EEX. In: Wirtschaftsministerkonferenz am 19./20, Darmstadt (November 2007)
[26] StromNZV, Verordung über den Zugang zu Elektrizitätsversorgungsnetzen (Stromnetzzugangsverordnung) vom 25. Juli (BGBL Teil I, S. 2243) (2005)
[27] Uyar, S., Eryiğit, G.: Improvements to penalty-based evolutionary algorithms for the multi-dimensional knapsack problem using a gene-based adaptive mutation approach. In: GECCO 2005: Proceedings of the 2005 conference on Genetic and evolutionary computation, pp. 1257–1264. ACM, New York (2005)
[28] VDEW, Repräsentative VDEW-Lastprofile (1999)
[29] VDEW, Energie im Haushalt, Berlin (2002)
[30] Roon, S.V., Arndt, U., Wagner, U.: Simulation von Mikro-KWK-Anlagen zur Bewertung von Netzintegrationskonzepten. In: 5th International Energiewirtschaftstagungs 2007 an der TU Wien (2007)
[31] Wacker, J., Schulz, C., Kurrat, M.: Virtuelle Kraftwerke mit Mini-Blockheizkraftwerken - eine wirtschaftliche Utopie? EW 104(17/18), 20–23 (2005)
[32] Ziesing, H.J.: KWK-Potentiale in Deutschland und deren Erschließung. Energiewirtschaftliche Tagesfragen (ET) 58(3), 50–59 (2008)

An Analysis of Dynamic Mutation Operators for Conformational Sampling

Alexandru-Adrian Tantar, Nouredine Melab, and El-Ghazali Talbi

Abstract A comparison analysis of dynamic mutation operators is proposed, having the conformational sampling problem as a case study. The analysis is sustained by a parallel Optimal Computing Budget Allocation (OCBA) selection procedure, employed in order to attain computational speedup. A Pearson system distribution based mutation operator is proposed, allowing for a highly flexible construction. As defined by a set of four parameters, the mean, variance, skewness and kurtosis, a large number of distributions can be simulated. As determined by the analysis outcomes, the class of operators exhibiting significant energy minimization or Root Mean Square Deviation (RMSD) bias is identified. Experiments are carried out on a large number of computational resources, allowing for the outline of an automatic *a priori* operator tuning and selection methodology. Although not presented in this chapter, similar complementary studies have been conducted on intensification operators and local search algorithms.

1 Introduction

In the early 1960's the Nobel prize laureate Christian Anfinsen [3] postulated on the thermodynamic equilibrium of proteins, the exposed principles setting the basis for future conformational sampling computational approaches. Direct consequences of his work lead to the hypothesis on which proteins attain a thermodynamic equilibrium determined by a kinetically accessible *singular* conformational state.

INRIA Lille - Nord Europe, DOLPHIN Project Team,
LIFL UMR USTL/CNRS 8022,
Parc Scientifique de la Haute Borne,
40, avenue Halley, Bât. A, Park Plaza,
59650 Villeneuve d'Ascq Cedex, France
Alexandru-Adrian.Tantar@inria.fr
Nouredine.Melab@lifl.fr
El-Ghazali.Talbi@lifl.fr

Furthermore, the aforementioned postulate lead to the conclusion that a *unique* connection exists between the constituent chain of amino-acids of a specified protein and its corresponding native structure conformation.

The importance of the conformational sampling problem is determined by the ubiquitousness of proteins in the living organisms – proteins intervene in intracellular signaling modulation mechanisms, interactions between antibodies and antigens, inhibitor design, etc. Computational modeling and prediction offer an alternative to laboratory *in vitro* experimentation, infeasible for large domain analysis study. Furthermore, with the arrival and development of high performance and high throughput computing, being on the edge of entering the petascale computing era, complex conformational sampling problems can be addressed. Introductory notions on conformational sampling, protein structure and large scale high performance computing for biology-related problems may be found in Neumaier[35], Cozzone[15] and Stewart[42].

Nowadays, of significant impact and importance, conformational sampling, mainly addressed through the protein structure prediction (PSP) and molecular docking problems, is one of the current intensive study topics of computational biology. In the following the protein structure prediction problem is to be considered, which, relying on the previous considerations, consists in determining the ground-state conformation of a specified protein, given its amino-acids sequence – the *primary structure*. The ground-state conformation term designates the associated tridimensional native form, also referred to as *zero energy tertiary structure*.

The conformational sampling problem is computationally difficult, as the number of possible conformations to explore exponentially increases in correlation with the magnitude of the involved molecular complexes. No further details are included here, more in-depth discussions, addressing complexity matters, being exposed in previous works [44, 36, 45]. Considering the aforementioned aspects, evolutionary algorithms , enclosing landscape adapted operators, stand as highly efficient approaches in order to address the high complexity of the problem and the afferent multi-modal landscape characteristics. Allowing for complex hybridization patterns and relying on the intrinsic parallel nature of the paradigm, evolutionary algorithms (EAs) represent the solution adopted for AutoDock [39, 32], ArgusLab [29], etc. A more detailed discussion on the principles of EAs is to be presented in a following section. Nevertheless, no extensive study exists on adaptive and dynamic techniques specifically designed for the afferent classes of problems, namely protein structure prediction and molecular docking.

Consequently, adaptive and dynamic aspects are considered in the following, with a focus on mutation operators. Relying on a parallel design selection procedure, the behavior of multiple operators is analyzed at the different stages of an evolutionary exploration strategy. Selection procedures are employed as a powerful alternative to classical statistical tests, allowing for efficient designs in conjunction with evolutionary algorithms. An introductory review on selection procedures may be found in the works of Branke, Chick and Schmidt [5, 6]. A first sequential example of combining an EA with a selection procedure is given in Schmidt, Branke and Chick [40]. An extension of this hybrid model and a wide range of designs can

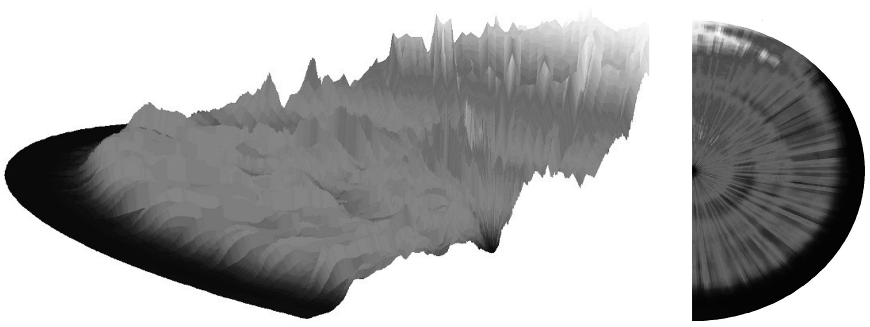


Fig. 1: Free energy surface cross-section (*tryptophan-cage* - 1L2Y) around a deep optimum point, exposing a large number of local optima. The equivalent mirrored map representation is offered on the right side of the figure. Lighter areas on the surface correspond to high energy points while low energy regions are depicted in darker colors.

be envisaged by introducing parallelism. Different designs can be put in place for parameter selection and during execution adaptive behavior, selection of the most appropriate algorithm at a given phase, etc. While being computationally expensive in a sequential design, a parallel integration of different metaheuristics and selection procedures can provide a powerful adaptive solution to difficult problems.

The analysis presented herein is part of a larger study on diversification, intensification operators, i.e., crossover operators and local search algorithms, the afferent work being under publishing process at the time being. While relying on analogous experimentation and statistical mechanisms, for brevity only mutation operators are included in the this study.

The following sections, including the contribution of this chapter, analysis, etc., can be resumed as follows:

Formalization and notational basis. A minimal notational formalization is defined. Exhaustive to no extent, the proposed definitions set the foundation for later modeling and presenting in a unified manner the considered operators, statistical components, etc.

Evolutionary Algorithm Analysis Context. Encoding of the conformations, evaluation function and general principles of an evolutionary algorithms are exposed. Different diversification operators are presented as well and employed for the conformational sampling problem. Dynamic mechanisms are considered, determining and influencing the intrinsic characteristics of the embedding exploration algorithm, specific distributions, etc. Pearson system based mutation operators are introduced and analyzed.

Statistical Selection Procedures. The formal fundamental support of a specific statistical selection procedure is exposed, as to sustain and justify the selection criteria employed in later setting the basis of the envisaged adaptive approaches.

Analysis of Mutation Operators. A statistical analysis of the defined operators is performed, identifying the components which, at different exploration stages, exhibit a superior energy or RMSD minimization tendency. A ParadisEO[1] based parallel implementation of the previously presented selection procedure is employed for testing, further, the implementation aspects, setup of parameters, etc., of the operators being detailed. Following the results provided by the statistical analysis procedure, the basis for later constructing adaptive and dynamic approaches is set.

The extracted analysis results further offer information on convergence and bias, in terms of *energy* and *Root Mean Square Deviation (RMSD)* minimization. Analysis conclusions are hence employed for defining and setting the basis of dynamic and adaptive resolution approaches. The underlying basis sustaining the aforementioned class of landscape adapted approaches relates to particularities of the fitness function which may potentially exhibit significant differences near the global optimum as compared to distant, far from optimum points. Finally, conclusions are made, laying the path for future study directions.

All experimentations were carried using an MPICH2 [18, 19] based version of ParadisEO [8, 9], a framework dedicated to the reusable design of parallel hybrid meta-heuristics and which provides a broad range of features, including EA support, local search methods, parallel and distributed models, hybridization mechanisms, etc. The algorithms were executed on Grid5000[2], a French nation-wide experimental grid [11], connecting several sites which host clusters of PCs interconnected by RENATER[3] (the French academic network). At this time Grid5000 is gathering more than 4000 processors with more than 100 TB of non-volatile storage capacity. For the hereafter analysis experiments, up to 500 computing cores per test were employed, execution times ranging from 15 minutes to several hours. Given the concepts on which the selection procedure is constructed, the time required for each analysis test is highly dependent on the statistical evidence attained after analyzing the execution results. If no significant statistical evidence is attained after a first run, additional samplings are requested so as to set a distinct difference between the compared operators, resulting in an increase of the execution time.

[1] http://paradiseo.gforge.inria.fr
[2] https://www.grid5000.fr
[3] http://www.renater.fr

2 Definitions and Formal Aspects

Considering the nature of the different exposed notions, relating to evolutionary algorithms, statistical analysis and selection, etc., the definitions herein are set on a generality basis in order to enclose terms derived from different areas and domains. Consequently, *de facto* notational standards are not closely followed due to an effort of unifying the different presented aspects.

Let Γ, Λ and Ξ be the set of solutions, operators and algorithms, respectively (e.g., replications and systems as for statistical selection procedures). With no generality restriction there can be assumed that the operators set, Λ, is a particular subset of Ξ:

$$\Gamma = \{\gamma \,|\, \gamma = (\gamma_1, \gamma_2, \cdots, \gamma_N),\ \alpha_i \prec \gamma_i \prec \beta_i\}, \qquad \hat{\Gamma} = \{\mathscr{S} \,|\, \mathscr{S} \overset{def}{=} \cup_{\gamma \subset \Gamma} \gamma\} \tag{1}$$

$$\Lambda = \{\lambda \,|\, \lambda : \hat{\Gamma} \times \hat{\chi} \rightarrow \hat{\Gamma}\}, \qquad \hat{\Lambda} = \{\mathscr{L} \,|\, \mathscr{L} \overset{def}{=} \cup_{\lambda \subset \Lambda} \lambda\} \tag{2}$$

$$\Xi = \{\mathbf{X} \,|\, \mathbf{X} : \hat{\Gamma} \times \hat{\Lambda} \rightarrow \hat{\Gamma}\} \tag{3}$$

As a general assumption it may be considered that $\gamma \in \Re^N_{\prec}$, $\alpha_i, \beta_i, \gamma_i \in \Re_{\prec}$, where $\Re^N_{\prec}$ represents an arbitrary solution space, with an associated arbitrary partial order relationship. An abstraction is made for now as for the type of the γ_i encoding values (discrete, continuous, etc.), to be later considered as notions are introduced. A solution $\gamma \in \Gamma$ is here defined as an encoding ensemble of bounded parameters with cardinality $N = |\gamma|$. Further, α_i and β_i denote the lower and upper limits of the γ_i encoding value, respectively, for $1 \leq i \leq N$. As a particular example, this notion relates to the genotype definition in the case of evolutionary algorithms where each γ_i value corresponds to a *locus* in the *genotype*. $\hat{\Gamma}$ denotes the set of all solution sets. $\hat{\chi}$ corresponds to the set of control parameter sets associated with a specific operator while $\hat{\Lambda}$ stands for the set of all the operator sets. As operator selection and parameter tuning are later discussed, the operators are presented as distinct entities in order to support the formal modeling of the two phases, selection and tuning.

Following the previous notations, an evaluation function E can be associated with the given solution space (e.g., a fitness function in the case of evolutionary algorithms) as well as an extension $\hat{E}$ acting on solution sets:

$$E : \Gamma \rightarrow \mathfrak{F}, \qquad \hat{E} : \hat{\Gamma} \rightarrow \mathfrak{F},\ \hat{E}(\mathscr{S}) = \min_{\gamma \in \mathscr{S}} E(\gamma),\ \mathscr{S} \in \hat{\Gamma} \tag{4}$$

For the general case consider the $\min_{\gamma \in \Gamma} E(\gamma)$ minimization problem, further assuming, with no strong restrictions for the following presented topics, that E is nonlinear and $\mathfrak{F} \subseteq \mathbb{R}$. Based on the nature of the E function, optimal solution points can be defined by employing neighborhood, norm or derivative notions. In the discrete case, γ^+ is said to be a local optimum solution if, for a given arbitrary $T(\cdot)$ neighborhood topology, $T : \Gamma \rightarrow \hat{\Gamma}$, or, for an arbitrary norm $\|\cdot\|$, defined on the solution space, the following stand, respectively:

$$E(\gamma^+) \le E(\gamma),\ \gamma \in T(\gamma^+) \tag{5}$$

$$E(\gamma^+) \le E(\gamma),\ \|\gamma - \gamma^+\| < \tau_\varepsilon,\ \tau_\varepsilon > 0 \tag{6}$$

If E is continuous and double differentiable, γ^+ is considered to be a local optimum point if, for all $\gamma_i^+ \in \gamma^+$:

$$\frac{\partial E}{\partial \gamma_i^+} = 0,\ \frac{\partial^2 E}{\partial^2 \gamma_i^+} > 0. \tag{7}$$

The global optima solution set, further denoted as Γ^*, can be defined as a subset of Γ:

$$\Gamma^* = \{\gamma^* \mid E(\gamma^*) \le E(\gamma),\ \gamma^* \in \Gamma, \forall \gamma \in \Gamma\},\ \Gamma^* \in \hat{\Gamma} \tag{8}$$

As finding the global optimum represents almost exclusively a *desideratum* attainable for theoretical cases only, it is generally accepted as sufficient a result identifying the *near global optimum solution set* which can be defined by setting a threshold $\varepsilon, \varepsilon > 0$:

$$\Gamma_\varepsilon^* = \{\gamma_\varepsilon^* \mid E(\gamma_\varepsilon^*) \le E(\gamma) + \varepsilon,\ \varepsilon > 0, \gamma_\varepsilon^* \in \Gamma, \forall \gamma \in \Gamma\},\ \Gamma_\varepsilon^* \in \hat{\Gamma} \tag{9}$$

Note that, while not directly addressed in the concepts modeled here, parameter tuning and operator selection can be defined by considering analogous formulations in definitions 8 and 9. In this context, the problem of selecting a particular operator out of a specified set can be indirectly seen as a parameter tuning or selection mechanism.

For the following sections, given the different aspects involved by each algorithm, notations may be introduced on a *per* requirement basis.

3 Evolutionary Algorithm Analysis Context

3.1 Encoding and Evaluation of the Conformations

The algorithmic resolution of the protein structure prediction problem, in heuristic context, is directed through the exploration of the molecular energy surface. The sampling process is performed by altering the sustaining structure of the considered molecule, i.e., backbone structure, associated torsional angles, etc., in order to obtain structural variations.

Different encoding approaches have been mentioned in the literature, including direct coding of atomic Cartesian coordinates [38], all-heavy-atom, C_α, backbone and residues , hydrophobic/hydrophilic lattice models relying on amino-acid based codings [17], etc. Amino-acids sequences derived representations are further employed in homology and threading based modeling [4], secondary structures being isolated by analogy with already known folds or structure databases, out of which the tertiary structure is determined.

For the case herein, an indirect, less error-prone, torsional angle based representation has been preferred. More specifically, each individual is coded as a vector of torsion angle values. The defined number of torsion angles represents the degree of flexibility. Apart from torsion angles which move less than a specified parameter value, e.g., within a specified interval, all torsion angles are flexible. Rotations are performed in integer increments, the intra-molecular atomic interactions being quantified by employing a Consistent Valence Force Field (CVFF) [16]. The resulting energy value is employed as a fitness evaluation, high values denoting atomic clashes and conformational inconsistencies while structural coherence is attained for low energy levels. Interactions are modeled as harmonic forces, the molecular conformation being regarded as a ball-and-spring elastic ensemble.

An extensive discussion on force fields designed for protein simulations, with in-depth details, is offered in Ponder and Case [37]. The first part of the mentioned work covers the evolution of the force fields, starting from the 1980s, and discusses various formulations which include the AMBER (Assisted Model Building and Energy Refinement), CHARMM (Chemistry at HARvard Macromolecular Mechanics) and OPLS (Optimized Potentials for Liquid Simulations) force fields. Another important referential work, Neumaier [35], encompasses a variate range of protein-related aspects, from the structure of proteins to algorithmic approaches and mathematical modeling.

3.2 Evolutionary Algorithms

Evolutionary algorithms are stochastic, iterative search techniques, inspired from Darwinian evolutionary theory, having a large area of appliance – epistatic, multimodal, multicriterion and highly constrained problems [9]. Stochastic operators are applied for evolving in an iterative manner an initial randomly generated population, i.e., following a generational concept. Each of the individuals composing the population contains *genotype* information encoding its defining features - the *phenotype*. Each generation undergoes a selection process, the individuals being evaluated by employing a problem specific fitness function.

The pseudo-code in Algorithm 1 exposes the generic components of an EA. The main subclasses of EAs are the genetic algorithms, evolutionary programming and evolution strategies.

Due to the nontrivial addressed problems, requiring extensive processing time, different approaches were designed in order to reduce the computational costs. Complexity is also addressed by developing specialized operators or hybrid and parallel algorithms. We have to note that the parallel affinity of the EAs represents a feature determined by their intrinsic population-based nature. In Cahon, Melab, and Talbi [8] three main parallel models are identified – the island synchronous cooperative model, the parallel evaluation of the population and the distributed evaluation of a single solution.

Algorithm 1 EA pseudocode.

```
$t \leftarrow 0$
Generate($P(0)$)
**while** $\neg$Termination_Criterion($P(t)$) **do**
    Evaluate($P(t)$)
    $P'(t) \leftarrow$ Selection($P(t)$)
    $P'(t) \leftarrow$ Apply_Reproduction_Ops($P'(t)$)
    $P(t+1) \leftarrow$ Replace($P(t), P'(t)$)
    $t \leftarrow t+1$
**end while**
```

Genetic algorithms (GAs) are population-based metaheuristics that allow a powerful exploration of the conformational space. However, they have limited search intensification capabilities, which are essential for neighborhood-based improvement (the neighborhood of a solution refers to part of the problem's landscape). Therefore, different approaches combine GAs with local search methods, in order to improve both the exploration and the intensification capabilities of the two techniques.

As the focus of this chapter mainly considers mutation operators, no further details are included – please refer to the work of Michalewicz [31] for references and further details. Additionally, for a comprehensive overview on parallel and grid specific metaheuristcs, refer to Cantu-Paz [10], Alba and Tomassini [1], Cahon, Melab and Talbi [9], Talbi [43], Alba, Luque and Melab [2] and Cahon et al. [8].

3.3 Mutation Operators

As the diversification characteristics of evolutionary algorithms are significantly determined by mutation, important information can be derived by analyzing the behavior of EAs as determined by these components. To the extent of our knowledge, no analogous large scale analysis and experimentation has been previously carried on the conformational sampling problem, although similar studies exist on artificial academic problems. Namely, refer to the surveys of Herrara, Lozano and Verdegay [21, 22, 23, 24] and Yao, Liu and Lin [48], addressing intensification crossover operators.

