Deploying Metaheuristics for Global Optimization

M. Davarynejad

Deploying Metaheuristics for Global Optimization

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op 18 juni 2014 om 15:00 uur
door Mohsen Davarynejad
Master of Science in Electrical Engineering
Ferdowsi University of Mashhad, Iran
geboren te Sary, Iran.

Dit proefschrift is goedgekeurd door de promotoren:
Prof.dr.ir. J. van den Berg

Copromotor: Dr.ir. J.L.M. Vrancken

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof.dr.ir. J. van den Berg	Technische Universiteit Delft, promotor
Dr.ir. J.L.M. Vrancken	Technische Universiteit Delft, copromotor
Prof.dr. D.E. Goldberg	University of Illinois at Urbana-Champaign, USA
Prof.dr.ir C.W. Oosterlee	Centrum voor Wiskunde en Informatica (CWI)
Prof.dr.ir U. Kaymak	Eindhoven University of Technology
Prof.dr.ir. J.N. Kok	Leiden University
Prof.dr.ir. C. Vuik	Technische Universiteit Delft
Prof.dr.ir M. Reinders	Technische Universiteit Delft, reservelid

The research described in this thesis received funding from the European Communitys Seventh Framework Programme within the “Control for Coordination of Distributed Systems” (Con4Coord - FP7/2007-2013 under grant agreement no. INFSO-ICT-223844).

Published and distributed by: M. Davarynejad
WWW: http://davarynejad.com/Mohsen/

Cover design: Ehsan Davarynejad, WWW: http://davarynejad.com/Ehsan/

ISBN 978-90-5584-173-8

Keywords: metaheuristics, fitness approximation, fuzzy granulation, simulated big bounce, center-seeking bias, initialization region bias.

Printed in the Netherlands

تقدیم به پدرم که ستودنی است.

تقدیم به مادرم که تمام زندگی من است.

تقدیم به همسرم که لیلی است.

Acknowledgements

The road to understanding ourselves as human beings and mother nature is not paved. To keep from being overwhelmed by the bewildering scale and challenges of the journey, we have designed some stations along the way, one of which is *PhD station*. My main concern during my studies has not been to find the best way. Instead, I have tried to find "a way", for which, like anybody, I needed a mentor to put me in the right direction, and colleagues, friends and families to stand by me on my journey. I have been very fortunate in both respects.

First of all I would like to express my deepest gratitude to my promoter Jan van den Berg, who has been a tremendous help to me on my way to the PhD station, for his continuous support and guidance in this path, and for his patience, enthusiasm, and immense knowledge that has helped me a lot. I have been fortunate to have a mentor who gave me the freedom to explore on my own and to develop my scientific work independently. Not only did I enjoy our scientific discussions, but also the numerous personal talks and free lectures: you talked with great enthusiasm about big data, inductive bias, cyber security, etc. I enjoyed all of our meetings: thanks for all the good advice and discussions! Jan, it was a great pleasure working with you!

I would like to express my deep gratitude to Jos Vrancken for his guidance in conducting this research. I have been very much appreciated his willingness to give his time, his enthusiastic encouragement along with his constructive critiques.

I would like to thank my brother Ehsan and my best friends Jafar and Ebrahim for the many prolific brainstorming discussions we had.

During my PhD, I had the opportunity to work on some other projects and scientific papers with some of my best colleagues, graduate students and friends: Carlos Coello Coello, Ehsan Davarynejad, Sobhan Davarynejad, Gary Fogel, Ebrahim Rahimi, Jafar Rezaei, Chang Wook Ahn, Andreas Hegyi, Ewa Snaar-Jagalska, Yubin Wang, Vincent Marchau, Jelmer van Ast, Ron van Duin, Guido van Heck, Maarten Janssen and Zary Forghany from whom I learned a lot. Thank you all!

I had the pleasure to co-supervise a number of very bright graduate students. I learned how to supervise projects from the collaboration with Jan van den Berg supervising the projects of Guido van Heck, Mohamad Alamili, Maarten Janssen and Antonio Spadaro. I learned a lot from you guys.

Although I have been complaining about many relocations not only in our building, but also within sections, I should have celebrated it. I had the chance to share my office with several brilliant colleagues, with whom I discussed many interesting ideas and shared wonderful time. Thank you Andreas Schmidt, Yusasniza Mohd Yunus, Ebrahim Rahimi, Sam Soleimani, Devender Maheshwari, Thieme Hennis, Tanja Buttler, Yakup Koç, Mingxin Zhang, Reza Haydarlou, Evangelos Pournaras, Çagri Tekinay and Yilin Huang.