A general mutation operator acting on a solution $\gamma \in \mathbb{R}^N$ induces a perturbation as determined by an *a priori specified* or *implicit* distribution. Further, the particularities of each distribution determine the behavior of the operator. The most common example is the Cauch based mutation operator which, as compared to the Gaussian based one, should ensure for a lower probability of getting the exploration algorithm trapped in local optima due to larger generated deviates. AutoDock, in addition to relying on simple mutation operators (uniform, binary representation bit flip, etc.), also includes a Cauchy based operator; refer to the articles of Morris, Goodsell, Halliday, Huey, Hart, Belew and Olson [32] and Thomsen [47] for further details. A study on annealing schemes incorporated into mutation operators is also conducted in the latter.

Following a study conducted by Taveres, Tantar, Melab and Talbi [46], in a joint work, focused on mutation locality analysis, it seems sensible to introduce a distribution scaling factor as it reflects in the exploration-exploitation balance properties of the operator. Thus, a large scaling factor can result in highly perturbed solutions with a bias on exploration while reduced scaling factors may result in a tendency towards exploitation. A balance of the two aspects is required as the global exploration properties are derived from the exploration-exploitation tendency. An exploration inclined algorithm does not comport fast convergence characteristics whilst, at the opposite extreme, an exploitation biased algorithm is more prone in getting trapped in local optima.

Furthermore, *annealing schemes* may dynamically influence the behavior of the operators by acting on the impact of the generated deviates [47]. Note that annealing schemes can be applied on different control parameters, dynamically modifying the magnitude of the deviates, the standard deviation of the considered distribution, etc. Thus, the influence of the operator can be reduced as the exploration advances, resulting in less disrupting mutations near the final phases of the algorithm and focusing the search on a reduced region. Alternatively, an inverse annealing scheme can be employed where, instead of alleviating the operator's impact, the generated deviates are amplified, allowing for larger magnitude mutations. In the latter case, the mutation operator should ensure that the exploration algorithm, near the final stages of the search, is not allowed to block in local optima. Note that the amplified deviates may require to be counterbalanced by a more reduced mutation probability in order to allow the algorithm to converge.

Considering the previously stated, for γ a given solution, as a general formulation, a mutation operator can be defined as follows:

$$\tilde{\gamma} = \gamma + \eta \ \mathscr{D}(\mu, \sigma_t^2)$$

where η represents a pre-specified fixed scaling factor, $\mathscr{D}(\mu, \sigma_t^2)$ stands for a generic distribution with mean μ and σ_t^2 variance. In this context, the variance factor σ_t^2 is defined as a t generation-index dependent annealing scheme. As an example, in the work of [47], the following annealing schemes were employed (see Fig. 2 A):

$$\sigma_t^2 = \frac{1}{1+t} \qquad\qquad \sigma_t^2 = \frac{1}{\sqrt{1+t}}$$

In order to offer an improved control over the annealing scheme, the following designed formulation is employed for the following operators:

$$\sigma_t^2 = e^{\log(\omega)\cos\left[\frac{\pi}{2}\left(1-\frac{4st(t-G)-t(2t-3G)}{G^2}\right)\right]},\ 0 \le t \le G,\ 0 \le s \le 1$$

In the above expression, G denotes the maximum number of generations executed by the global exploration algorithm. Furthermore, ω stands for the target value to be

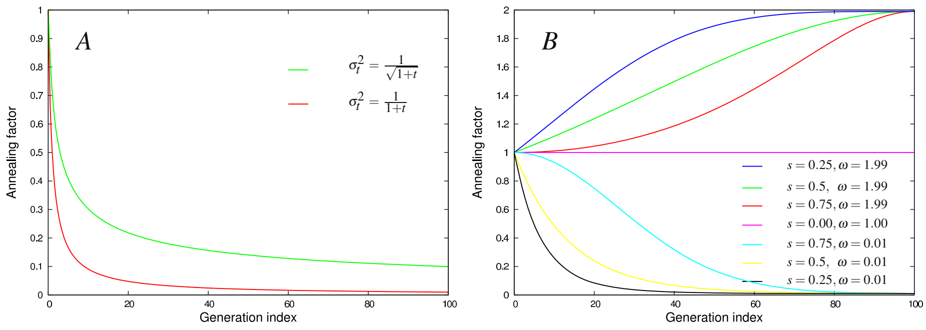


Fig. 2: **A** - Thomsen's annealing schemes for the mutation operator. **B** - A more flexible annealing scheme determined by s, a shape control parameter, and by ω, a final target value.

attained by σ_t^2, when $t = G$. Note that it is assumed that the σ_t^2 term is updated at each iteration, the generated values evolving from 1.0, for $t = 0$, to the specified ω value, for $t = G$. In addition, s offers control over the shape of the annealing scheme. A balance between a high rate of change and a more smooth transition is obtained when varying the s parameter in the $[0,1]$ interval. Depending on the value of ω, a more smooth annealing scheme is obtained when $0.5 \leq s \leq 0.75$. Refer to Fig. 2 for a graphical example depicting the annealing schedules.

Under the specifications, the following operators are considered for the designed approach – refer to the work of Michalewicz [31] and Thomsen [47] for further references and details:

Swap. For a solution γ, the γ_i, γ_j *loci* are interchanged, with $1 \leq i, j \leq N,\ i \neq j$ randomly (uniform) chosen indexes. Note that, while not being the case for PSP, restrictions may be imposed, as for molecular docking, due to the genotype's representation, e.g., it may not be allowed to interchange a translation *locus* with a torsional angle one.

Uniform. A uniform perturbation is applied to each *locus* γ_i (or to a randomly chosen set of *loci*), the resulting genotype $\tilde{\gamma}$ being defined as $\tilde{\gamma} = \gamma + \eta\ U[0,1]$.

Gaussian. The resulting solution is constructed as $\tilde{\gamma} = \gamma + \eta\ N(\mu, \sigma_t^2)$, where μ, σ_t^2 respectively denote the mean and the generation dependent variance of the distribution. For the most common employed cases the constant $\mu = 0.0$ and $\sigma_t^2 = 1.0$ values are specified for $0 \leq t \leq G$.

Cauchy. The Cauchy mutation operator relies on similar design principles, the perturbed solution being defined as $\tilde{\gamma} = \gamma + \eta\ C(\mu, \sigma_t^2)$.

In addition to the above operators a *Pearson-distribution system* based mutation is proposed. Naming in fact a class of distributions with various *skewness* and *kurtosis*, the Pearson system includes the Gamma, Beta, *t*-distribution, etc. In addition, the Normal distribution is also modeled as a special case of the Pearson system. Therefore, the Pearson distribution system represents a complex, highly flexible, parameter controllable mean of generating random deviates. References and an in depth discussion may be found in the article of Nagahara [34]. The four parameters which model the included distributions are the mean, the variance, the skewness and the kurtosis, further denoted as μ, σ^2, ς and κ, respectively. The type of the distribution is determined by the last two parameters, the skewness and the kurtosis, resulting in seven different classes denoted as *Pearson I* through *Pearson VII*. All the modeled classes, except the Pearson IV distribution, can be expressed by employing the Normal and the Gamma distributions. Transformation formulae for generating the Pearson I-III, V-VII distributions are presented in the article of Nagahara, additionally, an iterative rejection algorithm for the Pearson IV type distribution is proposed. Considering the complexity of the involved formalism standing as the basis for constructing the Pearson system, details and synthesis are not offered here. Additional information is provided in the previously mentioned article Nagahara [34], as well as in Nagahara [33] and Heinrich [20]. Following the specifications of Nagahara, a custom implementation of the sampling methods and the probability density functions has been constructed for the herein analyzed operators.

Noting a Pearson distribution system as $P(\mu,\, \sigma_t^2,\, \varsigma,\, \kappa)$, where, as previously defined, σ_t^2 represents a generation-index dependent variance, the modeled mutation operator can be defined as $\tilde{\gamma} = \gamma + \eta\ P(\mu,\, \sigma_t^2,\, \varsigma,\, \kappa)$, with η a scaling factor.

For all the here presented operators, three variants are considered, including a descending annealing scheme ($s = 0.25$, $\omega = 0.01$), an inverse annealing ($s = 0.25$, $\omega = 15.0$) and no annealing scheme ($s = 0.0$, $\omega = 1.0$). Furthermore, for the Pearson system all seven types are analyzed, hence there are 21 different operators. Adding the previously presented operators, with identical annealing schemes for the Gaussian and Cauchy based operators, 29 operators are studied overall. For the analysis details refer to Section 5.2.

4 Statistical Selection Procedures

As an important set of operators and parameters are considered for this study, the construction of a hybrid global exploration approach relies on non-negligible design decisions. A first problem consists of identifying the operators which comport superior minimization characteristics at the different stages of the search. A second question to answer relates to operators which induce a bias in the exploration process. Furthermore, multiple scenarios can be envisaged leading to dynamic or adaptive hybrid models for which operators are *a priori* or selected *online*, in an automatic manner. As a consequence it becomes of extreme importance to have the means for discriminating between the different envisaged models.

Selection procedures aim at identifying the *best system*, *e.g.*, for this case, operators, local search algorithms, etc., out of a finite set of alternatives [6]. In this context, *best* is inferred in terms of *minimum mean output*[4] for a series of *replications*, *i.e.* independent samplings of the compared systems. The main formulations of selection procedures, according to Branke et al. [5], Chick and Inoue [14] and Branke et al. [6], are: Indifference Zone (IZ) approaches, the expected Value of Information Procedures (VIP) and the Optimal Computing Budget Allocation (OCBA). The mentioned approaches are differentiated based on the assumptions employed when judging the evidence of correct selection. A comparison study [6] confronts several selection procedures. The OCBA procedure, initially developed by Chen [13] and Chen and Chick [12] has been here adopted for sustaining the analysis of operators and local search algorithms. An example of a hybrid model, combining an evolutionary algorithm with the OCBA selection procedure is presented by Schmidt et al. [40].

Note that, throughout the following, *minimization* problems (as for the mean output of the algorithms) are considered. The initial formalism has been hence adapted to reflect this consideration. Furthermore, it is *assumed* that the outcomes of the simulation process, derived out of independent executions, follow a *normal distribution*. Refer to Branke at al. [6] for details regarding the underlying assumptions which set the basis of the selection OCBA procedure.

A set of algorithms $\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_m$, $m \geq 2$, $1 \leq i \leq m$, $\mathbf{X}_i \in \Xi$ is considered, subject to undergo a selection process to identify the *best* algorithm. For an algorithm $\mathbf{X}_i \in \Xi$, given a series of simulations, i.e., independent executions, the outcome results are further denoted as $\mathbf{X}_{ij}$, $j \geq 1$. For the case addressed herein, as a notational assumption, it is considered that $\mathbf{X}_{ij} \equiv \hat{E}(\Gamma_{\mathbf{X}(\mathscr{S},\mathscr{L})})$, where $\mathscr{S}$, $\mathscr{L}$ respectively identify the input and the parameterization of the algorithm. Otherwise stated, for $\mathscr{S}$ a given input and for $\mathscr{L}$ a specific set of parameters, $\mathbf{X}_{ij}$ designates the *fitness value* of the best solution provided by the algorithm. Refer to Section 2 for notational details. Denoting μ_i, σ_i^2 as the mean and variance respectively of the $\mathbf{X}_i$ algorithm, based on the previous specifications, an ordering $\mu_{[1]} \geq \mu_{[2]} \geq \cdots \geq \mu_{[m]}$ can be constructed. Here, the $_{[\cdot]}$ permutation operator corresponds to an *unknown* ordering for which the algorithm $\mathbf{X}_{[m]}$ has the *lowest output mean*. Otherwise stated, the algorithm $\mathbf{X}_{[m]}$ offers, *in average*, the *lowest output values* which, given the minimization context, represent the desired result. Furthermore, for n_i consecutive executions of the $\mathbf{X}_i$ algorithm, the *recorded* mean and variance respectively are defined as follows:

$$\tilde{\mu}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} \mathbf{X}_{ij}, \qquad \tilde{\sigma}_i^2 = \frac{1}{n_i - 1}\sum_{j=1}^{n_i} (\mathbf{X}_{ij} - \tilde{\mu}_i)^2 \tag{10}$$

The $\tilde{\mu}_i, \tilde{\sigma}_i$ descriptive measures stand as estimates of the "*true*", unknown, μ_i, σ_i. Thus, the $\tilde{\mu}_{(1)} \geq \tilde{\mu}_{(2)} \geq \cdots \geq \tilde{\mu}_{(m)}$ ordering can be constructed, which, for

[4] Maximization problems were addressed in the original specifications; without any generality restrictions, by inversion, the same constructions can be employed by considering the *minimum mean output*.

$n_i \to \infty, 1 \le i \le m$, should result in the $\mu_{[1]} \ge \mu_{[2]} \ge \cdots \ge \mu_{[m]}$ ordering. From this point, assuming that independent executions are performed, at consecutive iterations, a measure quantifying the *Probability of Correct Selection* (*PCS*) has to be defined. In basic terms, for $\tilde{\mu}_{(1)} \ge \tilde{\mu}_{(2)} \ge \cdots \ge \tilde{\mu}_{(m)}$ one has to estimate the extent of the $_{(\cdot)}$ ordering "*maps*" on the $_{[\cdot]}$ permutation. Thus, based on the performed simulations, the probability of having $\mu_{(i)} > \mu_{(m)}$, for $1 \le i < m$ is estimated, i.e., the probability that the *predicted* algorithm $\mathbf{X}_{(m)}$ is indeed better (has a better mean) than all the other algorithms included in the test. The overall probability can be approximated by *Slepian's inequality*, shown below – refer to Branke et al. [6] for details. If (*explicitly*) no difference is made between two algorithms $\mathbf{X}_i, \mathbf{X}_j$ for which $|\mu_i - \mu_j| < \delta$, i.e., a *near optimal* comparison is sufficient, a measure of Probability of Good Selection ($PGS_{Slep,\delta}$) can be defined as:

$$\prod_{1 \le i < m} P\{\mu_{(i)} > \mu_{(m)}\} \approx PGS_{Slep,\delta} = \prod_{1 \le i < m} \Phi_{\nu_{(i)(m)}}(\varepsilon_{(i)(m)}^{1/2}(\delta + \tilde{\mu}_{(i)} - \tilde{\mu}_{(m)})),\ \delta \ge 0$$

$$\varepsilon_{a,b} = \frac{1}{\tilde{\sigma}_a^2/n_a + \tilde{\sigma}_b^2/n_b},\ \nu_{a,b} = \frac{(\tilde{\sigma}_a^2/n_a + \tilde{\sigma}_b^2/n_b)^2}{(\tilde{\sigma}_a^2/n_a)^2/(n_a-1) + (\tilde{\sigma}_b^2/n_b)^2/(n_b-1)}$$

Note: In the previous expression, $\Phi_{\nu_{a,b}}$ denotes the *cumulative distribution function of the standard t-Student distribution* with $\nu_{a,b}$ degrees of freedom. The *PCS* measure is obtained by setting $\delta = 0$ ($PCS_{Slep} \equiv PGS_{Slep,0}$).

Hence, for a given ordering of the output means, based on the posterior distributions, the probability of correct selection can be estimated. Another desideratum to attain at this stage consists of minimizing the number of samplings required to obtain a significant PCS/PGS level. The OCBA selection procedure initially performs a number of τ_{init} simulations for *all* the algorithms to compare. If a high enough PCS/PGS level is not obtained, a number of τ_{adt} additional samplings are heuristically allocated, in iterative manner, to a number of τ_{sys} algorithms. The τ_{sys} algorithms are selected by computing an *Estimated Approximate Probability of Correct Selection* (*EAPCS*), for each algorithm $\mathbf{X}_i,\ 1 \le i \le m$:

$$EAPCS_s = \prod_{1 \le i < m} \Phi_{\tilde{\nu}_{(i)(m)}}(\tilde{\varepsilon}_{(i)(m)}^{1/2}(\tilde{\mu}_{(i)} - \tilde{\mu}_{(m)})),\ \tilde{\varepsilon}_{a,b}^{-1} = \frac{\tilde{\sigma}_a^2}{n_a + \tau_{adt}\zeta_{a,s}} + \frac{\tilde{\sigma}_b^2}{n_b + \tau_{adt}\zeta_{b,s}}$$

$$\tilde{\nu}_{a,b} = \frac{(\tilde{\sigma}_a^2/n_a + \tilde{\sigma}_b^2/n_b)^2}{(\tilde{\sigma}_a^2/n_a)^2/(n_a-1) + (\tilde{\sigma}_b^2/n_b)^2/(n_b-1)},\ \zeta_{a,b} = \begin{cases} 1, & a = b \\ 0, & a \ne b \end{cases}$$

It follows that it suffices to iteratively allocate samplings for the algorithms which maximize the $EAPCS_s - PGS_{Slep,\delta}$ measure, until a satisfactory confidence level is attained for the determined ordering of algorithms.

A parallel construction of the algorithm has been developed and employed here, relying on the ParadisEO framework, the basic OCBA pseudocode of the parallel implementation that was shown in Algorithm 2.

The algorithm is executed by first setting a δ indifference threshold and the desired confidence level τ_{conf}. In a second step the sampling parameters are set:

Algorithm 2 OCBA algorithm pseudocode.

```
1: Set $\delta$, $\tau_{conf}$ (indifference threshold and confidence level to be attained, respectively)
2: Set $\tau_{init}$, $\tau_{sys}$, $\tau_{adt}$ (number of initial samples, number of algorithms to re-sample, number of additional samples)
3: Execute in parallel $m * \tau_{init}$ independent simulations. A number of $\tau_{init}$ samples are obtained for each of the $\mathbf{X}_i$, $1 \leq i \leq m$ algorithms.
4: For all $\mathbf{X}_i$, $1 \leq i \leq m$ set $n_i \leftarrow \tau_{init}$ and compute $\tilde{\mu}_i, \tilde{\sigma}_i$.
5: Construct the initial ordering $\tilde{\mu}_{(1)} \geq \tilde{\mu}_{(2)} \geq \cdots \geq \tilde{\mu}_{(m)}$.
6: while stopping criterion not met do
7:     Compute $EAPCS_i$ $1 \leq i \leq m$ and construct a set $\mathscr{X}$ of $\tau_{sys}$ algorithms maximizing the $EAPCS_i - PGS_{Slep,\delta}$ difference. For each algorithm $\mathbf{X}_s \in \mathscr{X}$ allocate $\tau_{adt}$ additional samplings.
8:     Execute in parallel $\tau_{sys} * \tau_{adt}$ independent executions for the algorithms $\mathbf{X}_s \in \mathscr{X}$. A number of $\tau_{sys} * \tau_{adt}$ additional samples are obtained.
9:     For all $\mathbf{X}_s \in \mathscr{X}$ set $n_s \leftarrow n_s + \tau_{adt}$ and update $\tilde{\mu}_s, \tilde{\sigma}_s$.
10: end while
```

τ_{init}, τ_{sys} and τ_{adt} for the initial number of samplings (to be performed for each algorithm), respectively the number of algorithms to be additionally sampled and the number of samples to request at each iteration. A first independent parallel sampling is performed, following a multi-start model (line 3), the descriptive measures $\tilde{\mu}_i, \tilde{\sigma}_i^2$, $1 \leq i \leq m$ are computed (line 4) based on the retrieved data. An initial ordering is constructed and the algorithm ends the execution if the associated *PCS/PGS value* reaches, at least, the specified confidence level. In case the obtained ordering is not significant, additional samples are allocated in an iterative manner (lines 6-10).

At each iteration the $EAPCS_i - PGS_{Slep,\delta}$, $(1 \leq i \leq m)$ difference is computed, a number of τ_{adt} additional samples being requested for the τ_{sys} most promising algorithms (maximizing the difference). Subsequently, for the selected algorithms, a parallel sampling is performed (line 8), the n_i, $\tilde{\mu}_i, \tilde{\sigma}_i^2$, $1 \leq i \leq m$ measures being updated accordingly.

The algorithm finishes when a maximum number of iterations is reached (which is equivalent to a budget allocation analogy, considering $\sum_{1 \leq i \leq m} n_i < B$ as the constraint, where B represents a maximum affordable budget) or when a high enough confidence level is reached ($PGS_{Slep,\delta} > 1 - \tau_{conf}$).

5 Analysis of Mutation Operators

Depending on multiple factors, as the considered instances, the nature and quality of the initial solutions, the exploration stage, etc., the landscape to be addressed may comport significantly different particularities. Hence, it may prove to be of interest to employ multiple specific and adapted operators, capable of exploiting the characteristics of the explored local landscape. Furthermore, the analysis of the previously presented operators may provide important information, allowing to later define dynamic and adaptive exploration systems. Incipient directions as conclusions following the analysis study, to be detailed in a following work.

As a large number of mutation operators is considered, a critical analysis study is required in order to identify the operators which comport significant energy or RMSD bias and minimization characteristics. A ranking of the employed operators can be constructed using the previously presented statistical selection procedure, possibly isolating the *best* operator(s). In this context, the term *best* is employed under statistical confidence assumptions and as determined by the considered particular benchmarks and parameters setup – no *absolute best* is provided as a result. The conformational sampling process is conducted on energy minimization criteria, aiming to find the conformational structure with the lowest RMSD. Hence, *best* can be used to nominate the operator with significant energy minimization characteristics and a strong RMSD bias, leading towards the crystallographic native structure.

As mentioned in the introduction of this chapter, a previous study has been conducted by Tavares et al. [46], as a joint work, with a focus on the locality analysis of mutation operators. For that study, an independent, stand-alone component perspective was followed, the operators being analyzed in a disconnected, offline manner, i.e., no evolutionary algorithm was employed. In contrsast, the aspects addressed herein do not aim to describe, in an independent manner, the constructed operators. Instead, a simplified EA is employed as an embedding environment for the operators, in order to construct a heuristic based analysis context. The details of the evolutionary algorithm, on a per case basis, are described in the following experimentation sections, the parameters setup being also discussed.

Assuming the aforementioned considerations, the analysis accounts for the effects induced by the operators when placed inside a heuristic context. First the output energy values are used as the criterion and, in a second phase, the RMSD of the resulting conformations is employed. Note that none of the analyzed components include RMSD information or minimization mechanisms, relying exclusively on energy evaluations as the guiding criterion. Furthermore, for the later analysis case, the *less* destructive instead of the *best* RMSD operator is identified. Hence, the second case is only applicable for the considered benchmark conformations (1L2Y, 1LE1, α-cyclodextrin – refer to Section 5.1), for which the crystallographic structure is known. As no *a priori* information on the native structure is provided for the general case, the results extracted in the RMSD analysis phase only serve as a second criterion for the energy-minimization ranking. For example, it may be preferable to use a (energy) second rank component with low RMSD destructive features than a (energy) first rank component with high RMSD destructive characteristics.

For each analysis phase, energy and RMSD, three distinct sections are considered, the embedding evolutionary algorithm having first as input, random conformations, secondly, optimized conformations and final, near-global conformations. Each of the three studied sections corresponds to a different stage of the exploration process: random conformations for the early execution phases of the algorithm, optimized conformations for the intermediary steps of the exploration strategy and near-global conformations, assuming convergence, addressed in the final execution stages. For each stage, solutions are generated as follows:

Random solutions. For each torsional angle *locus* $\gamma_i \in \gamma$, $1 \leq i \leq N$ a uniform random value out of the $[0, 360]$ interval is assigned.

Optimized solutions. A random solution γ is initially uniformly generated; in a second phase, the obtained solution undergoes several optimization steps, multiple local search methods (ASA [27, 28], L-BFGS [7, 49], conjugate gradient [26, 25], etc.) being sequentially applied on the initial γ encoding. A set of 3000 solutions is thus constructed. Elements of the set are then uniformly drawn, as required for the analysis.
As a consequence of undergoing multiple minimization steps, an *optimized* solution can be described as a strong local optimum, with no quality assumptions as compared to the crystallographic conformation.

Near-global solutions. By convention of implementation, the reference crystallographic conformation is represented as a solution γ having all *loci* $\gamma_i \in \gamma$, $1 \leq i \leq N$ set to 0. Consequently, a near-global solution (from the RMSD perspective) can be generated by considering the surrounding region. Thus, for this particular case, solutions are uniformly generated out of the $[-10, 10]$ interval. Afterwards the obtained angle values are mapped onto the $[350,\ 360) \cup [0,\ 10]$ domain.

As the crystallographic conformation encoding is represented as a vector of null values, $\gamma = ((0))$, one particular aspect has to be addressed. Depending on the nature of each operator, e.g., type of calculations, numerical instability, a bias towards generating a null vector may exist. Having a null vector bias, for the problem studied herein, can be considered as an equivalent of *a priori* inserted knowledge. In order to not account for the null vector bias effect in the selection procedure, a penalizing factor is introduced for the energy evaluation function. Thus, for all solutions γ with an Euclidean distance $d_\gamma = \|\gamma - \gamma^*\| < 30.0$, where γ^* represents the encoding of the crystallographic conformation, the energy evaluation is set to $E(\gamma) + (1.0e + 15) \times \exp(-d_\gamma^2)$. The penalizing effect is hence progressively reduced with the increase of the Euclidean distance – the impact is significant only in the close vicinity of the γ^* solution. Consequently, a greater penalty is applied for the operators or algorithms which have an express tendency of converging towards a null vector solution.

Note that the encoding of the crystallographic conformation *does not* necessarily have associated the *global minimum energy*. High energy – low RMSD conformations may exist while, at the opposite end, low energy conformations (below reference conformation energy) may have a high RMSD. This particularity represents the main reason sustaining the RMSD bias analysis, besides the energy minimization characteristics study. Thus, for the energy and RMSD analysis and for the three different phases addressing different quality solutions, a total of six distinct categories to be analyzed result.

The rest of this section is organized as follows: first the benchmarks employed for the analysis section are briefly presented; secondly analysis results and discussions are provided. The analysis setup is detailed, numerical results and graphical equivalent representations being included for each analysis stage.

5.1 Conformational Sampling Benchmarks

The molecular complexes for the conformational sampling algorithms assessment are *tryptophan-cage* (trp-cage - Protein Data Bank ID: 1L2Y), *tryptophan-zipper* (trp-zipper - Protein Data Bank ID: 1LE1) and *α-cyclodextrin*. The *trp-cage*, *trp-zipper* belong to the class of mini-proteins presenting particularly fast folding characteristics. *Cyclodextrins*, in α, β or γ conformations (containing 6, 7, 8 glucose units, respectively), due to their toroidal structure, are important for drug-stability applications, being used as protectors against micro-environment interactions or as homogeneous distribution stabilizers etc.

The selected benchmark conformations can be considered, to a certain extent, as being significant and representative as they comport different structural patterns. Hence they require a flexible enough algorithm to predict the different enclosed secondary structures. Refer to Fig. 3 for a graphical representation of the three molecular conformations. A schematic equivalent representation is also exposed in order to better exemplify the structural characteristics of each molecule (as the cyclic structure of α-cyclodextrin). The *tryptophan-cage* protein can be structurally characterized as having an α-helical N-terminal region, a short helix and a *polyproline* helix enclosing a *Trp* residue inside a hydrophobic core [30].

The *tryptophan-zipper* belongs to the class of so called β-hairpins, for which, opposing *Trp* residues form a non-hydrogen-bonded zipper [41]. *Cyclodextrins*, non-reducing macrocyclic *oligosaccharides*, build toroidal structures with a hydrophobic interior. The α-cyclodextrin molecule, while not being a protein, has been included due to its particular cyclic structure.

In addition, the addressed conformations, given the number of defined torsional angles, namely 64, 54, 73 angles for α-cyclodextrin, 1LE1, 1L2Y, respectively, offer the advantage of not requiring an extremely expensive energy evaluation computational time. Thus, with the support of parallel and distributed computing, multiple analysis studies, with varying parameters, can be performed in an acceptable time frame.

5.2 Experimentation and Analysis Results

As a first remark, no previous optimization, of any nature, has been carried out on the parameters of the operators. Secondly, accounting for the large number of operators, resulting in an even larger number of parameters, no extensive study is

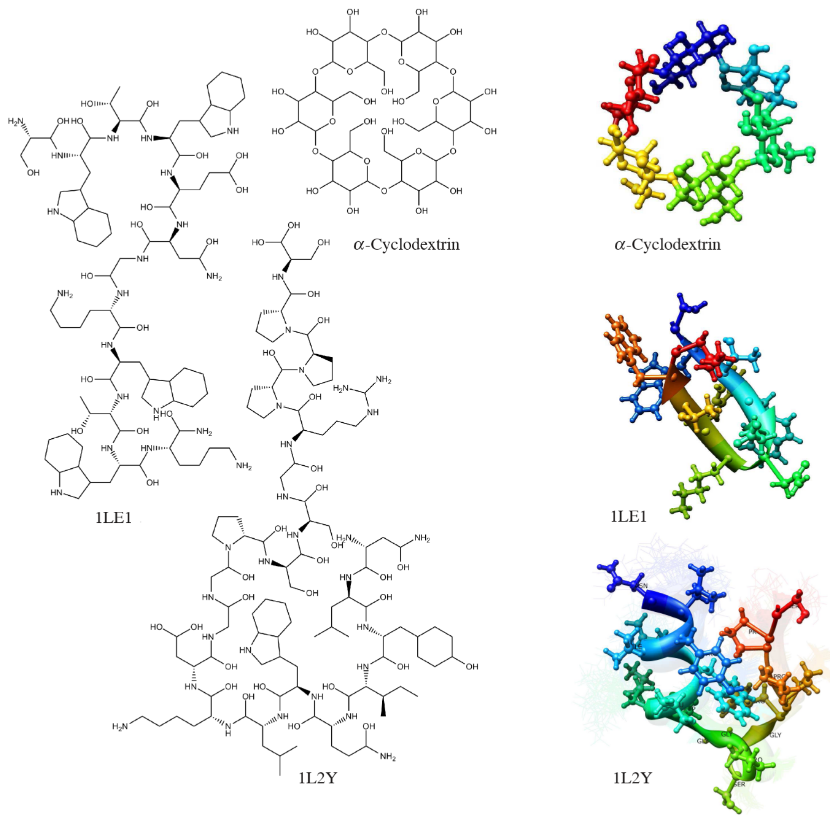


Fig. 3: Conformational sampling considered benchmarks: α-cyclodextrin, trp-cage (1L2Y) and trp-zipper (1LE1). A β-sheet and an α-helix (schematically depicted on the bottom row of the figure) can be distinguished in the graphical representation of the 1LE1 and 1L2Y proteins respectively.

possible. Nevertheless, the analysis performed may offer important information on the *classes* of operators which potentially exhibit interesting characteristics. In the following, addressing the experimental setup, either the classically recommended parameter values are set or extreme opposing values are specified, so as to evaluate the behavior of the operators under different scenarios.

Evolutionary Algorithm Setup. A basic EA is used as an embedding environment for testing the mutation operators. The algorithm is set to evolve an initial population of 150 solutions for 150 generations. At each iteration of the algorithm, the current population undergoes a stochastic tournament selection process, 150 individuals being chosen out of the population. At each selection step, the stochastic tournament component is set to return, out of two (uniform) randomly chosen solutions, the best individual, with a probability of 0.75. The resulting solutions become further subject to mutation, no crossover operator being employed. The embedded mutation operator is applied with a probability of 0.1, resulting in 2250 mutations for a complete execution (on average). Further, generational replacement is used, the entire population being replaced by the individuals resulting out of the mutation phase. A weak elitism scheme ensures that the best individual in the population to be replaced survives the replacement phase.

In order to offer a more consistent comparison basis, a null mutation operator has been introduced, designated in the following as NullM. This particular component in no way affects the solutions subjected to the mutation phase, solely standing as a reference for the analysis study. Note that, the resulting embedding EA conducts the exploration on the selection and the replacement schemes alone – no selection pressure is directly determined by the mutation operators.

The following labeling is used to designate the operators: SM for the Swap mutation, UM Uniform mutation, G for the Gaussian and C for the Cauchy based operators. Pearson system based mutations are nominated as P1 through P7 corresponding to Pearson type I to Pearson type VII distribution based operators. In addition, a prefix is added for the Gaussian, Cauchy and Pearson operators, indicating the type of annealing scheme employed: NA for no annealing, 01 for an annealing scheme with a 0.01 target value, and 15 for an inverse annealing, with a final value of 15.0 – refer to Section 3.3 for details. Consequently, the obtained code names are formed by the operator's designation and the annealing scheme label. As an example, for the Gaussian operator, the G-NA, G-01 and G-15 labels are used to refer to a Gaussian operator with no annealing scheme, an operator with a 0.01 final value for the annealing scheme and an operator with an inverse annealing having a target value of 15.0. In addition to employing annealing schemes, a fixed scaling factor of 30.0 has been specified for all operators. Therefore, for random deviates ranging in the $[0,1]$ interval, angle steps of at most 30.0 degrees are generated. As determined by the employed distribution and the annealing factor, which in this case acts on the operator's distribution variance, larger magnitude deviates may be generated.

The Gaussian, Cauchy, as well as all the Pearson based operators are defined as having the mean $\mu = 0$. In concordance with the Pearson distribution type, the following skewness and kurtosis values are set: $\varsigma = 1.0$, $\kappa = 4.0$ - P1; $\varsigma = 0.0$, $\kappa = 2.5$ - P2; $\varsigma = 1.0$, $\kappa = 4.5$ - P3; $\varsigma = 1.0$, $\kappa = 5.5$ - P4; $\varsigma = 1.0$, $\kappa = 4.97$ - P5; $\varsigma = 1.0$, $\kappa = 4.8$ - P6; $\varsigma = 0.0$, $\kappa = 4.0$ - P7.

Note that, while all the presented operators are ranked and analyzed, due to transience reasons, part of the operators had to be discarded from the numerical results and histograms. Namely, most of the 01 annealing scheme operators are not listed in the numerical results tables, given the poor fitness values and the afferent rankings. An exception is the P5-01 operator, scored second for one of the tests. The general failure of this particular class of operators can be partially explained by a too reduced magnitude of the generated deviates, rendering the EA algorithm unable to evolve the initial population more than determined by the selection and replacement schemes. Nevertheless, as previously stated, the constructed rankings, with the corresponding PGS values, are valid for the entire ensemble of operators.

Statistical Selection Procedure – OCBA. The parameters of the selection procedure can be regrouped as $\tau = (\tau_{init}, \tau_{adt}, \tau_{sys}, \tau_{conf})$ where τ_{init} defines the number of initial simulations to be performed, τ_{adt} represents the number of additional samplings allocated for the τ_{sys} heuristically selected systems. The desired confidence level to be attained is specified by the τ_{conf} factor. Additionally, a δ *indifference threshold* and a maximum number of iterations are specified. The *Probability of Correct Selection (PCS)* is estimated for $\delta = 0$ while, for $delta > 0$ the *Probability of Good Selection* is returned. Refer to Section 4 for details. For the analysis herein, 30 initial simulations (τ_{init}) were executed for each of the resulting EAs, in case of non-convergence, 25 additional replications (τ_{adt}) being requested for the 5 most promising mutation operators (τ_{sys}), based on the results of the afferent EAs. In addition a 1.0 indifference threshold (δ) has been set, i.e., no difference is considered for two mutation operators which offer results comparable within a maximum absolute difference of 1.0. The confidence level (τ_{conf}) has been set to 0.1 – a ranking of the operators is considered significant if a PGS value equal or superior to 0.9 is obtained. For details of the algorithm refer to Section 4. Furthermore, a maximum of 30 iterations has been specified.

As for each launched algorithm an array of conformations is returned after execution, the resulting replication value to be provided for the selection procedure is derived out of the lowest energy or RMSD encoding. As a consequence, the statistical measures presented in the numerical results tables and further exposed in the histogram, describe the energy or RMSD distribution of the best conformations provided by the EAs. As an example, the *Max* column *does not* refer to the maximum value obtained in the population after executing a specific EA but to the maximum value of the best provided solutions for the specific analysis case, e.g., 1L2Y Energy - Random analysis for P5-01.

For each analysis case (1L2Y Energy - Random, 1L2Y Energy - Optimized, etc.) *at least* three independent tests were performed. As for the three considered benchmarks a total of 18 analysis cases are analyzed, it follows that more than 54 independent tests were launched. Considering that each algorithm performs on average 2400 conformation energy evaluations (~2250 mutated solutions + 150 initial solutions) and that *at least* 48600 EAs are independently launched (30 initial samplings for each of the 30 operators, for each of the 54 tests), it follows that a total of more than 1.1e+8 conformation energy evaluations were computed. For the mutation analysis alone, the amount of generated raw data cumulated to almost 350MB of data.

Analysis of the Selection Results. Following the execution of the selection procedure, including both energy and RMSD studies, numerical statistical data has been synthesized in Tables 1, 2 and 3 for 1L2Y, 1LE1 and α-Cyclodextrin, respectively. Corresponding histograms, depicting the obtained results for the best three ranked mutation operators and constructed out of the (for comparison) sampled conformations, are given in Fig. 4, Fig. 5 and Fig. 6. Abscissae axis stands for normalized energy levels while the ordinates axis denotes the number of conformations.

Each table is vertically divided into two sections, afferent to energy and RMSD analysis respectively, superposed on three horizontal sections – random, optimized and near-global initial solutions. Thus, the six distinct cases under study are delimited, for each case and for each mutation operator, the minimum, maximum, average and standard deviation obtained values being listed. As an example of reading the tables, for the 1L2Y protein (Table 1), the best ranked operator on random conformations for the energy analysis test may be found under the **Tryptophan-Cage (1L2Y) - Energy** vertical section, in the first horizontal section of the table. P3-15 is ranked first, P4-15 second and P6-15 third. Note that, for most of the cases, no clear distinction can be made between the operators by following the histograms alone, hence, requiring at times to rely on the numerical results.

As previously mentioned, multiple tests were performed for each case. While only the highest obtained PGS is marked on the histograms, the operators designated by the afferent ranking being depicted, the numerical results and the equivalent graphical representations account for all the tests performed for the considered case. Thus, the listed average values may not follow the highest ranking, as influenced by tests terminated with a lower PGS.

As a reminder, the computed PGS value is valid and stands *only* for the designated *best* system and *not* for the entire resulting ranks array. Consequently, for two independent tests, the second and the third ranked methods may not be identical. Nevertheless, relating to the two criteria to be analyzed, more than one operator can be selected. As previously discussed, only the energy minimization ranking is to be considered – the RMSD results are included as to sustain or to reject the selection of the designated energy best operator. Hence, for each analysis case, only the energy best three mutation operators are marked in the presented tables, according with the corresponding rank; the afferent row is listed in boldface.

Table 1: 1L2Y - Numerical results for the mutation operator. For each stage (random, optimized and near-global solutions), energy and RMSD values are shown, the best three ranked operators (energy criterion), being marked in boldface with their rank.