Finally I would like to thank all my family and friends who supported me. I would like

to thank my friends, Mohammad and Saeed, for asking zillions of annoying, but critical and useful, questions about my work. Furthermore I would like to thank Zary, Sara and Elnaz for keeping them off my back every once in a while.

زری نازنینم، دو چیز هیچ گاه یادم نمیره. اولیش اون ۱۰ ماه دوران انتظار، و دومیش ۴۰ روز تنها بودن در هلند. تنها چیزی که اون چهل روز رو برام قابل تحمل میکرد، امید به اومدن تو به اینجا بود. ممنونم از اینکه من رو تنها نذاشتی، و ۶ سال اقامت در هلند رو برام قابل تحمل کردی.

مامان و بابای عزیزم: از اینکه شیرین‌ترین راه زندگی کردن که همون زحمت کشیدن و هراس نداشتن از مشکلات بود رو به من یاد دادید سپاسگزارم. کاش همه دوباره دور هم جمع بشیم، و قطار (در حال حاضر) ۱۲ نفریمون رو دوباره به حرکت در بیاریم. دلم فقط به امید اون روز زنده است.

احسان، ملیحه و سبحان عزیز: از همتون ممنونم. از احسان به خاطره اینکه اجازه نمیده جعبه سیب حتی یک روز دوام بیاره، از ملیحه به خاطر مهربونی ها و از سبحان به خاطر اینکه با مامان و بابا به پیاده روی میره.

از حسین، زینب و آزاده به خاطر نبودن ولی همدلی‌هاشون ممنونم.

سوده عزیزم: تو چه زود بزرگ شدی، و من چه ساده لذت بزرگ شدن با تو رو از دست دادم. حالا دیگه سوده خانم شدی، ولی هنوز دوست دارم شیطنت‌های بچه گونه‌ات رو ببینم. ممنون از اینکه در طی این چند سال عکس‌های خوشگلت رو برای من میفرستادی، تا کمی دنیای کودکانه‌ات رو بهتر درک کنم. سهم من از لذت دیدن تو کمتر از اون چیزی بود که تصور می‌کردم. برای بزرگ شدن عجله نکن عزیزم.

از النا کوچولو و یاسمن به خاطر شیرین کاریها و مهربونی هاشون ممنونم.

قدردان همه هستم: مامان جون و بابا جون به خاطر حمایت‌ها ومهربونی هاشون؛ سرور به خاطر خنده هاش؛ داداش محمد به خاطر بازی قشنگش در نقش باجناق؛ دانیال به خاطر آتیش بازیاش؛ نازنین به خاطر نامه‌های قشنگش؛ بابا جان و مادر جان به خاطر بزرگوارياشون؛ خاله شوکت و اکبر آقا به خاطر اذیت کردناشون؛ حسین، حامد و مهناز به خاطر همراهی هاشون و دایی حسین به خاطر ایمیل هاشون.

Mohsen Davarynejad,
The Hague, January 2013.

Contents

"If I have seen further it is by standing on the shoulders of giants."

Sir Isaac Newton - 1675

1 Introduction

When resources are limited in nature, competition is inevitable. This observation also holds for society as we can conclude from competition occurring in business, politics, etc. due to limitations of available resources. Competition over limited resources enforces exploration of possibilities to improve the current status quo. Achieving optimal utility out of limited resources is often desired and sometimes, crucial. This exploration process, which takes as its objective to improvement any given current situation, is referred to as *optimization*. In an ideal situation, the objective of optimization is to find values for a vector of parameters that *minimize* or *maximize* a given objective function subject to given constraints [7]. A vector of parameters where all constraints are satisfied is called a *feasible solution*. Feasible solutions to the optimization problem are *optimal solutions* when their objective function value(s) are superior to those of any other feasible solution. Optimization problems are ubiquitous, from planning a route for a "Zombie walk" to designing a strong but light airplane wing, and from holiday planning to finding a secret message hidden in a signal.

Optimization encompasses maximization and minimization of an objective function $f_0 : E \to \mathbf{R}$ where $E \subseteq \mathbf{R}^D$ and D comprise the dimensions of the search space E. A maximization problem can be transformed into a minimization problem and *vice versa* by taking the negative of the objective function. The terms maximization, minimization and optimization, therefore, are used interchangeably throughout this thesis.