		Tryptophan-Cage (1L2Y) - Energy				Tryptophan-Cage (1L2Y) - RMSD			
		Min	Max	Avg	StD	Min	Max	Avg	StD
Random Confs	NullM	1625.76	7252.17	4100.34	102.83	8.23	24.34	13.70	0.25
	SM	280.35	517.46	358.66	1.64	6.73	11.68	8.76	0.08
	UM	280.94	926.80	452.26	5.21	7.19	21.24	11.91	0.21
	G-NA	408.83	4579.33	801.08	15.75	6.07	20.10	12.17	0.24
	G15	279.04	697.04	399.19	3.00	6.44	13.52	8.93	0.11
	C-NA	402.38	1562.63	730.56	11.26	6.91	19.93	12.56	0.20
	C15	277.44	689.11	399.69	2.95	6.85	11.32	8.85	0.08
	P1-NA	302.07	1260.82	524.11	8.31	7.17	18.16	11.96	0.20
	P1-15	252.95	568.12	369.54	2.23	7.03	11.54	8.85	0.08
	P2-NA	373.16	1330.24	618.02	9.48	6.96	21.22	12.38	0.25
	P2-15	284.81	639.45	370.21	2.41	6.20	11.98	8.82	0.09
	P3-NA	272.75	774.24	415.18	4.46	7.28	17.33	11.66	0.19
	P3-15^1	**242.66**	**499.97**	**321.05**	**1.31**	**6.80**	**17.31**	**10.24**	**0.17**
	P4-NA	266.31	954.46	419.38	4.51	6.18	17.53	11.25	0.20
	P4-15^2	**240.36**	**555.21**	**333.72**	**1.53**	**6.43**	**16.85**	**9.78**	**0.14**
	P5-01	859.31	5458.39	2704.33	41.19	7.41	24.29	13.52	0.25
	P6-NA	261.16	966.90	422.28	4.10	7.38	20.69	11.88	0.21
	P6-15^3	**251.65**	**550.02**	**338.02**	**1.49**	**7.11**	**14.61**	**9.54**	**0.12**
Optimized Confs	NullM	65.79	93.51	81.34	0.38	6.12	16.49	10.12	0.23
	SM	65.78	94.30	81.85	0.20	5.78	11.11	8.01	0.09
	UM^1	**65.66**	**94.95**	**79.84**	**0.22**	**5.70**	**15.12**	**9.02**	**0.19**
	G-NA	65.77	94.26	80.26	0.21	5.47	15.54	9.86	0.21
	G15	65.78	94.26	80.98	0.22	5.97	12.26	8.02	0.10
	C-NA	65.79	94.26	81.10	0.21	5.98	15.46	9.45	0.20
	C15	65.76	94.87	80.75	0.21	5.93	11.02	7.97	0.09
	P1-NA	65.79	95.54	81.33	0.22	5.49	15.88	8.85	0.19
	P1-15	65.75	94.00	80.99	0.22	5.89	10.92	8.01	0.10
	P2-NA	65.79	92.19	81.05	0.22	5.95	14.59	9.31	0.20
	P2-15	65.74	95.67	81.40	0.23	5.46	10.68	7.70	0.08
	P3-NA	65.79	93.21	81.24	0.22	5.98	14.30	8.72	0.16
	P3-15	65.73	94.52	80.92	0.22	5.51	13.91	8.31	0.15
	P4-NA	65.79	95.00	81.78	0.25	5.84	14.50	8.79	0.17
	P4-15^3	**65.72**	**92.98**	**80.24**	**0.24**	**5.88**	**12.98**	**8.46**	**0.14**
	P5-01^2	**65.37**	**94.79**	**79.89**	**0.25**	**5.78**	**15.96**	**9.58**	**0.20**
	P6-NA	65.79	95.54	81.33	0.25	5.98	13.98	9.01	0.17
	P6-15	65.79	94.04	81.08	0.22	5.61	12.86	8.02	0.11
Near-Global Confs	NullM	1.93e+14	2.67e+14	2.40e+14	9.55e+11	0.78	3.96	1.89	0.06
	SM	2.03e+14	2.68e+14	2.40e+14	6.88e+11	0.52	1.28	0.85	0.01
	UM	166.64	979.49	286.04	3.83	2.14	13.34	7.13	0.17
	G-NA	2.35e+05	2.23e+09	3.36e+08	1.50e+07	1.39	10.52	4.95	0.16
	G15	209.22	484.10	290.11	1.83	4.51	11.31	7.89	0.10
	C-NA	320.06	2.17e+08	2.16e+07	1.24e+06	1.16	11.46	5.35	0.17
	C15	201.40	524.89	292.21	2.15	4.29	10.61	7.76	0.09
	P1-NA	212.44	3.39e+04	716.02	103.73	2.76	12.94	7.06	0.17
	P1-15	212.76	423.64	283.72	2.00	5.12	11.16	7.93	0.10
	P2-NA	806.87	3.46e+05	5.77e+04	3247.46	2.62	11.55	6.02	0.17
	P2-15	195.26	421.58	287.24	1.62	3.72	10.96	7.87	0.10
	P3-NA^3	**188.01**	**835.77**	**253.24**	**1.98**	**2.87**	**12.26**	**7.09**	**0.17**
	P3-15^1	**176.79**	**330.44**	**243.71**	**1.01**	**3.19**	**12.87**	**8.70**	**0.16**
	P4-NA	177.66	670.44	264.06	2.30	2.54	12.91	6.92	0.18
	P4-15^2	**186.77**	**344.45**	**251.43**	**1.32**	**4.11**	**13.55**	**8.32**	**0.14**
	P5-01	2.18e+13	1.61e+14	1.38e+14	8.89e+11	4.54	10.54	7.65	0.11
	P6-NA	189.97	939.27	269.06	2.97	3.27	15.02	7.05	0.19
	P6-15	186.73	379.94	258.72	1.42	2.46	12.77	8.39	0.13

A first element to notice is that, as compared to the null mutation EA (NullM), addressing energy minimization, the defined operators have a significant improvement contribution. Large differences of the average obtained values can be observed for all the three considered phases – random, optimized and near-global conformations. An important exception is represented by the class of operators relying on the 01 annealing scheme, as previously mentioned, only one operator of this class is listed in the numerical results tables, namely the P5-01 operator. In addition, the SM operator determined a degradation of the exploration process, scoring bellow or close to the null mutation operator for the final section, for all the three considered benchmarks.

A similar, less noticeable, improvement effect is obtained when having RMSD as criterion - statement valid for the first two phases, employing random and optimized solutions. Nevertheless, no important RMSD minimization is obtained, the large majority of the operators comporting a highly destructive effect - distinctively visible for the last phase. A particular exception is the SM operator which not only did not result in a degradation, as compared to the null mutation, but *constantly* offered an improvement. This is distinctly visible for the last section, relying on near-global solutions.

An analogous behavior can be associated also with the P5-01 operator, for both mutations, P5-01 and SM, the antagonism relies in the rank obtained for the energy section as compared to the RMSD one. In fact, at a more attentive examination of the extracted data, neither SM nor P5-10 is able to delocalize the search from the initial specified region. Hence, the EA is not able to escape the initial range of $[-10, 10]$ employed for generating the near-global solutions, consequently remaining in the close vicinity of the crystallographic conformation. Thus, instead of observing a bias, an operator's non-effectiveness is described by the numerical results. As a conclusion, for this case, it can be stated that the selection procedure offers a *correct* result but with *erroneous* semantics.

A second remark has to be made on the Pearson system based mutation operators. The obtained rankings are *dominated* by the Pearson based operators, especially when considering the energy minimization criterion. This comes to sustain the choice of using the Pearson system of distributions and also to reinforce the idea of employing the correspondingly defined operator as part of the adaptive schemes. As the Pearson system can mimic a large number of distributions, the resulting operator represents an ideal candidate for this specific type of approach. As the operator can be completely controlled by affecting the four afferent parameters (mean, variance, skewness and kurtosis), minimal mechanisms have to be designed in order to build dynamic operators.

Note that, as compared to the desired confidence level, significant PGS levels were attained, for all the analyzed cases, with one exception. Recall that a minimal PGS level of 0.9 has been specified so as to retain and consider the provided ranking as being significant. The exception is given by the 1L2Y benchmark, optimized solutions case, for which a PGS of only 0.342456 has been reached – see Fig. 4.

Table 2: 1LE1 - Numerical results for the mutation operator. For each stage (random, optimized and near-global solutions), energy and RMSD values are shown, the best three ranked operators (energy criterion), being marked in boldface with their rank.

		Tryptophan-Zipper (1LE1) - Energy				**Tryptophan-Zipper (1LE1) - RMSD**			
		Min	Max	Avg	StD	Min	Max	Avg	StD
Random Confs	NullM	792.34	6548.79	2564.11	121.01	7.97	19.75	12.93	0.23
	SM	176.54	313.18	229.93	2.73	5.91	11.47	8.89	0.10
	UM	152.28	349.86	241.59	4.37	7.22	17.25	11.50	0.17
	G-NA	245.65	711.80	377.73	8.90	6.02	16.25	11.97	0.18
	G15	187.85	312.81	228.81	2.49	6.05	12.84	9.30	0.11
	C-NA	216.39	701.92	348.53	8.80	7.17	20.13	12.20	0.20
	C15	171.21	319.58	239.09	3.14	6.14	12.15	8.91	0.12
	P1-NA	166.96	405.68	252.17	4.85	7.22	15.83	11.62	0.16
	P1-15	175.14	328.68	231.98	3.11	6.25	11.67	8.94	0.10
	P2-NA	218.92	492.52	311.33	6.92	6.38	15.35	11.34	0.19
	P2-15^3	**170.06**	**282.96**	**221.45**	**2.71**	**4.99**	**11.57**	**8.88**	**0.11**
	P3-NA	138.38	303.01	223.98	3.28	6.93	14.92	11.00	0.17
	P3-15^1	**147.03**	**256.81**	**193.04**	**2.21**	**6.73**	**15.23**	**10.13**	**0.16**
	P4-NA	167.85	355.14	229.11	4.36	7.06	15.49	11.09	0.16
	P4-15^2	**159.55**	**270.66**	**206.76**	**2.67**	**6.85**	**13.25**	**9.93**	**0.13**
	P5-01	616.74	1.50e+06	1.80e+04	1.66e+04	6.99	17.67	12.65	0.21
	P6-NA	166.04	344.83	234.16	4.36	6.62	16.15	11.01	0.17
	P6-15	162.98	267.62	211.77	2.70	6.01	13.68	9.54	0.14
Optimized Confs	NullM	484.07	3317.61	1140.38	54.88	7.49	18.77	13.07	0.24
	SM^1	**114.14**	**202.42**	**157.47**	**1.83**	**5.38**	**13.36**	**9.03**	**0.13**
	UM	142.02	242.66	185.24	2.15	6.16	17.71	11.41	0.19
	G-NA	180.17	328.76	246.40	3.96	7.61	18.32	12.12	0.20
	G15	155.50	279.64	195.27	2.64	6.83	11.78	9.24	0.11
	C-NA	181.01	371.31	237.48	3.73	7.69	16.73	11.92	0.19
	C15	152.32	235.31	193.00	2.08	5.95	12.08	8.93	0.12
	P1-NA	149.16	256.44	196.03	2.63	6.81	16.36	11.46	0.18
	P1-15	144.77	272.31	194.54	2.54	5.91	12.25	8.74	0.12
	P2-NA	158.07	294.70	218.65	3.27	7.13	21.13	11.64	0.22
	P2-15	147.45	242.74	190.62	2.05	6.11	11.77	8.91	0.10
	P3-NA	135.35	221.82	178.63	2.29	7.04	16.28	11.18	0.17
	P3-15^2	**135.99**	**222.63**	**174.09**	**1.74**	**6.03**	**14.43**	**10.29**	**0.18**
	P4-NA	140.58	232.76	176.87	2.31	7.20	16.51	11.50	0.17
	P4-15	148.79	246.41	184.49	2.06	6.62	14.01	9.99	0.16
	P5-01	333.74	1606.79	780.57	28.87	7.70	18.02	12.83	0.21
	P6-NA^3	**138.78**	**230.10**	**177.18**	**2.35**	**6.73**	**16.11**	**11.34**	**0.18**
	P6-15	139.83	215.79	178.52	1.87	5.78	12.83	9.42	0.13
Near-Global Confs	NullM	1.89e+14	2.53e+14	2.29e+14	9.57e+11	0.66	3.80	1.38	0.05
	SM	1.84e+14	2.55e+14	2.28e+14	4.91e+11	0.40	1.00	0.68	0.01
	UM	85.13	281.57	130.96	0.63	1.93	17.84	7.33	0.29
	G-NA	177.64	2.97e+06	3.59e+05	1.65e+04	1.04	14.02	4.77	0.24
	G15	114.42	207.27	158.81	0.65	4.19	12.25	8.45	0.14
	C-NA	92.37	2.80e+05	2.73e+04	1463.37	1.08	15.86	5.34	0.25
	C15	100.47	215.27	156.58	0.70	3.32	11.15	8.05	0.16
	P1-NA	99.05	478.92	161.76	1.60	2.10	16.40	8.04	0.28
	P1-15	106.52	218.88	155.79	0.67	5.17	11.49	8.34	0.13
	P2-NA	117.17	4.72e+04	1084.74	114.12	2.28	14.85	6.52	0.26
	P2-15	115.09	217.40	157.59	0.66	4.31	11.81	8.25	0.14
	P3-NA^1	**75.07**	**184.25**	**125.53**	**0.47**	**2.22**	**15.16**	**7.62**	**0.28**
	P3-15	93.14	181.98	135.38	0.43	3.90	16.02	9.62	0.23
	P4-NA^3	**83.26**	**230.34**	**127.49**	**0.52**	**2.96**	**17.15**	**8.15**	**0.29**
	P4-15	105.70	186.75	140.18	0.52	2.84	13.73	8.91	0.22
	P5-01	9.08e+12	1.40e+14	1.06e+14	5.51e+11	0.59	6.07	1.94	0.08
	P6-NA^2	**79.90**	**224.49**	**127.67**	**0.55**	**2.86**	**17.02**	**7.33**	**0.26**
	P6-15	108.04	191.77	144.40	0.54	3.37	13.70	8.61	0.21

Table 3: α-cyclodextrin - Numerical results for the mutation operator. For each stage (random, optimized and near-global solutions), energy and RMSD values are shown, the best three ranked operators (energy criterion), being marked in boldface with their rank.

		α-cyclodextrin - Energy				**α-cyclodextrin - RMSD**			
		Min	Max	Avg	StD	Min	Max	Avg	StD
Random Confs	NullM	2.44e+04	5.16e+04	3.85e+04	481.47	3.58	11.32	6.78	0.16
	SM	3532.26	1.65e+04	9256.60	131.55	2.83	7.36	4.19	0.08
	UM	4671.56	2.30e+04	1.08e+04	222.20	2.62	8.80	5.37	0.16
	G-NA	9169.77	3.01e+04	1.70e+04	255.84	3.03	9.03	5.59	0.15
	G15	3847.41	1.68e+04	1.06e+04	166.40	2.69	7.33	4.26	0.09
	C-NA	9028.46	2.46e+04	1.68e+04	258.17	3.13	8.60	5.73	0.14
	C15	4800.02	1.63e+04	9746.97	150.13	3.02	6.81	4.33	0.08
	P1-NA	5776.05	2.04e+04	1.25e+04	212.82	2.81	8.29	5.27	0.15
	P1-15	4142.86	1.77e+04	1.10e+04	170.23	2.77	7.67	4.44	0.09
	P2-NA	5818.68	2.43e+04	1.42e+04	224.97	2.69	8.56	5.54	0.15
	P2-15	5051.90	1.73e+04	1.07e+04	162.48	2.66	6.93	4.28	0.07
	P3-NA	4351.93	1.94e+04	9391.93	173.83	2.79	8.45	5.11	0.14
	P3-15^{1}	**3334.07**	**1.50e+04**	**7567.77**	**128.33**	**2.63**	**7.88**	**4.70**	**0.14**
	P4-NA	3323.74	1.68e+04	9924.91	210.49	2.55	8.48	5.35	0.16
	P4-15^{3}	**3629.05**	**1.40e+04**	**8125.95**	**120.62**	**2.30**	**7.82**	**4.55**	**0.12**
	P5-01	2.32e+04	4.85e+04	3.43e+04	342.89	3.72	14.37	6.83	0.15
	P6-NA	4080.37	1.96e+04	9773.16	166.46	2.61	8.12	5.11	0.15
	P6-15^{2}	**4544.29**	**1.44e+04**	**8966.39**	**129.75**	**2.53**	**8.04**	**4.44**	**0.12**
Optimized Confs	NullM	2.68e+04	5.49e+04	3.59e+04	685.01	3.83	10.50	6.89	0.15
	SM^{1}	**1952.11**	**1.09e+04**	**5981.14**	**160.10**	**2.26**	**6.95**	**4.16**	**0.10**
	UM	3203.01	1.53e+04	8770.11	268.55	2.14	8.55	5.24	0.15
	G-NA	6523.79	2.37e+04	1.46e+04	325.08	3.29	8.70	5.95	0.14
	G15	3967.96	1.49e+04	8839.00	223.87	2.88	7.49	4.75	0.10
	C-NA	4486.20	2.25e+04	1.32e+04	361.20	2.60	8.27	5.71	0.15
	C15	4005.52	1.48e+04	8684.44	216.31	2.44	7.77	4.51	0.10
	P1-NA	3517.32	1.71e+04	9981.99	299.95	2.71	8.37	5.45	0.15
	P1-15	4675.13	1.59e+04	9939.46	203.76	2.68	7.32	4.39	0.08
	P2-NA	3801.12	2.02e+04	1.18e+04	328.44	3.17	8.40	5.74	0.15
	P2-15	3425.52	1.53e+04	9443.50	211.43	2.61	6.88	4.25	0.08
	P3-NA	3208.31	1.39e+04	8105.34	231.76	2.70	8.58	5.57	0.15
	P3-15^{2}	**2648.83**	**1.12e+04**	**6622.94**	**153.01**	**2.49**	**8.25**	**4.99**	**0.14**
	P4-NA	2335.74	1.41e+04	7781.31	224.13	2.39	8.47	5.30	0.17
	P4-15^{3}	**3282.53**	**1.39e+04**	**7470.86**	**165.15**	**2.69**	**8.13**	**4.95**	**0.13**
	P5-01	2.18e+04	4.74e+04	3.21e+04	507.61	3.74	10.64	6.82	0.15
	P6-NA	3627.75	1.37e+04	8581.58	224.06	2.42	8.59	5.31	0.16
	P6-15	3225.20	1.41e+04	7896.56	188.64	2.59	7.78	4.67	0.13
Near-Global Confs	NullM	2.09e+14	2.59e+14	2.37e+14	5.85e+11	0.36	2.26	1.06	0.03
	SM	2.07e+14	5.24e+15	2.38e+14	3.67e+12	0.38	0.66	0.51	0.01
	UM^{3}	**404.53**	**8584.35**	**1788.03**	**16.93**	**0.92**	**4.88**	**1.96**	**0.06**
	G-NA	1.25e+04	2.61e+08	2.37e+07	5.32e+05	0.97	8.74	3.11	0.16
	G15	963.61	7932.57	3244.76	26.82	1.39	4.12	2.40	0.05
	C-NA	778.64	1.08e+07	9.54e+05	2.65e+04	0.89	10.15	3.15	0.18
	C15	1083.76	3.77e+13	2.74e+10	2.74e+10	1.36	4.62	2.43	0.05
	P1-NA	439.03	1.80e+04	3019.91	47.07	0.86	6.77	2.11	0.09
	P1-15	1004.52	8425.18	3815.86	30.88	1.53	5.24	2.48	0.06
	P2-NA	999.39	9.54e+04	1.45e+04	228.14	0.92	8.56	3.24	0.18
	P2-15	697.72	8272.48	3586.77	29.05	1.02	4.19	2.37	0.05
	P3-NA^{1}	**477.05**	**7558.03**	**1562.08**	**12.67**	**0.86**	**5.19**	**1.83**	**0.05**
	P3-15	711.49	6676.50	2330.09	20.46	1.07	5.76	2.16	0.05
	P4-NA	404.43	6649.77	1611.17	13.19	1.02	6.13	1.96	0.07
	P4-15	837.97	6729.36	2683.48	23.55	1.12	3.95	2.25	0.04
	P5-01	1.03e+13	1.54e+14	1.26e+14	2.96e+11	0.52	3.03	1.26	0.05
	P6-NA^{2}	**427.66**	**7311.48**	**1661.02**	**13.92**	**0.93**	**6.65**	**2.02**	**0.08**
	P6-15	799.63	7908.68	3014.23	25.29	1.20	7.14	2.35	0.06

Studying the obtained numerical results, it follows that, while still standing as a diversification factor, none of the defined mutation operators were able to introduce significant perturbations to enable the embedding EA to attain improvement. As in the presence of strong local minima, the selection and the replacement schemes gain a more significant role, combined mechanisms have to be employed in order to advance the search. Thus, more complex exploration strategies may be required as to counterbalance redundancy in the search and to avoid continuously focusing on a limited region of the exploration space. Whilst part of the issues are addressed by the crossover operators, a stronger impact may be introduced by the local search mechanism.

Analyzing the complete set of results it follows that the Pearson types III, IV and VI based operators attain the best (energy) rank in most of the studied cases. Notwithstanding, none of these operators ranked in the first three mutations for the RMSD section. Instead the Pearson types I, II and VII (exclusively a histogram illustration is enclosed for Pearson type VII due to the poor results on the energy criterion) based operators were classed among the firsts in all the RMSD analysis tests. Consequently, no coherent inference can be made for accepting or rejecting a specific operator, as designated in the energy analysis section. Nevertheless, relying on the heuristic nature of the hybrid approach to be constructed, a straightforward design would consist of using a combined mutation operator.

For example, the P$\{3, 4, 6\}$-$\{$NA, 15$\}$ set of operators may be employed, with a higher incidence of the P3 operator and with a higher probability of using the inverse annealing scheme. As the Pearson operators not relying on annealing schemes had a tendency of better scoring in the final phases, it would seem accurate to dynamically modify the annealing scheme. Hence, for the early stages of the search a factor of 15.0 can be used as a target value for the annealing scheme, to be gradually reduced to 1.0 (no annealing scheme) as the algorithm advances.

In addition to relying on dynamic resolution schemes, adaptive approaches can be addressed. Note that all the three considered operators, P3, P4 and P6, result by modifying the mean, variance, skewness and kurtosis factors of a unique system. Thus, a polymorphic mutation would seem of interest, capable of standing as a substitute for the defined operators – the Gaussian and the Cauchy distributions can be seen as particular cases, depending on the specified parameters. Moreover, highly flexible exploration strategies can be designed, allowing, for example, for a smooth transition from a Pearson type III based mutation, in the early stages of the search, to a Pearson type VI derived operator for the final part of the exploration.

6 Conclusions and Future Directions

Analysis results come to sustain dynamic approaches – most of the considered operators rely on a generation index dependent variance. Furthermore, significant differences exist when comparing the results obtained for the considered annealing schemes.

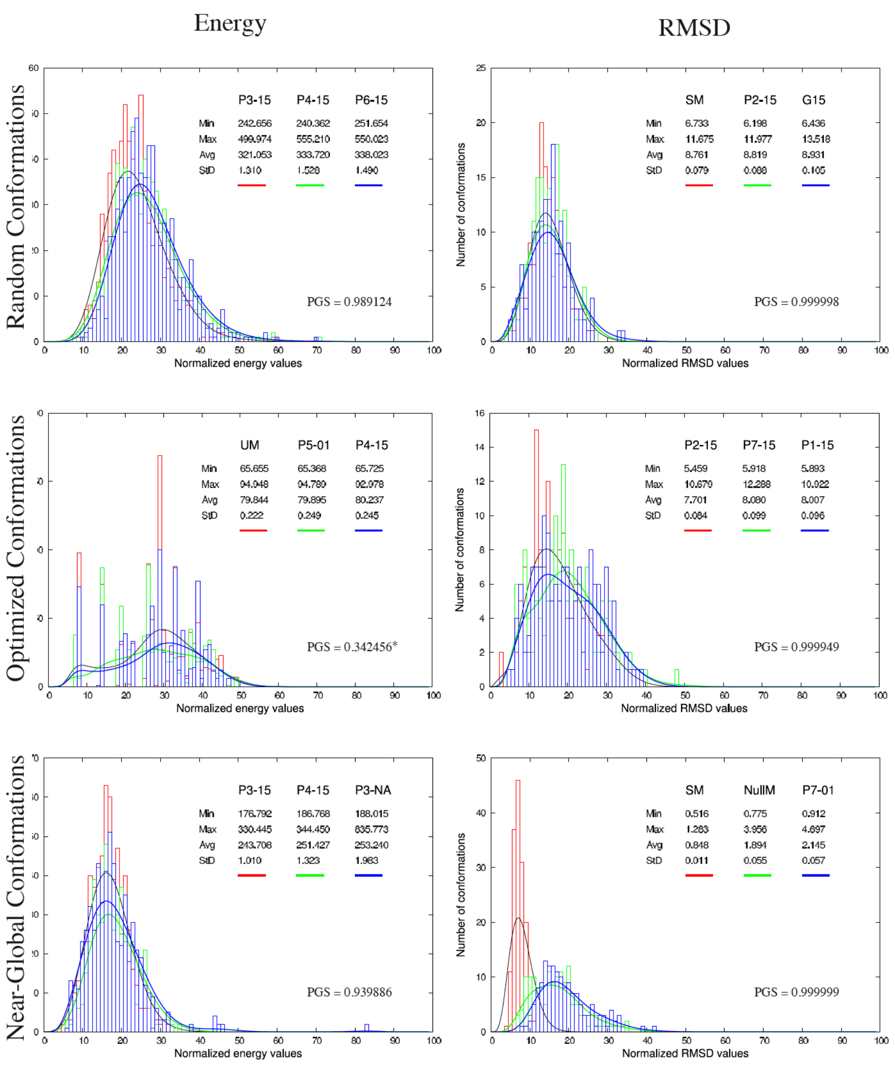


Fig. 4: 1L2Y - Analysis and selection of mutation operators. Energy and RMSD histograms for the results obtained when launching the embedding evolutionary algorithms with random, optimized and near-global solutions, respectively. Only the best three ranked operators are depicted with the corresponding PGS value.

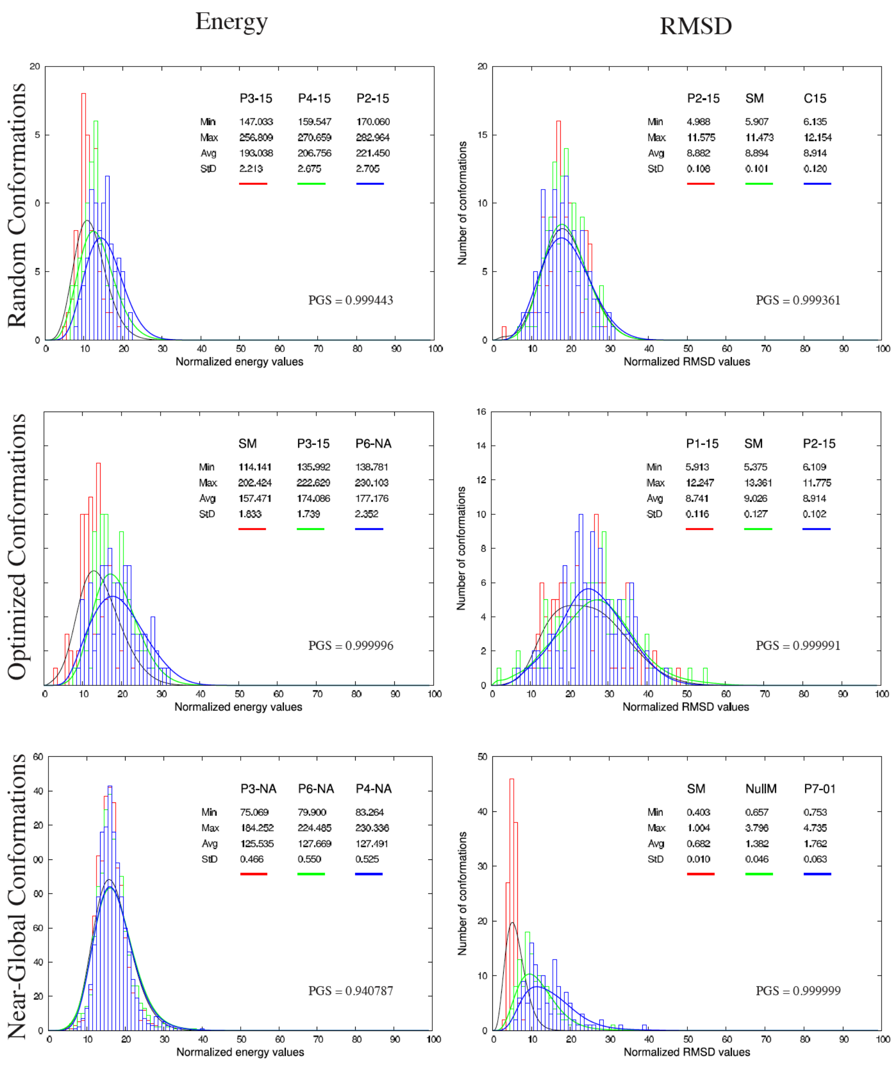
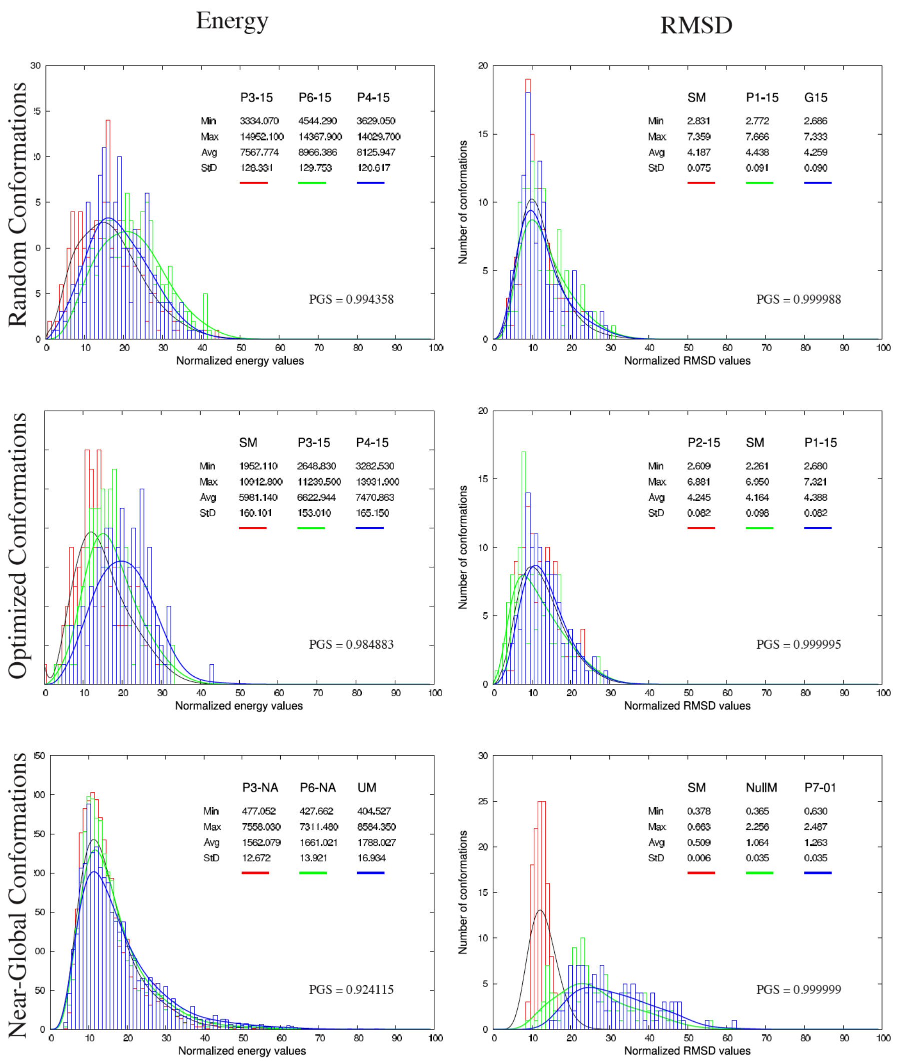


Fig. 5: 1LE1 - Analysis and selection of mutation operators. Energy and RMSD histograms for the results obtained when launching the embedding evolutionary algorithms with random, optimized and near-global solutions, respectively. Only the best three ranked operators are depicted with the corresponding PGS value.



	P3-15	P6-15	P4-15
Min	3334.070	4544.290	3629.050
Max	14952.100	14367.900	14029.700
Avg	7567.774	8966.386	8125.947
StD	128.331	129.753	120.617
	🟥	🟩	🟦

	SM	P1-15	G15
Min	2.831	2.772	2.686
Max	7.359	7.666	7.333
Avg	4.187	4.438	4.259
StD	0.075	0.091	0.090
	🟥	🟩	🟦

	SM	P3-15	P4-15
Min	1952.110	2648.830	3282.530
Max	10012.800	11239.500	13931.900
Avg	5981.140	6622.944	7470.863
StD	160.101	153.010	165.150
	🟥	🟩	🟦

	P2-15	SM	P1-15
Min	2.609	2.261	2.680
Max	6.881	6.950	7.321
Avg	4.245	4.164	4.388
StD	0.082	0.098	0.082
	🟥	🟩	🟦

	P3-NA	P6-NA	UM
Min	477.052	427.662	404.527
Max	7558.030	7311.480	8584.350
Avg	1562.079	1661.021	1788.027
StD	12.672	13.921	16.934
	🟥	🟩	🟦

	SM	NullM	P7-01
Min	0.378	0.365	0.630
Max	0.663	2.256	2.487
Avg	0.509	1.064	1.263
StD	0.006	0.035	0.035
	🟥	🟩	🟦

Fig. 6: α-Cyclodextrin - Analysis and selection of mutation operators. Energy and RMSD histograms for the results obtained when launching the embedding evolutionary algorithms with random, optimized and near-global solutions, respectively. Only the best three ranked operators are depicted with the corresponding PGS value.

As most of the analyzed operators can be simulated by employing the Pearson system based mutation, it would seem sufficient to act on the four parameters determining the Pearson distribution type. Consequently, an *a priori* operator selection phase can be considered, preceding the execution of the evolutionary algorithm. Addressing the different parameters of the operators, this results in a parameter tuning process – in this case an ensemble of parameters is to be selected from a specified set. Nevertheless, to a certain extent, the analysis results may stand only for a reduced set of conformations, hence, requiring the introduction of adaptive schemes.

A straightforward automatic tuning and operator selection approach consists in using the parallel selection procedure as a first phase of the conformational sampling algorithm. As previously mentioned, similar studies, not included here, were conducted on intensification operators and local search algorithms. Thus, by considering distinct configuration phases for each class of components embedded by the evolutionary exploration approach, multiple designs can be constructed by combining the obtained results. This is equivalent to reusing the analysis architecture developed here, in conjunction with an evolutionary approach.

As differences exist between the operators that scored in the first three at different stages, as an incipient design, the parameters of the Pearson system based operator can be integrated in the coding information of the individuals. Consequently, the evolution process can act not only on the resulting individuals but also on the parameters to be adapted. Having a large percentage of the individuals as output of a particular parameter setup can hence introduce a bias, discarding individuals generated as a result of lower quality setups. The main inconvenience of this particular approach resides in the fact that an individual's encoding has to be modified, along with the way operators act on the constructed encoding. Thus, a less intrusive approach, relying on the parallel environment setup consists of having a distinct algorithm exploring the parameter space, concurrently with the main exploration algorithm. While imposing higher design and implementation difficulties, a separate parameter – the solution exploration paradigm introduces important advantages, allowing for different experimental designs. A decoupling construction, including a parameter optimization process, executed in parallel with the evolutionary algorithm, is the subject of a distinct study, following the results.

References

[1] Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. IEEE Trans. Evolutionary Computation 6(5), 443–462 (2002)

[2] Alba, E., GLuque, E.G.T., Melab, N.: Metaheuristics and parallelism. In: Alba, E. (ed.) Parallel Metaheuristics. Wiley Series on Parallel and Distributed Computing. Wiley, Chichester (2005)

[3] Anfinsen, C.B., Haber, E., Sela, M., White, F.H.: The Kinetics of Formation of Native Ribonuclease During Oxidation of the Reduced Polypeptide Chain. Proceedings of the National Academy of Sciences of the United States of America 47(9), 1309–1314 (1961)

[4] Blundell, T.L., Sibanda, B.L., Sternberg, M.J., Thornton, J.M.: Knowledge-based prediction of protein structures and the design of novel molecules. Nature 326(6111), 347–352 (1987)

[5] Branke, J., Chick, S.E., Schmidt, C.: New developments in ranking and selection: an empirical comparison of the three main approaches. In: WSC 2005: Proceedings of the 37th Conference on Winter Simulation, pp. 708–717. ACM, New York (2005)
[6] Branke, J., Chick, S.E., Schmidt, C.: Selecting a selection procedure. Management Science 53(12), 1916–1932 (2007)
[7] Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.Y.: A limited memory algorithm for bound constrained optimization. SIAM Journal on Scientific Computing 16(6), 1190–1208 (1995)
[8] Cahon, S., Melab, N., Talbi, E.G.: Paradiseo: A framework for the reusable design of parallel and distributed metaheuristics. Journal of Heuristics 10(3), 357–380 (2004)
[9] Cahon, S., Melab, N., Talbi, E.G.: An enabling framework for parallel optimization on the computational grid. In: CCGRID, pp. 702–709 (2005)
[10] Cantu-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Norwell (2000)
[11] Cappello, F., Caron, E., Dayde, M., Desprez, F., Jegou, Y., Primet, P., Jeannot, E., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Quetier, B., Richard, O.: Grid'5000: a large scale and highly reconfigurable grid experimental testbed. In: The 6th IEEE/ACM International Workshop on Grid Computing, pp. 99–106 (2005)
[12] Chen, C., Lin, J., Yücesan, E., Chick, S.E.: Simulation budget allocation for further enhancing the efficiency of ordinal optimization. Journal of Discrete Event Dynamic Systems: Theory and Applications 10, 251–270 (2000)
[13] Chen, C.H.: A lower bound for the correct subset-selection probability and its application to discrete event system simulations. IEEE Transactions on Automatic Control 41(8), 1227–1231 (1996)
[14] Chick, S.E., Inoue, K.: New results on procedures that select the best system using crn. In: Simulation Conference Proceedings, vol. 1, pp. 554–561 (2000)
[15] Cozzone, A.J.: Proteins: Fundamental chemical properties. Encyclopedia of Life Sciences, pp. 1–10. Macmillan Publishers Ltd, Nature Publishing Group (2002), www.els.net
[16] Dauber-Osguthorpe, P., Roberts, V.A., Osguthorpe, D.J., Wolff, J., Genest, M., Hagler, A.T.: Structure and energetics of ligand binding to proteins: Escherichia coli dihydrofolate reductase-trimethoprim, a drug-receptor system. Proteins: Structure, Function, and Genetics 4(1), 31–47 (1988)
[17] Dill, K.A.: Theory for the folding and stability of globular proteins. Biochemistry 24(6), 1501–1509 (1985)
[18] Gropp, W.: Mpich2: A new start for mpi implementations. In: Proceedings of the 9th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, p. 7. Springer, London (2002)
[19] Gropp, W., Lederman, S.H., Lumsdaine, A., Lusk, E., Nitzberg, B., Saphir, W., Snir, M.: MPI: The Complete Reference, The MPI-2 Extensions, vol. 2. MIT Press, Cambridge (1998)
[20] Heinrich, J.: A guide to the Pearson Type IV distribution (2004)
[21] Herrera, F., Lozano, M., Verdegay, J.L.: Fuzzy connective based crossover operators to model genetic algorithms population diversity. Tech. Rep. DECSAI-95110, University of Granada (1995)
[22] Herrera, F., Lozano, M., Verdegay, J.: Dynamic and heuristic fuzzy connectives-based crossover operators for controlling the diversity and convengence of real-coded genetic algorithms (1996)
[23] Herrera, F., Lozano, M., Verdegay, J.L.: Fuzzy connectives based crossover operators to model genetic algorithms population diversity. Fuzzy Sets Syst. 92(1), 21–30 (1997)

[24] Herrera, F., Lozano, M., Sánchez, A.M.: A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. International Journal of Intelligent Systems 18(3), 309–338 (2003)

[25] Hestenes, M.R.: Iterative methods for solving linear equations. Report 52-9, NAML (1951); Reprinted in the Journal of Optimization Theory and Applications 11, 323–334 (1973)

[26] Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving linear systems. Journal of Research of the National Bureau of Standards 49(6), 409–436 (1952)

[27] Ingber, L.: Adaptive simulated annealing (ASA): Lessons learned. Control and Cybernetics 25, 33–54 (1996)

[28] Ingber, L.: Adaptive simulated annealing (asa) and path-integral (pathint) algorithms: Generic tools for complex systems. Tech. rep., Lester Ingber Research, Chicago, IL (2001)

[29] Joy, S., Nair, P.S., Hariharan, R., Pillai, M.R.: Detailed comparison of the protein-ligand docking efficiencies of gold, a commercial package and arguslab, a licensable freeware. Silico Biology 6(6), 601–605 (2006)

[30] Linlin Qiu, S.J.H.: Internal friction in the ultrafast folding of the tryptophan cage. Chemical Physics 1(312), 327–333 (2005)

[31] Michalewicz, Z.: Genetic algorithms + data structures = evolution programs, 2nd edn. Springer, New York (1994)

[32] Morris, G.M., Goodsell, D.S., Halliday, R.S., Huey, R., Hart, W.E., Belew, R.K., Olson, A.J.: Automated docking using a lamarckian genetic algorithm and an empirical binding free energy function. Journal of Computational Chemistry 19(14), 1639–1662 (1999)

[33] Nagahara, Y.: The PDF and CF of Pearson type IV distributions and the ML estimation of the parameters. Statistics & Probability Letters 43(3), 251–264 (1999)

[34] Nagahara, Y.: A method of simulating multivariate nonnormal distributions by the pearson distribution system and estimation. Computational Statistics & Data Analysis 47(1), 1–29 (2004)

[35] Neumaier, A.: Molecular modeling of proteins and mathematical prediction of protein structure. SIAM Review 39(3), 407–460 (1997)

[36] Parent, B., Tantar, A., Melab, N., Talbi, E.G., Horvath, D.: Grid-based evolutionary strategies applied to the conformational sampling problem. In: IEEE Congress on Evolutionary Computation pp. 291–296 (2007)

[37] Ponder, J.W., Case, D.A.: Force fields for protein simulations. Advances in Protein Chemistry 66, 27–85 (2003)

[38] Rabow, A.A., Scheraga, H.A.: Improved genetic algorithm for the protein folding problem by use of a Cartesian combination operator. Protein Sci. 5(9), 1800–1815 (1996)

[39] Rosin, C.D., Halliday, R.S., Hart, W.E., Belew, R.K.: A comparison of global and local search methods in drug docking. In: Bäck, T. (ed.) Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA 1997). Morgan Kaufmann, San Francisco (1997)

[40] Schmidt, C., Branke, J., Chick, S.E.: Integrating techniques from statistical ranking into evolutionary algorithms. In: Rothlauf, F., Branke, J., Cagnoni, S., Costa, E., Cotta, C., Drechsler, R., Lutton, E., Machado, P., Moore, J.H., Romero, J., Smith, G.D., Squillero, G., Takagi, H. (eds.) EvoWorkshops 2006. LNCS, vol. 3907, pp. 752–763. Springer, Heidelberg (2006)

[41] Snow, C.D., Qiu, L., Du, D., Gai, F., Hagen, S.J., Pande, V.S.: Trp zipper folding kinetics by molecular dynamics and temperature-jump spectroscopy. Proc. Natl. Acad. Sci. U S A 101(12), 4077–4082 (2004)

[42] Stewart, C.A., Müller, M.S., Lingwall, M.: Progress towards petascale applications in biology: Status in 2006. In: Euro-Par Workshops, pp. 289–303 (2006)
[43] Talbi, E.G.: A taxonomy of hybrid metaheuristics. Journal of Heuristics 8(5), 541–564 (2002)
[44] Tantar, A.A., Melab, N., Talbi, E.G.: A grid-based genetic algorithm combined with an adaptive simulated annealing for protein structure prediction. Soft Computing 12(12), 1185–1198 (2008)
[45] Tantar, A.A., Melab, N., Talbi, E.G., Parent, B., Horvath, D.: A parallel hybrid genetic algorithm for protein structure prediction on the computational grid. Future Generation Computer Systems (in press)
[46] Tavares, J., Tantar, A.A., Melab, N., Talbi, E.G.: The influence of mutation on protein-ligand docking optimization: a locality analysis. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 589–598. Springer, Heidelberg (2008)
[47] Thomsen, R.: Flexible ligand docking using evolutionary algorithms: investigating the effects of variation operators and local search hybrids. Biosystems 72(1-2), 57–73 (2003)
[48] Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation 3(2), 82–102 (1999)
[49] Zhu, C., Byrd, R.H., Lu, P., Nocedal, J.: Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. ACM Trans. Math. Softw. 23(4), 550–560 (1997)

Evolving Computer Chinese Chess Using Guided Learning

H.Y. Quek, H.H. Chan, K.C. Tan, and A. Tay

Abstract This chapter explores the feasibility of using genetic algorithms to improve the evaluation of Chinese chess programs. A game engine that uses the negascout search algorithm in combination with internal iterative deepening search is developed. As a means to enhance the search process, techniques such as null-move-pruning, futility pruning, razoring and selective search extensions are used. Unnecessary expensive re-searches for the negascout are avoided through move ordering techniques, which are governed by the Most Valuable Victim (MVV) / Least Valuable Attacker (LVA), killer and history heuristics. To evaluate the game positions at any point of time, a static evaluation function (using hand-tuned weights) is utilized in conjunction with quiescent search, whose weights are tuned by a genetic algorithm using a population of chromosomes. Moves taken from grandmasters' games are used as training data to evaluate the fitness of chromosomes during evolution. This is determined based on the number of 'correct' moves made by the program. The evolved programs are benchmarked against the un-evolved version and random online human players. Results show that evolution with guided learning does improve the playing strength of the Chinese chess program significantly.