A single-objective optimization problem can be defined as follows[1] [7]

$$\begin{aligned} &\text{Given } f_0 : E \to \mathbf{R} \text{ where } E \subseteq \mathbf{R}^D \text{ and } D \text{ is the dimension of the search space } E \\ &\text{find } \mathbf{x}^* \in E \text{ such that } f_i(\mathbf{x}) \leq b_i,\ i = 1,\ldots,m \\ &\text{and } f_0(\mathbf{x}^*) \leq f_0(\mathbf{x}),\ \forall \mathbf{x} \in E. \end{aligned} \tag{1.1}$$

[1] For now we will be focusing on single-objective optimization.

Here the vector $\mathbf{x}$ is the *optimization variable* of the problem, the function f_0 is the *objective function*, the functions $f_i : \mathbf{R}^D \rightarrow \mathbf{R}$, $i = 1,\ldots,m$, are the *constraint functions*, b_i, $i = 1,\ldots,m$ are *bounds* for the constraints, and vector $\mathbf{x}^*$ is the *global optimal solution* of f_0. $\mathbf{x}_B^*$ is a the *local optimal solution* of the region B when $f_0(\mathbf{x}_B^*) \leq f_0(\mathbf{x})$, $\forall \mathbf{x} \in B$, where $B \subset E \subseteq \mathbf{R}^D$. Note that when dealing with unconstraint problems $E = \mathbf{R}^D$.

Also interchangeable are the terms *optimization variable*, *decision variable* and *design variable*. They refer to the vector $\mathbf{x}$. We also use *objective function*, *fitness function*, *cost function* and *goodness* interchangeably to refer to f_0. Vectors are set in bold face throughout this thesis.

Optimization problems encountered in practice appear in various types and with various mathematical properties. As an example, the optimization problem is called a *linear program* if both the objective function f_0 and the constraints are linear, i.e., satisfy [7]

$$f_i(\alpha\mathbf{x} + \beta\mathbf{y}) = \alpha f_i(\mathbf{x}) + \beta f_i(\mathbf{y}),\ i = 0,\ldots,m, \tag{1.2}$$

for all $\mathbf{x}, \mathbf{y} \in E$ and $\forall \alpha, \beta \in \mathbf{R}$.

When the optimization problem is not linear, the problem is referred to as a *nonlinear program*.

Global optimization is the process of finding the true global optimal solution. This process begins by choosing initial starting solutions. A *global optimizer* is a *solution method* which can find $\mathbf{x}^*$ regardless of the initial starting point $\mathbf{x}_0 \in E$. A solution method is an algorithm that finds the optimal solution (to some given accuracy) of a class of optimization problems.

Optimization is an active research topic in many areas, including engineering, business, the social sciences and mathematics. With the advent of new optimization algorithms, solution to various classes of optimization problems are gaining popularity [7]. Depending on particular forms of the objective function, constraints and decision variables, optimization problems can take various forms, with the following examples:

- Combinatorial optimization: where an objective function is defined over a finite set of solutions.
- Box-Bounded optimization: where an objective function is defined over lower and upper bounded design variables. The optimization problems addressed in this thesis belong to this class of optimization problems.

In a broad sense, search algorithms may be classified as *classical search methods* and *metaheuristics*. Partly in response to the limitations of classical search methods, metaheuristics are gaining increasing attention. The next section outlines key limitations of classical search methods. Before this, a brief introduction to them is provided.

1.1 Classical search methods

Classical search methods may be classified into *Gradient based algorithms* [11] and *Direct search algorithms* [36].

1.1.1 Gradient based algorithms

Gradient based optimization methods are of use when the objective function at hand is continuous and differentiable. These methods often locate an optimal solution by employing differential calculus.

In the 12^{th} century Sharaf al-Din al-Tusi, in an attempt to find a root of some single-dimensional function, developed an early form of Newton's procedure [45]. Following Newton's iterative procedure, and starting from a reasonable guess, the root is guaranteed to be found. This root finding method can be transformed to find either a local optimum or the saddle point of a function. Gradient based algorithms assume the availability derivatives. Newton's method requires the objective function to be twice differentiable, and uses first and second derivative information to construct a successive quadratic approximation of the objective function. It is thus known as a *second-order* model. Newton's procedure is perhaps *the* classic form of numerical optimization. The Secant method, a well-known extension of Newton's procedure, does not need the derivatives to be evaluated directly; rather, they are approximated.

Quasi-Newton methods generalize the Secant method to multi-dimensional problems where the inverse Hessian matrix of second derivatives is approximated. The Quasi-Newton methods not only require the existence of the gradient, they are also complex to implement. A well-known instance of Quasi-Newton methods independently developed by Broyden [8], Fletcher [18], Goldfarb [25], and Shanno [43], is known as the BFGS (Broyden Fletcher Goldfarb Shanno) method.

Steepest decent which uses the first-order Taylor polynomial, assumes the availability of the first derivatives to construct a local linear approximation of an objective function and is a *first-order* method.

1.1.2 Direct search algorithms

While Newton's method provides more than a decent direction, and has a quadratic convergence (compared to linear convergence of Steepest decent), its performance is hampered by the fact that the calculation of the Hessian matrix is required. That holds even when the complex and expensive task of Hessian matrix calculation is alleviated by approximations or variations. The main common practical difficulty, assuming a reasonable computing time for obtaining the Hessian matrix, arises when the Hessian matrix is singular, ill-conditioned, or is not positive definite. When the gradient of an optimization problem is not available, e.g., due to a partially discontinuous or non-differentiable objective function, then direct search methods are promising alternatives.

Direct search methods do not require derivative information, nor do they construct approximations of the objective function. They are thus also known as *zero-order* methods [7]. They are reasonably straightforward to understand and implement. Direct search methods rely on sampling of the objective function. While samples of the objective function may replace the actual gradient with an estimate of the gradient, precisely what it is that distinguishes them from gradient based algorithms is the sufficiency of the relative rank of solutions, rather than the actual values. Hooke and Jeeves [30] provide the following description of direct search in their 1961 paper:

> "We use the phrase 'direct search' to describe sequential examination of

> trial solutions involving comparison of each trial solution with the 'best' obtained up to that time together with a strategy for determining (as a function of earlier results) what the next trial solution will be. The phrase implies our preference, based on experience, for straightforward search strategies which employ no techniques of classical analysis except where there is a demonstrable advantage in doing so."

1.1.3 Limitations of classical search methods

When dealing with an optimization problem, several challenges arise. The problem at hand may have several local optimal solutions, it may be discontinuous, the optimal solution may appear to change when evaluated at different times, and the search space may have constraints. The problem may have a large "hilly" search space, making it intractable to try all candidate solutions in turn. The *curse of dimensionality* [3, 27], a notion coined by Richard Bellman, is another obstacle when the dimensions of the optimization problem are large.

Both gradient and direct search methods are generally regarded as local search methods [15, 26]. Nonlinear and complex dependencies that often exist among design the variables in real-world optimization problems contribute to the high number of local optimal solutions. Classical methods cannot escape from these local optimal solutions.

Another common difficulty is that they cannot be efficiently parallelized on multi-processor machines. This is especially important when measuring the fitness of candidate solutions is computationally expensive [12, 13].

Many real-world optimization problems [9] have mixed discrete and continuous design variables. A common approach to the optimization of this kind of problems, when using classic optimization algorithms, is to treat all variables as continuous, locate the optimal solution, and round off the discrete variables to their closest discrete values. The first problem with this approach is a considerable deterioration of the objective function. The second is the inefficiency of the search due to the evaluation of infeasible solutions. These difficulties may be avoided during the execution of the optimization process by taking into account the type of design variables.

Even if classical approaches offer quick convergence to an optimal solution when applied to a certain class of optimization problems, they may still not inefficient when applied to a specific optimization problem. They are mostly tailored to the salient characteristics of certain types of problems, e.g., they require a "high degree of interconnection between the solver and the objective function" [22]. A notable example is the geometric programming [6] method specifically designed to solve a posynomial-type objective function and constraints. The conjugate gradient method is suitable for strictly convex quadratic objective functions with finite and global convergence property, but it is not expected to work appropriately on multimodal optimization problems. While numerous nonlinear conjugate gradient methods for non-quadratic problems have been developed and extensively researched, they are frequently subject to severely restrictive assumptions (e.g., their convergence depends on specific properties of the optimization problem, such as Lipschitz continuity of the gradient of the objective function). Even when designing an algorithm, in some cases, efficiency is sacrificed in favor of appealing theoretical properties.

1.2 Metaheuristics

Classical search methods do not live up to the expectations of modern, computationally expensive optimization problems of today. The shortcomings (Section1.1.3) of classical search methods discussed above are partially addressed and remediated by metaheuristics. We will follow the convention of Glover [20, 21] and use the term metaheuristics to refer to all modern nature-inspired optimization algorithms. These are a class of iterative search algorithms that aim to find reasonably good solutions to optimization problems by combining different concepts for balancing *exploration* (also known as *diversification*, that is, the ability to explore the search space for new possibilities) and *exploitation* (also known as *intensification*, that is, the ability to find better solutions in the neighborhood of good solutions found so far) of the search process [39].