1 Introduction

Chess has always been an important area of research in artificial intelligence (AI). Its perfect information nature provides a relatively simple platform to test out new theories in AI. In particular, pitting chess-playing AIs against human grandmasters allows researchers to better understand the strengths and shortcomings of AI [1], unlike many real-life problems where no reliable human experts are available. Over past decades, AI has made tremendous progress in several chess variants. In checkers, Schaffer's Chinook defeated the human world champion, Don Lafferty in 1994 [2]. In international chess, IBM's Deep Blue also defeated the human world champion, Gary Kasparov in 1997 [3]. Since then, the performance of AI in

Department of Electrical and Computer Engineering,
National University of Singapore,
4 Engineering Drive 3, 117576, Singapore

international chess has improved further. Today, the strongest program, Hydra, which is estimated to have an Elo playing strength of 3000, has not lost a game to an unaided human opponent.

In contrast, the performance of AI in Chinese chess has lagged behind substantially. The best performance attained thus far was Neuchess's narrow victory over five masters and grandmasters in 2006 [4]. However, the human grandmasters were essentially playing at a disadvantage [5] since the games were not played using tournament time control. In the strict sense, AI is yet to defeat a world champion under tournament time control. Grandmaster Liu Da Hua commented on the strategic weakness of Neuchess – although it possesses superior speed, its judgment is weak especially near the endgame [5]; suggesting that there is still ample room for improvement in the quality of AI in Chinese chess.

The performance of a Chinese chess program is largely determined by two factors, speed of its search and accuracy of its evaluation. The speed of search is determined by the number of nodes that can be traversed in a fixed amount of time. As verified by an experiment conducted years ago, Thompson [6] showed that, with all things being equal, a program that can perform a deeper search will always beat one that only searches at a shallow level. Since a faster speed denotes the ability to search at a larger depth, this will likewise also translate to better performance. The search speed of chess programs has improved tremendously in the last few decades, mainly due to improvement in computer hardware. When Deep Blue defeated Kasparov in 1997, it was using specialized hardware such as VLSI chess chips, and was able to search 200 million positions per second [7]. In contrast, the strongest chess machine in the early 1980s, Belle, was only able to search 180,000 positions per second [8]. Despite this, raw brute-force search speed alone is not enough. Though a human grandmaster can at best only evaluate a handful of positions per second, his extensive chess knowledge allows him to conduct selective search on only a few moves at each search ply. This implies a search tree with a notably smaller branching factor than a chess machine's, which makes it possible for a human grandmaster to reach approximately the same search depth as a chess machine.

As the speed of the Chinese chess machine is already faster than that of a human grandmaster, the performance bottleneck is actually the accuracy of its evaluation. The need for an evaluation function was first proposed by Shannon [9] in 1949. In chess, this is usually made up of a mathematical function. Score weights are assigned to reward and penalize the AI, based on the material balance, King's safety, mobility and relative positions of the pieces. The conventional approach to improve the evaluation function is via "hand-tuning" [10] – where a programmer is tasked to adjust the weights manually, according to the performance of the program. This is a slow and tedious process. As expert knowledge is required, accuracy of evaluation is ultimately limited by the skill of the "tuner". Besides, it is difficult, even for a grandmaster, to quantify weights accurately. Genetic algorithms (GAs) offer an elegant solution to the problem as they do not require expert human chess knowledge to perform the modification of the weights. Moreover, it also possesses the potential to discover new Chinese chess strategies, which might allow the AI to evaluate even better than any human grandmaster.

For example, Blondie24, a checkers program which uses evolutionary algorithms, was able to make moves that were complimented by opponents as being “strange” and “very tough” [11].

To evolve an effective chess solution using GAs, one important aspect is the fitness evaluation of chromosomes e.g., descriptive of different instances of the chess program’s encoding operational parameters. As the ultimate objective is to win games, a popular and most direct method is to base the fitness on the number of games won against its peers or a “control” player as in the case of Blondie24 [11]. However, there are several problems associated with this method. Firstly, when playing strengths of chromosomes become very close, most of the games will probably end in a draw. In such situations, resolution of fitness evaluation might be too low and it becomes difficult to determine whether one chromosome is stronger than the other at a reasonable confidence level. To improve the resolution of the fitness evaluation, the number of games played during the selection process might have to be increased substantially and this will result in a long simulation time. Suppose 100 games are played on average between two chromosomes to decide the stronger of the two. If each is given one second to move and a game terminates in 100 moves[1] on average, an evolution process which uses tournament selection and a small population of 10, will take an unrealistic 231 days for a mere 200 generations.

Secondly, there is also a likely problem of intra-sensitivity. Suppose there are three chromosomes A, B and C. If A beats B; B beats C and C beats A, it will be difficult, if not impossible, to determine the fittest chromosomes among the three. This is because playing games amongst chromosomes only measures the differences in their subjective playing strengths. Albeit chromosome A has stronger overall playing strength than B, chromosome B can still win most of its games against A if it is able to exploit a particular weakness in A’s evaluation function consistently. Even if a “control” chromosome which all others are pitted against is present, the number of wins against the “control” chromosome still very much depends on how well a chromosome is able to exploit the weakness of the “control” chromosome. In general, it is difficult to assess the objective playing strength of a chromosome if it does not play against a large number of random opponents.

Thirdly, it is extremely difficult to teach a program to play chess by using only the end results of the game [12]. The chromosome might make 100 moves, of which just one bad move is enough to cause the program to lose a game. In such cases, informing the chromosome simply that it has lost the game does not aid in finding the erroneous move. This succinctly explains the inherent difficulty of using games between chromosomes as the basis for fitness evaluation.

Aimed at addressing the above issues, the proposal in this chapter uses moves of past grandmasters’ games as the basis of fitness evaluation. The fitness score of a chromosome is determined and will change in accordance to the number of “correct” moves it has made. “Correct” is defined as making the move that the grandmaster made in that identical game situation. Usage of training data allows for objective

[1] The number of steps will increase as playing strengths of the chromosomes get closer.

measurement of fitness, and could potentially avoid the afore-mentioned problems. Compared to playing games between chromosomes, this "guided learning" approach to training takes a relatively indirect route. However, by evolving the chess evaluation function towards that of the grandmaster, the accuracy of the program's evaluation and hence its playing strength could potentially be improved over time.

This chapter is organized as follows. Section 2 presents a brief introduction of the rules in Chinese chess and how its elements tend to differ from those in conventional chess. Section 3 describes the proposed Chinese chess engine while Section 4 illustrates the application of the GA. Section 5 discusses the requirements of the training data and the process of selecting it. Section 6 gives the results of simulations using the chosen dataset and Section 7 assesses the performance of the evolved program against the un-evolved one as well as random online human players, with a brief analysis of results. Section 8 concludes with a broad summary of the discussion on the overall findings and Section 9 discusses some possible works that can be embarked upon in future.

2 Chinese Chess Rules

The rules of Chinese chess are similar to that of chess where each player controls a set of pieces and tries to capture the opponent's King. Taking turns to move the pieces one at a time, the objective of the game is to checkmate the opponent i.e., by attacking his King (placed it in check) in a way that no further move is able to eliminate that check. Unlike chess, however, the Chinese chess board is a grid of

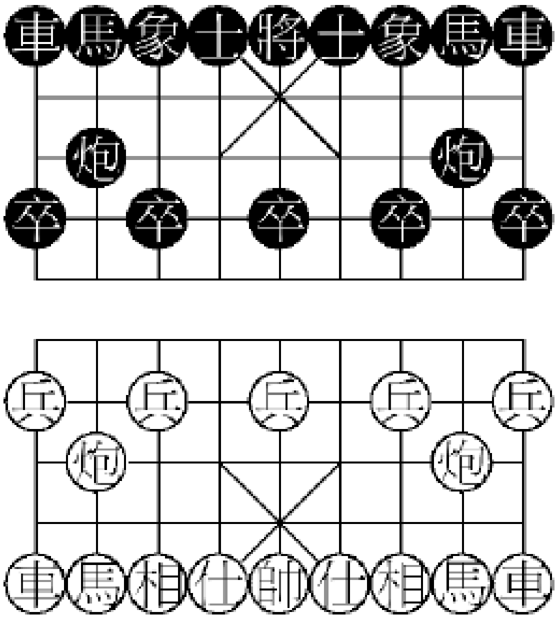


Fig 1. Initial position in a game of Chinese chess.

Table 1. Movement rules and mobility constraints of each Chinese chess piece.

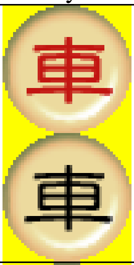
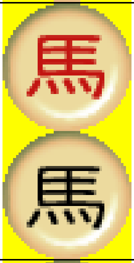
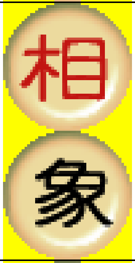
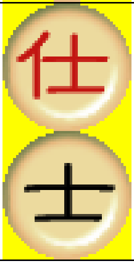
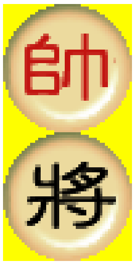
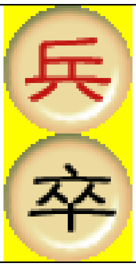
Chess pieces	Piece symbols	Movement/Mobility constraints
Rook		The Rook moves exactly the same way as in chess – in a straight horizontal or vertical line across any number of empty spaces. It can stop on empty spaces or on the first space it comes to that is occupied by an opponent piece. Passing over occupied spaces is prohibited.
Horse		The Horse is able to reach the same spaces that chess Knights can reach, with the exception of leaping over pieces. Its movement is characterized by an orthogonal move of one space followed by one that is diagonally outward. However, movement in a direction is blocked if the first space it would move over is occupied.
Elephant		The Elephant moves two spaces in the same diagonal direction. Like the Horse, it is not allowed to leap over occupied spaces. Its movement is blocked if the first step of its move is over an occupied space. Unlike the Horse, they are confined to their own side of the river (interruption/gap in the middle of the board).
Advisor		The Advisor moves one space along the diagonal lines in the palace (marked by x-shaped cross connecting its four corner points) that connects points it may reach. The piece is not allowed to move out of the palace.
King		The King moves one space orthogonally within the palace's confines. Like the Advisor, it cannot leave the palace. Two Kings are also not allowed to face each other on an open file e.g., a red King on c1 and a black King on c10, with no piece on the c-file between them. If either King sits exposed on an open file, the other is prohibited from moving to occupy that file.

Table 1. (*continued*)

Cannon		The Cannon moves differently when it captures than if it moves passively. It moves similarly as a Rook when not capturing a piece but moves in the same direction when capturing except that it must hop over a single intervening piece e.g., Cannons capture by hopping over a second piece to capture a third piece. They can never hop over more than one piece in a given move.
Pawn		The Pawn moves one space vertically forward but gets the added ability to move one space sideways after it crosses the river. Unlike the Pawn in chess, it moves and captures in the same manner, never gets a double move, and does not promote on the last rank.

ten horizontal lines and nine vertical lines where pieces are placed on intersections called points instead of squares. The setup of the various pieces in their respective initial positions at the beginning of a game is shown in Figure 1.

Similar to chess, each board piece has its own unique style of movement as well as mobility constraints. These are described and presented in Table 1. Other miscellaneous game rules include the conventions that Red moves first and perpetual check is prohibited e.g., a player is not allowed to check his opponent more than three times in a row using the same piece and board positions.

3 Chinese Chess Engine

Before a GA can actually be applied, a Chinese chess engine must first be created. For a chess program to compete effectively against a human grandmaster it has to be able to search 10 plies deep for each move [5]. Given that the average branching factor of Chinese chess is 38 [13], it becomes unviable to implement search on the classical mini-max algorithm, since the Chinese chess program will need to evaluate approximately 6×10^{15} nodes per move by traversing through every node in the game tree. Though Alpha-beta pruning can achieve an effective branching factor of $\sqrt{b}$, where b is the branching factor for plain mini-max [14], its performance is largely dependent on move-ordering e.g., pruning is efficient provided that better moves are tried before bad ones. Even if this inconsistency is ignored, the algorithm will still have to traverse 3×10^{8} nodes, which is still not quite fast enough to play a game of Chinese chess under tournament time control. A search strategy that is more efficient is required.

3.1 Principal Variation Search

Negascout, also called the Principal Variation Search, is used as the basis of search in the proposed Chinese chess engine. It is employed in combination with iterative deepening, internal iterative deepening and quiescent search to ensure the presence of a correct principal variation move, whenever possible. Proposed by Reinefeld in 1989, negascout dominates alpha-beta and never examines a node that can be pruned [15]. It assumes that the first move found is usually the best one. For subsequent moves, a minimal window search is carried out to prove that other moves' scores cannot improve upon the score on the first move. The main advantage is that such searches fail quickly and hence greatly increase the search speed. However, if a better score is found, a full-window search is repeated again for that move. If there are too many re-searches, performance of negascout will, however, deteriorate. The crucial performance factor of the algorithm is to ensure that the first move considered is the best move, at least for most of the time. However, if the best move is known, then there is no need to do a search in the first place. There is thus a need for the AI to make a calculated guess of what the best move is likely to be. This is done via the storage of principal variation moves, where each refers to the best move found for a particular position in earlier, shallower searches. Principal variation moves are stored in a hash table and are always searched before other moves. In the event that the search tree is traversed in an ordinary depth-first fashion, it is highly unlikely that a principal variation move can be found within the hash table. For negascout to work effectively, an iterative deepening search must be used in combination with it.

3.2 Iterative Deepening

In the ordinary depth first search, each branch is explored up to the target search depth (barring any pruning). With iterative deepening, the target search depth is gradually increased until the maximum search depth is reached or when the time limit is reached. It seems that iterative deepening wastes plenty of valuable time by searching repeated nodes over and over again, but due to the fact that previous search results are stored in the hash table, the increase in the time spent on researching is minimal. Also, because negascout is used, the time-savings offered by the principal variation of previous searches more than compensate for the additional overheads. In this work, iterative deepening is also taken to a much higher level and used within the negascout algorithm itself. Such internal iterative deepening is used when the remaining search depth is more than three and no best move is found in the hash table as yet. In such situations, a shallower search is first executed for *depth* = *remaining depth* – 2. The best move found via this internal iterative deepening is then used as the first move for the full depth negascout search.

3.3 Hash Table

Two hash tables are used in this program to store the principal variation moves and the evaluation scores of the previously searched positions. The key and lock of each game position are created by Zorbist Hashing using two arrays, one for the key and one for the lock. By using both the key and lock, the state of the game board can be represented with a 64-bit number. This removes the need to traverse through the entire board to check if the positions of all the pieces are identical for the game position and the hash table entry that is used to probe the hash tables. A simple check to ensure that both keys and locks are identical will suffice.

In addition, the program also reduces search redundancy via the use of scores and principal variation moves that were stored in the hash tables. For instance, if the program is currently searching a game position with remaining search depth of two, the result in the hash table is returned directly if the table already contains a result that corresponds to a search depth of more than or equal to two. In this case, the need to traverse through the remaining depth is thus avoided. Such reduction in search effort is extremely useful for Chinese chess programs since different orders of moves frequently transpose onto each other. In the event that the result stored in the hash table is shallower than the current remaining depth, the principal variation move from the shallower search can still be used as an "educated guess" of what the best move is. This is usually made the first move to be searched by negascout even when the score in the hash table is not directly used. As the size of the hash table is finite, there is a need to determine when to replace the original entry. Two main replacement strategies are employed in the Chinese chess engine.

(a) Always replace
Under this strategy, the original hash table entry is always overwritten, regardless of the useful search results which it might contain. For example, if the original entry is from a search of depth 10 and the new entry is from a search of depth three, the older entry, which is likely to be more important, is overwritten.

(b) Overwrite shallower entry
Evaluation scores, at higher depths, are more accurate than those which are derived from shallower searches. Hence it makes sense to replace the original hash table entry, if the new evaluation score is from a deeper search. However, as the AI makes more and more moves, the hash table is bound to be filled up with older and irrelevant entries. If these entries are results from deeper searches, they will never be replaced, and this reduces the number of useful entries in the hash table. A possible solution is to use a stale flag. Every time the AI makes a move, the stale flags of all the hash table entries are set. Each time a hash table entry is successfully probed for the best move or evaluation score, its stale flag is reset. When deciding whether or not to overwrite an entry, its corresponding stale flag is checked first. If the stale flag is set, it is always overwritten. Otherwise, the entry is only overwritten when the new search result is from a deeper search.

In this work, both strategies are used in unison. This is taken from the concept used by Tu's Chinese chess program [16]. Instead of using only one hash table, two hash tables are used. The first uses the "overwrite shallower entry strategy", while the second uses the "always replace" strategy. When an entry of the first hash table is overwritten, and its stale flag is not set, the entry is copied into the second hash table. This way, it is possible to have the best of both worlds.

3.4 Move Ordering

As mentioned earlier, the performance of negascout is dependent on the quality of its move-ordering, which refers to the order in which the moves are generated and searched. To avoid unnecessary, expensive re-searches, the Most Valuable Victim (MVV) / Least Valuable Attacker (LVA), killer and history heuristics are used to order moves. The order in which the moves are searched is as follows:

(1) Principal variation move
This is the best move that is stored in the hash table as obtained from the previous, shallower searches. If no best move is found, an internal iterative deepening search is carried out.

(2) Killer Moves
After trying the principal variation move, four killer moves are tried. These are moves which produced a beta-cutoff at the same depth in another branch of the search tree and are used only if the move is legal in the current position. The idea is that there is usually a move that we want to prevent the opponent from making and/or a move that we'll always want to make, whenever possible. Using such moves removes the need to go through the time-consuming process of move generation.

(3) Capture moves, governed by the MVV/ LVA heuristic
Capture moves are moves which capture an opponent piece. They are sorted in a most-valuable-victim and least-valuable-attacker order. This means that the moves that capture the opponent King are tried first, and those that capture the opponent Pawns are tried last. Also, among the moves that capture the same piece, for example a Rook, the Pawn captures Rook moves are tried before the Rook captures Rook move. This is used because capturing a major piece (such as a Rook) is usually better than capturing a minor piece (such as a Pawn) in chess. Due to the possibility of the opponent recapturing the very next turn, it is also better to capture with a less valuable attacker. All capture moves in this work are generated in a piecemeal fashion. The King-captures are generated and searched before generating any Rook-captures. This search procedure improves the speed of the program as there is no need to generate the other capture moves if a King-capture has created a beta-cutoff.

(4) Remaining moves, sorted according to history heuristic
Finally, the non-capture moves are generated. The ordering of the non-capture moves is governed by a history heuristic which is represented by a 14 by 256 by 256 array. The first index indicates the color and type of the piece, the second index the source square on the game board, and the third index the destination

square. Each time a non-capture move creates a beta-cutoff in the search tree, the corresponding element in the array is increased by the square of the remaining search depth. All the non-capture moves are sorted in a descending order, based on their respective scores in the history array. The moves with higher scores are tried first as a move which creates an earlier beta-cutoff is likely to be a better move. On the whole, the history heuristic allows the AI to learn the moves which are likely to be better. It is similar in concept to the killer heuristic, just that the killer heuristic is more of a "short-term" memory, whereas the history heuristic is a "long-term" memory.

3.5 Quiescent Search

The strength of a Chinese chess program depends heavily on the accuracy of the evaluation function. As evaluation depends on the number of pieces on the chess board, it is impossible to give an accurate evaluation if one of the major pieces is going to be captured in the next turn. To improve the evaluation accuracy, quiescent search is used. The idea of quiescent search is to increase the search depth until a "quiet" position is reached e.g., where there is no more capture or recapture of pieces. Instead of calling a static evaluation, quiescent search is called if the remaining search depth reaches zero.

If the King is currently in check, a full negascout search will be called to search for possible moves to escape the check. In addition, if the previous move is a capture move; and the current capture move has the same destination as the previous, with the difference in value of pieces captured in both moves falling within a predefined recapture window, the Chinese chess engine will realize that a recapture has occurred and that there is an exchange of pieces. When this happens, a full negascout is also called, instead of relying on quiescent search alone. This improves the tactical strength of the AI. If the above conditions fail, all possible good capture moves are considered and quiescent search is called recursively until a quiet position (with no possible good capture moves, and the King not in check) is reached. The evaluation score for that position is then returned.

The performance of the quiescent search depends heavily on the definition of good capture moves. When good capture moves are defined too loosely, a quiescent search tree explosion results as too many unnecessary quiescent search nodes will be evaluated. This slows down the performance of the AI. However, if good capture moves are defined too strictly, the AI might miss out some important moves, which will affect the tactical strength of the AI negatively. In this work, two different types of good capture moves will be searched in the quiescent search function.

(1) Capturing undefended pieces.
Moves which capture undefended pieces are considered to be good capture moves, since they win material.

(2) Capturing valuable pieces with an attacker of lower or equivalent value.
Valuable pieces refer to those whose score in the position value table exceeds a pre-defined cut-off value. Generally, this refers to the King, Rook, Horse and Cannon.

A futility clause is also added into the quiescent search. If the sum of the static evaluation score of the current condition and the material value that will be won by the next capture move is still lower than the alpha, then it is likely that it is a bad line of play. Such moves are not searched and the static evaluation score of the current position is returned instead.

3.6 Pruning

Additional pruning techniques, such as the null move pruning [17], futility margin and razoring [18] are also used to further decrease the average branching factor of the search tree. These, unlike alpha-beta pruning, are "imperfect", in that they might affect the final outcome of the search. In other words, the search speed and depth can be improved at the expense of more inaccurate evaluation. For instance, too many unnecessary prunings that result from the use of very large pruning margins might cause potential tactical weakness to the AI and compromise its playing strength. On the other hand, margins that are too low might also result in too few prunings and limit the improvement in search efficiency. To obtain programs that are fairly good in playing strength and speed, there is a need to find a balance between efficiency and accuracy via fine-tuning these margin values.

3.7 Extensions

As a counter to the potential detrimental effects of pruning, selective search extensions during checks, re-captures or mate-threats are used. These, on the contrary, increase the tactical strength of the program at the cost of slowing down the search speed and can thus balance out the tactical weakness of imperfect pruning.

3.8 Evaluation

Evaluation of a position is divided into two parts: simple evaluation and adjustment score.

3.8.1 Simple Evaluation

Simple evaluation is based on the number of pieces and their individual positions on the game board. Specifically, seven position value tables are used in the program to compute the simple evaluation score. These tables are adapted from an open-source Chinese chess program, "Elephant-eye" [19]. Each type of Chinese chess piece, such as a Rook or a Horse, has its own position table. The position value table is indexed by the position of the piece. Thus each piece is assigned a position value score based on its position on the game board.

(1) If it is Black's turn to move
Simple evaluation score = $\sum \mathit{black\ pieces'\ position\,values} - \sum \mathit{red\ pieces'\ position\,values}$

(2) If it is Red's turn to move
Simple evaluation score = $\sum \mathit{red\ pieces'\ position\ values} - \sum \mathit{black\ pieces'\ position\ values}$

The simple evaluation score is used to assess the relative material strengths of both sides. However, in Chinese chess, which is also true for other forms of chess, material strength is not everything. The key to winning is through attacking and capturing the enemy King. While having more material helps, the side with the higher material strength might not necessarily win the game. This is where the adjustment score comes into play.

3.8.2 Adjustment Score

The adjustment score is a form of evaluation heuristics which is based on the relative positions and mobility of all pieces, and the King's safety. A list of adjustment scores that is used in the proposed Chinese chess engine is as follows:

(1) Mobility of Rooks and Horses.
The side which has more mobile Rooks and Horses is awarded a bonus score.

(2) Proximity of "bridges" for Cannons.
The Cannon can only maximize its effectiveness when there are bridges nearby. The side that has closer Cannon "bridges" is awarded a bonus score.

(3) Pinned pieces.
These refer to the opponent's pieces that are pinned by one's Rooks and Cannons. The side that pins more of the opponent pieces is awarded a bonus score.

(4) Cannons, with no Elephants.
If a side has no Elephants left, the opponent is given a bonus score for each of its Cannon.

(5) Rooks and Pawns, with no Advisors.
If a side has no Advisors left, the opponent is given a bonus score for each of its Rooks and Pawns.

(6) End game bonus for defensive pieces, Pawns and Horses.
In the end game, values of all the defensive pieces, Pawns and Horses are increased.

(7) Type of major piece.
At the start of game, players have three types of major piece: Rook, Horse and Cannon. However, these pieces decrease as the game progresses. A bonus score is thus awarded to the side which can preserve more types of major piece.

(8) King's safety.
The side that has a greater number of pieces threatening the opponent's King and the King's adjacent squares is given a bonus score.

(9) Quadrant score.
The board is divided into four quadrants. The net number of pieces in each quadrant is computed. For every quadrant, a bonus score is awarded to the offensive side, based on the number of its offensive pieces, relative to the opponent's defensive pieces in that quadrant. This helps the AI to recognize the importance of concentrating the attack on one flank, rather than dividing up its troops.

(10) Mean square distance.
For each player, the mean square distance between his major pieces is computed. A lower value for mean square distance means that the pieces are closer together, which will thus imply a tighter formation. A bonus score is awarded for the side which has a lower mean square difference.

(11) Mean distance from the opponent's King.
The mean position of major pieces for each side is computed. A bonus score is awarded to the side whose mean position is closer to his opponent's King.

The final evaluation score is computed as the sum of the simple evaluation score and the adjustment score.

3.9 Expansion of Databases

3.9.1 Opening Database

Originally, the opening book from Elephant Eye is used. It contains 20,618 moves, which were taken from past grandmasters' games and this roughly ensures that the opening book consists of high quality moves. However, the opening book also has its drawbacks – since common "opening traps" are seldom used in grandmasters' games, the opening database will not contain the correct reply moves for some of the common "opening traps". In order to incorporate such replies, moves from Yan's and Zhang's "The Formats and Principles of Chess Openings" [20] were added to increase the database to a total of 26,454 moves.

3.9.2 Endgame Database

An endgame database is a common feature in any successful chess program since it is difficult to derive an accurate heuristic to evaluate the game position during the endgame. When the checkers program, Chinook, defeated the human world champion, it had one to eight piece-databases. This means that when eight or fewer pieces were left on board, Chinook could always determine with 100% accuracy if it was going to win, lose or draw and it would always make the best move [2]. Large databases typically contain exact information on the optimal moves, the outcome of the game (win/lose/draw) and remaining moves to the end of the game. However, due to its sheer size and the complexity involved in creating and verifying the correctness of one, a simple, compressed endgame database based on Tu's "Examples of Chinese Chess Endgame" [21] is used instead to store a win, lose or unknown result based on the material balance on the board e.g., results of two red Pawns, one black Elephant and one black Advisor is unknown whereas two red Pawns and two black Advisors will generate a win. In the latter, the database does not return a value that represents a win but adds the value of three Rooks to the side about to win.

In a way, the database provides a heuristic to guide the program towards attaining a favorable material balance during the endgame. This is not without drawbacks. For one, the database does not account for piece positions e.g., Red might win if both Pawns are in the middle of board, but the end result will be a draw if they are at the bottom. However, since position value tables are used, the program will know that Pawns at the bottom are worth notably less and will thus avoid advancing the Pawns too fast. Another drawback is that the database does not contain the optimal move. This can, however, be alleviated by incorporating a good negascout search algorithm in the Chinese chess engine.

4 Genetic Algorithm

A GA is a search technique that is used in optimization problems and was first proposed by Holland [22]. It is based on the Darwinian model of survival of the fittest. Candidate solutions for the search problem are encoded as chromosomes and are evaluated by a fitness evaluation function. The fitter chromosomes are selected for reproduction. Search operators, such as crossover and mutation, are then applied to the new chromosomes.

4.1 Chromosomal Representation

Weights encoded in each chromosome for the evolution process are shown in Table 2:

Table 2. List of weights encoded in each chromosome.

Variable	Number of bits	Range of values	Remarks
K_d	5	0 to 31	Used to compute the bonus score for mean square distance, *x* between the friendly major pieces, as denoted by $K_d \cdot \exp(P_d / 16x)$.
P_d	4	0 to 15	
K_k	5	0 to 31	Used to compute the bonus score for proximity of friendly major pieces to the position of the opponent's King, as denoted by $K_k \cdot \exp(P_k / 16x)$, where *x* is the distance between the mean position of friendly major pieces and the opponent's King.
P_k	4	0 to 15	
K_r	5	0 to 31	Used to compute the bonus score for the Rook's mobility, as denoted by $K_r \cdot \exp(P_r / 16x)$, where *x* is the number of possible Rook moves.
P_r	4	0 to 15	
P_m	4	0 to 15	Used to compute the safety margin used in the razoring function $80 + 520[1 - \exp(-P_m / 16d)]$ during pruning, where *d* is the depth of the current search ply.
K_3	5	0 to 31	Used to compute the Quadrant score $K_3 \cdot \exp[P_3 / 16(x+5)]$ that awards additional bonus points when the program is able to obtain numerical superiority on a particular flank on the opponent's half of board. *x* is the material balance between the offensive and defensive pieces that are involved in the flank.
P_3	4	0 to 15	
V_r	4	0 to 15	Bonus score awarded for having friendly major pieces around the river-region on the chess board.
V_p	4	0 to 15	Bonus score awarded based on number of enemy pieces pinned by friendly Rooks or Cannons.
V_h	3	0 to 7	Penalty-score imposed for each "blocked" Horse leg i.e., immobile Horse.
C_d	5	-15 to 15	Change in the position value table scores for all defensive pieces.
C_c	5	-15 to 15	Change in the position value table scores for all Cannon pieces.

Table 2. (*continued*)

C_h	5	-15 to 15	Change in the position value table scores for all Horse pieces.
C_r	5	-15 to 15	Change in the position value table scores for all Rook pieces.

For the razoring function, the upper and lower bounds of this function is fixed such that it will always return a value between 80 (the value of a minor piece) and 600 (three times the value of a Rook), which adequately covers the generally accepted range of values for razoring. By imposing such bounds, the bit length of the chromosomes can be lowered, without affecting the accuracy of the result. The value of a Pawn is kept constant throughout the evolution process and used as a reference for the values of other pieces. For example, the value of a Rook is usually estimated to be nine times that of a Pawn. If the value of a Pawn is 10, the value of Rook will be 90. The value of the King is not encoded within the chromosome, as capturing the enemy King is equivalent to winning the game. Its value is kept constant at 1000, which is significantly higher than the value of other Chinese chess pieces.

The encoded chromosome is represented as a 71 bit string. To improve the efficiency of GAs, gray coding is used to encode the binary strings so that small changes in the integer values can be achieved by a minimal number of mutations and crossovers. At the start, 11 chromosomes are randomly generated and are used as the initial population.

4.2 Selection

Stochastic universal sampling is used as the selection algorithm. A spinning wheel is first constructed, with each chromosome represented by its own individual sector. The width of the sector is proportional to the fitness score of the chromosome. 10 equally spaced pointers are then randomly generated. A single spin of the wheel will be used to select the 10 individuals for the next generation of evolution, based on intersection points between the pointers and the sectors. This selection algorithm has zero bias [23], and thus helps to reduce the probability of premature convergence. The best individual from the previous generation is kept as an archive and left unchanged in the next generation.

4.3 Crossover

Recombination of chromosomes is done using two-point crossovers. This scheme has an advantage over single point operators in the sense that it allows for a combination of certain schemas, which are not possible under single point crossover.

4.4 Mutation

A mutation operator is used to flip the chromosomal bits based on a mutation probability. This allows for the exploration of the search space. For all offspring, each of their bits is inverted with a probability of P_{mutate}. In the initial phase of search, a higher P_{mutate} is required to explore a larger region of the search space. However, towards the end of the search, where the chromosomes are close to the optimal solution, a smaller P_{mutate} is used to prevent unnecessary disruption of good schemas. Overall, P_{mutate} is defined [24] as:

$$P_{mutate} = 1/[2 + (L-2)t/T]$$

Where:

L is the length of the chromosome,
t is the number of generation elapsed and
T is the maximum number of generations.

Under this scheme, P_{mutate} decreases with the number of generations elapsed, allowing for a high mutation probability in the beginning e.g., 0.5 and a lower mutation probability towards the end (approximately $1/L$).

4.5 *Fitness Evaluation*

As past games by Chinese chess grandmasters are used as the training data to guide the evaluation process, the fitness score of a chromosome is determined by the number of "correct" moves made, when benchmarked against the moves made by the grandmasters in an identical game position. When evaluating the fitness of chromosomes, Niching and sigma truncation [23] are used in combination to prevent any highly-fit individual from dominating the population too quickly. This helps to avoid the scenario where the search process converges prematurely on a local maximum.

4.5.1 Niching

Niching increases the population diversity and helps to prevent premature convergence by penalizing the fitness scores of chromosomes that are too similar to others, using the adjusted fitness score:

$$f'(x) = f(x) \cdot n \cdot L / x$$

Where:

$f'(x)$ is the adjusted fitness score,
$f(x)$ is the original fitness score,
n is the population size,
L is the length of chromosome and
x is the total number of bits that are identical to other chromosomes.

4.5.2 Sigma Truncation

Sigma truncation helps to keep the selection pressure relatively constant and prevents a single highly fit chromosome from taking over the population too quickly. It is used to scale the fitness scores of chromosomes based on the mean fitness and standard deviation of their fitness scores via the following expression [25]:

$$f'(x) = \begin{cases} f(x) - m + 2\sigma, \text{if } f(x) > m - 2\sigma \\ 0, \text{otherwise.} \end{cases}$$

Where:

$f'(x)$ is the scaled fitness score,
$f(x)$ is the fitness score, before scaling,
m is the mean of the fitness scores and
σ is the standard deviation of the fitness scores.

4.6 Local Search

Due to the complexity involved in evaluating a Chinese chess game, it is likely that the fitness might not adequately converge within 200 generations. Nonetheless, it is also not feasible to increase the number of generations excessively due to time constraints. To solve this problem, a local search algorithm is executed every 10 generations with the aim of improving the fitness of chromosomes.

A description of the local search algorithm is:

1) Make five copies of the fittest chromosome in the original population.
2) Select the three fittest chromosomes from the current population.
3) For all the set of genes which encode a particular variable, (macrogene), check the degree of similarity in the three fittest chromosomes.
4) If the macrogene is highly similar between the three (maximum difference is less than 20% of the value of the macrogene in the fittest chromosome), do nothing. Otherwise, mutate the particular macrogene for the five new copies with a probability 1/L, where L is the bit length of the macrogene.
5) Repeat steps 3 and 4 until all macrogenes are evaluated.
6) Evaluate the fitness scores of the five new chromosomes.
7) If the fittest of the five chromosomes ($Best_{new}$) has a higher fitness score than that of the fittest chromosome in the original population ($Best_{original}$), replace $Best_{original}$ with $Best_{new}$, and use $Best_{new}$ as the archive for the next generation.

The local search algorithm assumes that if the three fittest chromosomes exhibit a similar value for a particular macrogene, then that value must be reasonably good and should be kept. Otherwise, mutation is applied to enable exploration in the vicinity of the fittest chromosome. If a fitter chromosome is found in the process of exploration, it can be used to improve the archive without affecting the genetic diversity of the population adversely.

5 Training Data

Before any simulation can be performed, the training data which consists of past games of grandmasters has to be selected for use.

5.1 Requirements for the Training Data

As each grandmaster is likely to be unique in his playing style and individual preferences, using moves from more than one grandmaster in the same training dataset might lead to potential conflicts in the eventual playing style of the evolved Chinese chess program. As such, only moves made by a single grandmaster are included in the training set. To ensure the quality of moves in the training dataset, some intrinsic requirements must be met:

(1) Games must be played against high caliber opponents in a competitive setting. All games are taken from past Five Rams Cup (五羊杯) tournaments where participation is by invitation only. As all players are past national champions, it is considered one of the most prestigious titles in the Chinese chess arena.

(2) Games chosen must reflect an even strategy mix e.g., the red side usually plays in a more offensive style while black takes up a more defensive stance. To prevent the evolved program from becoming overly aggressive or defensive, half of the training games will involve the grandmaster playing as Red, while the other half Black.

(3) There must not be any poor or sub-standard move in the training dataset. It is difficult to guarantee the fulfillment of this requirement since the identification of relatively bad moves requires a high degree of expertise which the authors do not possess. Moreover, the objective of using a GA instead of hand-tuning is to eliminate the need for such knowledge. To address this issue, the end-result of a game is used as a gauge. Only games in which the grandmaster has achieved good results were used. Intuitively, only games won by the grandmaster should be included. Nonetheless, the winning side typically possesses the upper hand throughout the game and is hence playing in a more aggressive and dominant style. To ensure a good mix of playing styles, drawn games are also used so as to allow the chromosomes to learn the means to play defensively in even or disadvantaged positions. As Red has an advantage

over Black, by virtue of its initial move, drawing as Red can also be considered as bad performance and games drawn as Red are not included. In summary, only games won as Red, games won as black and games drawn as black are included as part of the training dataset.

(4) The three grandmasters from whom the training data are extracted are:
- GM Hu Rong Hua – who won his first national Chinese chess title at 15 (youngest title holder); national Chinese chess title for a record 10 consecutive times; and is the record holder for the most number of national titles won (14 in total).
- GM Lu Qin – who won the prestigious Five Rams Cup a total of nine times.
- GM Xu Yin Chuan – who is a dominant player in recent years and has won the latest Five Rams Cup in 2007.

5.2 Building and Selecting the Training Data

As programs can rely on databases for the opening and ending phases of the game, the focus of the evolution process will thus be on the late-opening to mid-game phase. As such, only moves from round 15 (inclusive) to round 30 (exclusive) are used for training. To keep the simulation time within reasonable limits, only 14 games were chosen in total. Due to the relatively small number of games, a large random element is involved such that the chosen games might not be the most suitable for evolving an optimal program using the GA. To counter this problem, a 14-game dataset was built for each of the short-listed grandmasters according to the requirements in Section 5.1. The un-evolved program was benchmarked against the three datasets – one from each grandmaster, using 5s of computation time per move on a Pentium 4 PC with 512 megabytes RAM. Three GA simulation runs were performed for each dataset and the results are shown in Table 3.

Table 3. Performance of un-evolved program, when benchmarked against the three training datasets extracted from grandmasters.

Training Data	Correct moves
Hu	72 / 210
Lu	100 / 210
Xu	103/ 210

A 50 generation preliminary evolution was then carried out to investigate the most appropriate training dataset to employ, marked by the largest improvement over the performance of the un-evolved program. The general settings that were used for all the GA simulations are given in Table 4.

Table 4. Settings for GA simulation.

Attribute	Value
Population Size	10 + 1 archive
Crossover	2-point crossover with a crossover probability of 0.7
Mutation rate	$1/[2+(L-2)t/T]$, where L is the length of chromosome, t is the current generation and T is the maximum generation
Selection	Stochastic universal sampling
Move time	Each chromosome is given 5s of computation time per move

5.3 Results of Preliminary Experiment

Results of the preliminary experiment showed that the training dataset from Hu produced a best evolved fitness score that revealed the highest percentage of improvement (Table 5) from the corresponding fitness score of the un-evolved player (Table 3), and was used for the final GA simulations. However, this does not imply the superiority of Hu's games for training all Chinese chess programs in general. It merely suggests that this particular training dataset demonstrates the highest potential of improvement for the un-evolved Chinese chess program, relative to the datasets that are extracted from the other two grandmasters.