General applicability and effectiveness are particular advantages of metaheuristics. An appropriate balance between intensively exploiting areas with high quality solutions (the neighborhood of elite solutions) and moving to unexplored areas when necessary, is the driving force behind the high performance of metaheuristics [5]. Metaheuristics require a large number of function evaluations. They are often characterized as population-based stochastic search routines which assures a high probability of escape from local optimal solutions when compared to gradient-based and direct search algorithms. Metaheuristics do not necessarily require a good initial guess of optimal solutions, in contrast to both gradient and direct search methods, where an initial guess is highly important for convergence towards the optimal solution [14, 17]. Metaheuristics are also easy to hybridize [22], a property that makes it possible for them to exploit problem-specific heuristics.

Nature is the most complex system that has field tested solutions to many problems [22, 31, 46]. Imitation of natural processes has had a profound influence on solvers for challenging optimization problems, and has been transformed into a mature subfield existing somewhere in the intersection of computer science, physics and biology. There are a growing number of examples where nature-inspired algorithms have been successfully applied to practical problems.

Metaheuristics can be classified into *single-solution* search algorithms and *population-based* search algorithms [35]. Single-solution solvers are solution-to-solution search methods in which a single solution is evolved following a certain set of principles. Notable examples of single-solution solvers are simulated annealing and tabu search. Population-based search algorithms, such as genetic algorithms and particle swarm optimization, on the other hand, evolve a set of solutions in each iteration and generate new solutions by somehow combining multiple solutions. While the available nature-inspired metaheuristics share similarities in their search processes, their performance may differ considerably.

The most used metaheuristic in the literature is concerns *evolutionary algorithms*. A population of sample potential solutions provide information about the objective function. A new population of potential solutions is generated (stochastically, in the main) by selection and manipulation of those samples in the hope of approaching the optimal solution. One of the earliest metaheuristics concerns genetic algorithms (GAs) [23, 24, 29], a well-known approach that is based on the idea of natural selection. GAs have a very different working principle than most of the classical optimization problems. In GAs, a population of solutions evolve through a series of operators, including selection, crossover, and mutation. The section operator assures survival of the fittest solutions of the population. Crossover is an

operator that combines more than one fit solution, while mutation modifies each solution. These operators specify the neighborhood of a solution to be evaluated. While genetic algorithms may work with variables themselves, or a coding of variables, their most decisive characteristic is the selection operator.

The theory of evolution provides a sound explanation for numerous natural phenomena. The development of resistance in HIV to anti-retroviral drugs is only one instance of the laws of selection and mutation in evolution. Motivated by principles of biological evolution, genetic algorithms are search procedures that require minimal problem information. Attempts to implement some search strategies inspired by natural evolution were made by Fogel et. al [19]. More sophisticated algorithms, with some comparison of their property of convergence, are explored later [2, 42]. A systematic theoretical analysis of genetic algorithms is presented in [23, 24, 29].

Apart from GAs, this thesis will study variants of two popular optimization methods, namely *particle swarm optimization* (PSO) and *gravitational search algorithms* (GSA). PSO was originally proposed by Kennedy and Eberhart [34] as a model for the social behavior of individuals within a swarm. Each particle traverses the search space under the influence of its own best experience and that of its topological neighbors. GSAs [41] are among those population-based optimization algorithms that have been introduced recently, and is gaining popularity. It uses the concept of formation of complex structure in the universe. In GSA, the movement of each particle follows Newtons law of gravitation.

1.2.1 Convergence of metaheuristics

While mathematical proof for the convergence of global optimization algorithms can be appealing, such proofs are often of no use without practical value. The proofs available are often made in the form of infinity-limits [40], where an optimizer, provided enough iterations, is proven to find a small region surrounding the optimum. See for example [4]. When the exact same proof can be provided for random sampling search algorithms, proofs relying on infinity-limits are of no practical use.

Optimization algorithms, as stated by the *No Free Lunch* (NFL) set of theorems [47], will expose equal performance over all possible cost functions. This implies that no algorithm can be designed so as to maintain superiority over linear enumeration of the search space, or even a random sampling search algorithm. However, the NFL theorem does not hold for all subsets of the set of all possible cost functions. This implies that when developing an optimization algorithm, they can be tailored to the salient problem-specific characteristics to solve an optimization problem efficiently. Bearing in mind the *No Free Lunch* theorem, when developing metaheuristics, they need to be tested empirically.

1.3 Research goals

In this section we introduce the general context of our research by providing an overview of the challenges of metaheuristics. We also define the research goals and sub goals. Boundaries for our research are also explained.

Metaheuristics may suffer from a slow rate of convergence towards the global optimum, which implies that they may be too (computationally) expensive for certain problems.

Consequently, it is a challenge to develop computationally efficient evolution-based search methods. The aim of this research is to find ways or improve the currently exiting solutions to improve the performance of this type of search methods. This has lead us to our main research goal:

Central goal is to improve the performance of some metaheuristics by alleviating certain identified drawbacks.

This required us to look at the different situations where metaheuristics exhibit a slow convergence. We have identified that there are at least two main reasons responsible for their slow convergence: a) The large computation time required for calculating the fitness function, and b) High-dimensional search space with complex fitness landscape.

To alleviate the convergence time of computationally expensive optimization problems, a variety of techniques have been proposed in the literature. Perhaps the most obvious choice is to use parallelization techniques [1]. However, another alternative is to rely on *fitness approximation* techniques, which avoid evaluating every individual in the population of solutions (see [32, 33]). Based on an approximate model of the fitness landscape these approaches estimate the quality of some individuals. When using fitness approximation techniques, it is necessary to strike a balance between exact fitness evaluation and approximate fitness evaluation.

Lack of sufficient training data is the main problem when using most fitness approximation models currently available, hence the failure to obtain a model with sufficient approximation accuracy. Since evaluation of the original fitness function is time consuming and/or expensive, the approximate model may be of low fidelity and may even introduce false optima. Furthermore, if the training data does not cover the search domain, large errors may occur due to extrapolation. Errors may also occur when the set of training points is not sufficiently dense and uniform.

In multi-objective optimization problems (MOOP), the complexity of the problem is normally higher compared to that of single-objective optimization problems (SOOP) [10]. In general, although the fitness approximation approaches used in SOOP may be extended and adapted for MOOP, such adaptation may require more elaborate mechanisms.

Metaheuristics, by making a tradeoff between exploration and exploitation, are strategies used to guide the search process iteratively. When studying the properties of these algorithms, it turns out that some population-based optimization techniques suffer from a notable and specific search bias [37]. They tend to perform best when the optimum is located at, or near the center of the search space. This is known as center-seeking bias (CSB). General purpose optimizers are those which make no assumptions about the problem at stake. Consequently, if we want to compare the quality of the solutions found by a set of metaheuristics for a series of benchmark problems with the optimal solution near the center of the search space, the comparison becomes unfair. Metaheuristics may also suffer from bias towards the initialization region. This is known as initialization search bias (IRB). Observe that, while search algorithms may perform better when they are initialized within the whole search space, and benefit from knowing the search space, one with a lower bias towards the initialization region is preferable to one with a higher bias.

This led us to to following sub goals, which together with the main goal are addressed at the end of Chapters 2 to 6:

1. Reduction of computational complexity related to
   (a) slow convergence and/or
   (b) high computation costs of fitness evaluations
2. More effective search strategies
   (a) by improved balancing of exploration and exploitation and
   (b) measuring certain search biases (e.g. CSB and IRB)

In our research we focused on well-known and widely used metaheuristics including GAs, PSO and GSA. It was quite possible for us to include other global search algorithms, but we did not specially research them. Although the ideas presented here are applicable for other population-based search algorithms, we did not wish to continue the ongoing debate on which algorithm is superior to others in terms of convergence. We also did not study the CSB and IRB of a large class of global optimization algorithms, but we have presented a framework that enables such a study.

1.4 Research Approach

Addressing the challenges of metaheuristics raised in this thesis requires a thorough understanding of artificial intelligence in general and computational intelligence and approximate reasoning in particular. The research approach chosen in this thesis is presented below.

1.4.1 Research philosophy

The content of this thesis has been inspired by the philosophical school of *positivism* [11]. According to this philosophy, scientific knowledge must be based on logical inference from a set of objective, observable and measurable facts. The data-collection process and the findings that come from empirical evidence have to be repeatable. Positivists are reductionists, in that they break a problem down to its constituent parts, a common practice in complex systems analysis and design.

Research strategy

Among the two distinct paradigms that characterize much of the research in information systems, namely *behavioral science* and *design science* [28], the latter is the research strategy followed in this thesis. Design science is outlined in seven guidelines [28].