Table 5. Best fitness score obtained (out of three runs and 50 generations).

	Best Fitness Score	Percentage Improvement (%)
Hu	101	40.3%
Lu	118	18.0
Xu	116	12.6

6 Results Simulation

After finalizing the training data, the GA is applied to improve the performance of the existing Chinese chess program.

6.1 Changing the Fitness Evaluation

Based on the 72 out of 210 (34%) correct moves that are obtained when benchmarking against Hu's training dataset, the un-evolved program is able to get 27 out of 38 (71%) of the capture moves correct, but only 45 out of 172 (26%) correct for the non-capture moves. The significantly higher percentage of correct capture moves than non-capture ones is due to the fact that computer chess programs are typically strong in the tactical aspect, which allowed them to find the correct capture moves without much difficulty. With regards to the 29% error percentage for capture moves, a large part is attributed to the fact that the program uses a different piece to execute the capture move e.g., given the option of using either a Horse or Cannon to capture an undefended Pawn, Hu might have chosen to use the Horse while the computer the Cannon instead.

On the other hand, making the correct non-capture moves is much more difficult as the program will have to pick from a much larger array of seemingly equally good moves most of the time. To address this problem, five points were awarded for correct non-capture moves and four for correct capture moves during the final simulation. This higher weighting on the former steered the program to place a greater emphasis on learning correct non-capture moves, instead of correct capture moves which it is already fairly good at identifying. The reason for not reducing the weight value of capture moves further is to maintain the tactical strength of the Chinese chess program and avoid situations where it might over-emphasize positional advantage and sacrifice pieces unnecessarily. Under the new weights, the fitness score of the un-evolved Chinese chess program was 357.

6.2 Evolving the Chinese Chess Program

The GA was applied using the same settings as before for 200 generations using the new weights for capture and non-capture moves. Five simulation runs were performed and the fitness scores of the evolved players are listed in Table 6.

Table 6. Results of evolution – fitness scores of evolved players (200 generations).

Player	Fitness Score
Evolved Player 1	480
Evolved Player 2	463
Evolved Player 3	479
Evolved Player 4	451
Evolved Player 5	467

It is observed that the highest fitness score among the five evolved players is 480, which is a 34.5% improvement over the fitness score of 357 for the un-evolved player. Even the evolved player with the lowest fitness score of 451 produced a 26.3% improvement. This indicates that effective evolution is indeed taking place. However, it is observed that the maximum fitness achievable is 1012, but the highest evolved player score was only 480, which is comparatively much lower. This is attributed to three causes.

Firstly, grandmasters usually make their moves after thinking 10 moves in advance [13]. By limiting the program to 5s of move time, its search depth is limited to only six or seven. This relatively shallow search is likely to cause the program to make moves that are different from those of the grandmaster. Secondly, the grandmaster might have also chosen different styles against different opponents. Since the training data was selected to reflect a blend of dissimilar playing styles, offensive and defensive, the resultant conflict in styles increases the difficulty for the program to play all 210 moves correctly. Thirdly, only a small number of variables are evolved throughout the evolution process. On the contrary, the grandmaster's evaluation process probably uses more evaluation functions which might be different from what was used in the proposed Chinese chess engine. As such, it becomes almost impossible to make the exact same moves as the grandmaster. However, as the objective of evolution is not to predict the grandmaster moves, but rather to increase the overall playing strength of the Chinese chess program; the fact that the program is unable to get most of the moves correct should not pose a problem, so long as the evolved program can exhibit an increased playing strength eventually.

7 Evaluating the Results

Unlike the conventional means of improving fitness via playing games, using the training dataset of moves from past grandmasters' games as a benchmark for evaluation takes an indirect approach to improve the playing strength of the Chinese chess program. There is a need to verify if its playing strength has really been improved.

7.1 Subjective Measurement

The playing strength of the evolved players could be subjectively measured by pitting the evolved and un-evolved players against each other. Each would play 200 games (100 as red, 100 as black) against every other player. To prevent games from dragging on for too long, those that lasted more than 200 moves were terminated and a draw recorded. Each player was allowed 5s per move. As games could be deterministic, an opening database was used to ensure that games would start from random positions. In positions, especially those that can be reached in the initial rounds of the game, the opening database typically contained more than one possible move for each position and will return a move with a probability based on the win/loss record of the move in the grandmasters' games. Usage of the database thus allows the players to play each game with a random opening line.

The probability that a player would probe the opening database was defined as $(1-t/30)$, where t was the number of turns elapsed in the game. The minimum value of the probability was bounded at 0.5 i.e., if $(1-t/30)<0.5$, the probability would be set as 0.5. The probability was higher in the initial phase as moves made then would directly determine the choice of opening line. A higher probability was used for the first few turns to prevent the games from being too deterministic. However, as more moves were made, the probability of probing the database was lowered gradually so as to allow the program to rely more on its internal evaluation function.

Using the above settings, simulated results were obtained. The overall performance of all players, both evolved and un-evolved, as they play against one another,

Table 7. Overall performance after all players play 1000 games each.

Player	Wins	Draws	Loss	Wins - Loss
Un-evolved	321	311	368	-47
Evolved Player 1	395	330	317	36
Evolved Player 2	306	342	352	-46
Evolved Player 3	353	317	288	107
Evolved Player 4	357	339	304	53
Evolved Player 5	295	307	398	-103

Table 8. Performance against the un-evolved player.

Player	Fitness	Wins	Draws	Loss	Z-Value
Evolved Player 1	480	77	63	60	1.45
Evolved Player 2	463	76	59	65	0.93
Evolved Player 3	479	78	63	59	1.62
Evolved Player 4	451	73	59	68	0.42
Evolved Player 5	467	64	67	69	-0.43

is shown in Table 7 while the performance of the evolved players against the un-evolved one is given in Table 8.

Although the evolved players generally had much larger fitness scores than the un-evolved player, results showed that evolved Player 5 was unable to beat the un-evolved one (Tables 7 and 8). In addition, even the strongest player, evolved Player 3, could only beat the un-evolved player at the 94% confidence level. Closer inspection of the evolved players seemed to reveal problems with the values of the razoring function. In the Chinese chess engine, razoring was defined such that when the sum of the current simple evaluation score and razoring margin at the current depth was smaller than the alpha cut-off, the position was probably so bad that it would not be reached in the actual game; and hence, the remaining depth for the current line of search could be safely reduced by one.

For the five evolved players, the value of P_m in the razoring function ranged from 10 to 15. At a search depth of five, the safety margin for players would range from 577 to 595, which was close to three times the value of the most powerful piece – the Rook. As it was extremely rare for a side to obtain an advantage that was three Rooks above the alpha-cut-off, the evolved players would probably not receive great time savings from the razoring-prunings. In contrast, the razoring margin for the un-evolved player at the same depth is 200, which is much lower than that of the evolved players. While this meant that the tactical strength of the un-evolved player might be weakened, this disadvantage could be overcome by its deeper searches. As such, the un-evolved player was able to perform well against the evolved players despite its weaker evaluation function.

The poorly evolved razoring function that might have arisen as the training dataset did not provide adequate selection pressure for the macrogene which encoded P_{mutate} . This is because any additional ply of search depth that the razoring provided might not actually affect the total number of correct moves that were made by the Chinese chess program eventually. For instance, suppose the program could reach a depth of five in 5s with a bad razoring function and was able to find the correct move in five ply-search as a result of its good evaluation function. In this case, the program would still find the same correct move even when the razoring function was improved such that the program could subsequently reach a search depth of six in 5s. On the contrary, if the evaluation function was inherently bad and caused the program to move H0-G2 in a particular position, as opposed to the correct move C3-C4; the program might still be unable to locate the correct move despite the use of an improved razoring function and search depth.

Thus, it could be understand that the correctness of the evaluation function dominated the razoring function during the GA simulation. However, once the programs were pitted against each other, the advantages of a good razoring function would come through. The player with a weaker evaluation function could have a better razoring function that would enable it to search one ply deeper; and in turn allow it to find a winning line of play or a combination of moves that led to the capture of an opponent piece. Following the same line of reasoning, the evolved players that had somewhat random razoring functions were thus placed in an unfavorable position when playing against the un-evolved player.

To verify the validity of the hypothesis (which states that the razoring function was poorly evolved) and evaluate the playing strengths of the evolved and un-evolved players, all players would be adjusted to use the same razoring function (e.g., the one used by the unevolved player) for all the move decisions during the actual game play. The results of the simulation are presented in Tables 9 and 10.

Table 9. Overall performance after all players play 1000 games each (with revised razoring margin).

Player	Wins	Draws	Loss	Wins - Loss
Un-evolved	287	317	396	-109
Evolved Player 1	382	298	320	62
Evolved Player 2	292	363	345	-53
Evolved Player 3	409	306	285	124
Evolved Player 4	360	292	348	12
Evolved Player 5	320	324	356	-36

Table 10. Performance against the un-evolved player (with revised razoring margin).

Player	Fitness	Wins	Draws	Loss	Z-Value
Evolved Player 1	480	83	58	59	2.01
Evolved Player 2	463	74	65	61	1.12
Evolved Player 3	479	94	62	44	4.26
Evolved Player 4	451	76	57	67	0.75
Evolved Player 5	467	69	75	56	1.16

After the adjustments were made, all the evolved players were able to beat the un-evolved player as observed in Tables 9 and 10. In particular, the strongest player, evolved Player 3, was able to defeat the un-evolved player at the 99.99% confidence level, winning 47.5% of the games. The above simulated results not only demonstrated that the playing strength of the program was significantly improved by the GA, but also ascertained that the razoring function was not adequately trained by the dataset.

To achieve sufficient training for the razoring function, one possible alternative was to evolve it using co-evolutionary approaches [26] and evaluating the fitness of the chromosomes by playing games between them. However, such schemes might be plagued by associated problems of low selection pressure and intra-sensitivity etc, as stated in the introductory section. Even if the above could somehow be resolved, one additional problem remained – razoring was dependent on the current depth of search. In tournament time control, the Chinese chess program might make a move every few minutes and hence, be searching at a depth of 10 or more plies per move. If the Chinese chess program was only allowed 5s to make a move during the evolutionary phase, it might only reach a search depth of five or six plies, with no selection pressure for the higher plies. This would likely entail an inaccurate razoring function for plies seven and above. To overcome this problem, the program must be given at least a minute (or an amount of time required to reach 10 plies) per move, which would unrealistically increase the amount of time required for evolution. Taking all factors into consideration, the original razoring function from the un-evolved player was thus retained and used in the final Chinese chess program.

Simulated results in this setup also showed that the evolved players with slightly lower fitness scores were able to outperform those with higher scores e.g., evolved Player 3 – with a fitness score of 479, was able to outperform evolved Player 1 – which had a higher fitness score of 480. The fitness score was thus not an absolutely accurate indicator of playing strength, especially when the scores were close to one another. However, if the difference in fitness scores was significant (as in the case of the score disparities between the evolved and un-evolved players), the fitter player could indeed outperform the less fit one. This suggested the crucial need to conduct multiple instances of the simulations in order to evolve the best Chinese chess program. This problem might also be alleviated by increasing the size of the training data, which would in turn improve the resolution of the fitness evaluation function.

7.2 Objective Measurement

Playing games among a small pool of players only provides a subjective measurement of playing strength. Although random moves from the opening book are used, a player will still be able to win a large proportion of the games against another, so long as it is able to exploit a particular vulnerability of his opponent's evaluation function consistently. Thus, such a measurement might not be indicative of the difference between the players' overall playing strengths. An objective measurement is required for more accurate assessment.

To obtain an objective measurement, performance of the evolved and un-evolved players needs to be benchmarked against a large number of random players, each with its own unique blend of strengths, weaknesses and playing style. This is done by having the evolved and un-evolved players setup to play against human players online at a website named Club Xiangqi (http://www.clubxiangqi.com), which uses the Elo scoring system – a statistical rating system where the points lost/won by players are determined by their relative scores/ratings. For instance, defeating an opponent who has a higher rating will earn a player more points than when he plays with one of only mediocre rating. Similarly, losing to an opponent of lower rating will also cost the player more points. As the playing histories of players are considered, it is possible to obtain a reasonably accurate objective estimation of one's playing strength by sampling a small number of games.

Since Club Xiangqi operates through java applets, there is no convenient way of automating the game-playing process between the Chinese chess program and the human opponents. As such, the author acts as the "go-between" during the course of the games e.g., when the human opponent makes a move on the Club Xiangqi window, the author will translate the same move into an equivalent move in the window of the Chinese chess program, as though the human opponent is playing directly with the program. Once the program makes its reply, the author will execute that reply on the Club Xiangqi window, on its behalf. This process will continue until an entire game is completed. Owing to the long playing time involved, only the strongest evolved player (evolved player 3) and the un-evolved player are tested online over a span of 50 games.

Whenever possible, games are played in public tables e.g., those that operate on the principle of "winner stays, and loser cannot play", to ensure a steady stream of random opponents instead of constant rematches against the same opponent. Each opponent has between one to five minutes per move, depending on the settings of the public table, of which the author has no control. Similarly, the total time allocated to each player varies from 10 to 25 minutes. Regardless of these settings, the program is configured to make a move in five seconds with pondering enabled for both the evolved and un-evolved players. Instead of idling, the pondering mode allows the Chinese chess program to evaluate the board position actively during its opponent's move time.

Results of the simulation are shown in Table 11. Although evolved player 3 only won four games more than the un-evolved player, its score was nonetheless 93 points higher. This notable win suggested that most of the victories came when evolved player 3 was playing against higher-calibre opponents. In addition, evolved player 3 was also able to secure a winning streak for all its initial 21 games. To put the scores into perspective, the average score of a human Chinese chess player is only 1500, which implied that the evolved player 3's score of 1877 actually placed it at a much higher level than most human Chinese chess players. The fact that the evolved Chinese chess program was still able to obtain such respectable score despite a move time of only 5s suggested that it had acquired a reasonably high level of playing strength through the evolutionary process.

Table 11. Performance at Club Xiangqi.

Player	Win	Draw	Loss	Score *
Un-evolved	38	2	10	1784
Evolved Player 3	42	2	6	1877

* Score of an average player in the website is 1500.

8 Conclusions

In conclusion, the chapter explores the feasibility of using past grandmasters' games as training data to improve the playing strength of a Chinese chess program. Although the razoring margin failed to be adequately tuned during experimentation, evolved players do demonstrate superior playing strength over the unevolved one. Significant improvements are obtained despite the fact that only a small number of parameters were evolved. Overall simulation results highlight the tremendous potential of using GAs to increase a program's playing strength by evolving its evaluation function closer to that of the grandmasters. In addition, the subjective and objective fitness tests also indicated that grandmaster games can be effectively used to increase the playing strength of Chinese chess programs.

A drawback that is associated with the usage of training datasets in such guided learning approaches pertains to the fact that the accuracy of the evaluation function is essentially limited by the skill of the grandmaster e.g., the evaluation of Chinese chess programs can only become as good as that of the grandmaster but cannot improve beyond what was achieved by the grandmaster himself. The current performance bottleneck of the Chinese chess programs is thus the evaluation function. Once the program is capable of evaluating accurately, it will be in a better position to compete with human grandmasters through its superior search speed and infallibility (i.e., no human fatigue).

9 Future Work

One limitation of the current work pertains to its relatively small scale. In a bid to keep the evolution time within reasonable limits, the population size, number of generations, chromosome length (which directly affects the number of macro-genes) and size of the training dataset were kept minimal. To build a Chinese chess program that truly plays at the grandmaster level, a GA evolutionary process of much larger scale is required. While the population size, number of generations and size of training data can be expanded directly, it might not be worthwhile to increase the chromosome length simply by increasing the number of evaluation function variables which are encoded within it. This is because it is extremely difficult, if not impossible to know the exact evaluation rules or functions that are used by grandmasters, much less the appropriate types of mathematical functions to model them. For example, modelling rules as exponential functions are likely to be too simplistic.

A better and possible improvement to the existing work is to use neural networks as part of the evaluation function in the Chinese chess program. A neural network models the human brain and has the potential to encode complex evaluation functions that might be impossible to model otherwise. By building neural networks over important areas of the chess board, such as the river and palace regions, GAs will be able to better approximate the grandmaster's evaluation function. However, due to the increase in simulation time, distributed computing should be used instead as it will no longer be feasible to use only a single computer per evolution. In the distributed framework, each computer in the network will be tasked to evaluate the fitness of a single chromosome and a master computer will then collect all the fitness scores from these auxiliary computers and evolve the next generation of chromosomes. Such increase in the speed of evaluation will allow the GA to be employed on a much larger scale, and this will hopefully entail an increase in the potential playing strength of the evolved Chinese chess programs.

References

[1] Bierman, A.: Theoretical issues related to computer game playing programs. Personal Computing, 86–88 (1978)
[2] Schaffer, J.: One Jump Ahead. Springer, New York (1997)
[3] Newborn, M.: Kasparov versus Deep Blue. Springer, New York (1997)
[4] Yan, A. (ed.): Chinese-chess computer crushes grandmasters. China Daily, `http://en.chinabroadcast.cn/2946/2006/08/10/167124918.htm` (accessed October 1, 2006)
[5] Chen, X.: First human vs computer Chinese chess challenge: Neuchess wins by a narrow margin of 11-9. 首届中国象棋人机大**战见分晓浪潮天梭**11比9 **险胜** (2006), `http://www.ciw.com.cn/News/hotnews/2006-08-16/7478.shtml` (accessed October 1, 2006)
[6] Thompson, K.: Computer chess strength. In: Clarke, M. (ed.) Advances in computer chess 3. Pergamon Press, Oxford (1982)
[7] Hamilton, S., Garber, L.: Deep Blue's hardware-software synergy. IEEE Computer 30, 29–35 (1997)
[8] Duran, F.: Advance and be mechanized (2003), `http://www.fduran.com/wordpress/?p=10` (accessed October 10, 2006)
[9] Shannon, C.: Programming a computer for playing chess. Philosophical Magazine, 256–275 (1950)
[10] Hsu, F., Anantharaman, T., Campbell, M., Nowatzyk, A.: A grandmaster chess machine. Scientific American 263, 44–50 (1990)
[11] Fogel, D.: Blondie24: Playing at the edge of AI. Morgan Kaufmann, San Francisco (2002)
[12] Fogel, D.: Evolutionary computation: Toward a new philosophy of machine intelligence. IEEE Press, New York (2005)
[13] Hsu, S.: Introduction to computer chess and computer Chinese chess. Journal of Computer 2, 1–8 (1990)
[14] Moreland, B.: Alpha-beta search (2002), `http://www.seanet.com/~brucemo/topics/alphabeta.htm` (accessed October 3, 2006)
[15] Reinefeld, A.: Spielbaum-Suchverfahren. Informatik-Fachbericht 200. Springer, Berlin (1989)
[16] Tu, Z., Zong, L.: (2003) (accessed October 5, 2006)

[17] Donniger, C.: Null Move Forward Pruning (1993)
[18] Heinz, E.: Scalable search in computer chess: Algorithmic enhancements and experiments at high search depths. Viewag-Verlag, Wiesbaden (2000)
[19] Anon, Source code for open-source Chinese chess program: Elephant Eye (2006), `http://www.elephantbase.net/download/xqwizard_source.7z` (accessed October 2, 2006)
[20] Yan, W., Zhang, Q.: The formats and principles of chess openings. 布局定式与**战理**. Beijing University of Physical Education (2004)
[21] Tu, J.: Examples of Chinese chess endgame. 象棋残局例典. Shanghai (1990)
[22] Holland, J.: Outline for a logical theory of adaptive systems. Journal of the Association for Computing Machinery 3, 197–217 (1962)
[23] Baker, J.: Reducing bias and inefficiency in the selection algorithm. In: Proceedings of the second international conference on genetic algorithms and their applications. Lawrence Erlbaum Associates, New Jersey (1987)
[24] Bäck, T., Schütz, M.: Intelligent mutation rate control in canonical genetic algorithms. In: Ras, Z., Michalewicz, M. (eds.). Lecture notes in artificial intelligence. Springer, Berlin (1996)
[25] Goldberg, D.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Massachusetts (1989)
[26] Ong, C., Quek, H., Tan, K., Tay, A.: Discovering Chinese chess strategies through coevolutionary approaches. In: Proceedings of the IEEE symposium series on computational intelligence: Computational intelligence and games. IEEE Press, Piscataway (2007)

Author Index

Subject Index