The contribution of a design science research to the user community is a purposeful and innovative artifact that delivers utility in terms of solving a relevant problem. In this thesis, a number of algorithms is proposed to address some areas of concerns discussed in Section1.3 (guideline 1). While the artifacts are of importance in both the current reality and practice of real-world optimization problems (guideline 2), their effectiveness is shown using relevant and well-established test problems and real-world optimization problems (guideline 3) which provide verifiable contributions to the studied research area (guideline 4). The construction and evaluation of each artifact designed relies on appropriate performance metrics

(guideline 5). The artifacts are improved during the evaluation process, where design alternatives are tested so to satisfy the research problem and objectives (guideline 6). Various artifacts discussed throughout this thesis are a result of these design alternatives. It is generally acknowledged that the results of design science should be properly communicated to the relevant audience as a measure to strengthen cumulative knowledge, as well as to motivate future work (guideline 7). Chapter 7 in general, and each other chapter in particular, discusses and envisions the impact of this research on the field.

1.4.2 Research instruments

The research in each chapter of this thesis involves four types of research tool [44]: (i) literature review, (ii) experiment (iii) evaluation and (iv) case study. *Literature review* provides the background knowledge required to conduct research and motivates the research question. *Experiment* is the tool to test the functionality of the proposed solutions in addressing the challenges identified. The empirical evidence that this thesis provides comes from controlled experiments performed in simulation environments. *Evaluation* follows each of the performed experiments. For some of the cases, the applicability of the solutions introduced is studied using real-world *case studies*.

1.5 Contributions

While the existing methods aim to reduce computational cost by approximating the fitness function, the prevalent problem of interpolation in rough surfaces remains. If the assumption of smooth continuity is invalid, interpolation might even yield values that are not physically realizable. Furthermore, in using interpolation, we may be blind to optimal solutions, as interpolation assumes a pattern of behavior that may not be valid around optimal peaks.

This thesis addresses this problem by introducing the concept of *information granulation*. With a view to reducing computational cost, the concept of fuzzy granulation is deployed to effectively approximate the fitness function. The advantages of this approach over others are that no training samples are required, and that the approximate model is updated dynamically with negligible overhead cost.

Some evolutionary computing techniques have advantages over others in terms of ease of implementation, preservation of diversity of the population, efficiency, etc. [16]. For advancement of their performance they may be simplified, hybridized etc. There has also been a steady increase in the number of global optimization algorithms, each characterized by its unique population dynamics. Different population dynamics characterize the way two conflicting goals, exploration (diversification) and exploitation (intensification), are balanced. In practice, metaheuristic algorithms have been shown to often find local minima, sometimes of low quality, meaning that the chosen balance between exploration and exploitation is not adequate to the problem at stake. We aim at presenting a solver that, next to exploitation, applies robust exploration in order to escape from local minima.

A review of the literature reveals the lack of an appropriate quantification metric for measuring CSB and IRB. Quantitative measures are succinct and are the preferred disclosure form not only for a) comparison of the degree of bias of a set of search algorithms, but are also desirable when the task is to b) discover whether a single search algorithm has

any search bias at all. In this thesis, two metrics are introduced, one for measuring center-seeking bias (CSB) and one for initialization region bias (IRB). An alternative for *center offset* [38], a common approach to analyzing center-seeking behavior of algorithms, is also proposed, as we noticed that its assumption did not always hold.

1.6 Dissertation Outline

The reminder of this thesis is organized as follows.

Chapter 2 starts with an introduction to the structural design optimization problems as an example of computationally expensive optimization problems. This is followed by an extensive review of the existing fitness approximation approaches. This is followed in turn by a proposition for an adaptive predictive model for fitness approximation, with the goal of deciding on the use of expensive function evaluations. Empirical analysis of the performance of the proposed algorithm, when applied to a set of four structural design problems, is then presented.

Chapter 3 presents the development of an auto-tuning strategy with the aim of avoiding the tuning of the parameters of the algorithm introduced in Chapter 2. Empirical analysis of the behavior of the proposed predictive model, applied to two sets of problems, then follows. The first of these is a set of several numerical benchmark problems with various optimization characteristics. The second is the real-world problem of the detection of the hidden information in a spread spectrum watermarked signal.

An extension of the developed fitness approximator to multiobjective problems is presented in Chapter 4. The proposed extension is then applied to a set of synthetic benchmark functions. These synthetic benchmarks facilitate specific aspects of the proposed extension to be tested.

Chapter 5 presents a new metaheuristic based on the theory of Big Bounce. The algorithm has been tested by comparing its performance with the performance of five different variations of other metaheuristics.

In Chapter 6, two metrics are introduced, one for measuring center-seeking bias (CSB) and one for initialization region bias (IRB). The metrics introduced are used to evaluate the bias of three algorithms while running on a test-bed of optimization problems which have their optimal solution at, or near to, the center of the search space. The most prominent finding is considerable CSB and IRB of gravitational search algorithm(GSA). In addition, a partial solution to the center-seeking and initialization region bias of GSA is proposed. The performance of the proposed variant of GSA which promotes the global searching capability of GSA is verified using a number of synthetic benchmarks problems. The solvers studied are used to identify the parameters of a gene regulatory network system.

Chapter 7 provides a summary of the main findings of this thesis and presents future research directions.

The appendices present a glossary of the terms and a list of publications derived from this work.

Bibliography

[1] Alba, E. and Tomassini, M. (2002). Parallelism and Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462.

[2] Bagley, J. (1967). *The behavior of adaptive systems which employ genetic and correlation algorithms*. PhD thesis, PhD thesis, University of Michigan, Ann Arbor.

[3] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, N.J.

[4] Birbil, Ş., Fang, S., and Sheu, R. (2004). On the convergence of a population-based global optimization algorithm. *Journal of global optimization*, 30(2):301–318.

[5] Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308.

[6] Boyd, S., Kim, S., den berghe, L. V., and Hassibi, A. (2007). A tutorial on geometric programming. *Optimization and Engineering*, 8(1):67–127.

[7] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

[8] Broyden, C. (1970). The convergence of a class of double-rank minimization algorithms. *IMA Journal of Applied Mathematics*, 6(1):76–90.

[9] Chiong, R., Weise, T., and Michalewicz, Z. (2011). *Variants of evolutionary algorithms for real-world applications*. Springer.

[10] Coello Coello, C. A., Lamont, G. B., and Van Veldhuizen, D. A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, second edition. ISBN 978-0-387-33254-3.

[11] Creswell, J. W. (2013). *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage Publications, fourth edition.

[12] Davarynejad, M., Akbarzadeh-T, M.-R., and Pariz, N. (2007). A novel general framework for evolutionary optimization: Adaptive fuzzy fitness granulation. In *IEEE Congress on Evolutionary Computation (CEC'07)*, pages 951–956. IEEE.

[13] Davarynejad, M., Vrancken, J., van den Berg, J., and Coello Coello, C. (2012). A Fitness Granulation Approach for Large-Scale Structural Design Optimization. In Chiong, R., Weise, T., and Michalewicz, Z., editors, *Variants of Evolutionary Algorithms for Real-World Applications*, pages 245–280. Springer-Verlag, Berlin.

[14] Deb, K. (1999). An introduction to genetic algorithms. In *Sadhana (Academy Proceedings in Engineering Sciences)*, volume 24, pages 293–315.

[15] Deb, K. and Goyal, M. (1997). Optimizing engineering designs using a combined genetic search. In *Proceedings of the seventh international conference on genetic algorithms*, pages 521–528.

[16] del Valle, Y., Venayagamoorthy, G., Mohagheghi, S., Hernandez, J., and Harley, R. (2008). Particle swarm optimization: basic concepts, variants and applications in power systems. *IEEE Transactions on Evolutionary Computation*, 12(2):171–195.

[17] Dorsey, R. and Mayer, W. (1995). Genetic algorithms for estimation problems with multiple optima, nondifferentiability, and other irregular features. *Journal of Business & Economic Statistics*, 13(1):53–66.

[18] Fletcher, R. (1970). A new approach to variable metric algorithms. *The Computer Journal*, 13(3):317–322.

[19] Fogel, L., Owens, A., and Walsh, M. (1966). Artificial intelligence through simulated evolution.

[20] Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549.

[21] Glover, F. and Kochenberger, G. (2003). *Handbook of metaheuristics*. Springer.

[22] Goldberg, D. (1994). Genetic and evolutionary algorithms come of age. *Communications of the ACM*, 37(3):113–119.

[23] Goldberg, D. (2002). *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers.

[24] Goldberg, D. and Holland, J. (1988). Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99.

[25] Goldfarb, D. (1970). A family of variable metric methods derived by variational means. *Mathematics of computation*, 24(109):23–26.

[26] Haddad, O., Afshar, A., and Marino, M. (2006). Honey-bees mating optimization (hbmo) algorithm: a new heuristic approach for water resources optimization. *Water Resources Management*, 20(5):661–680.

[27] Hastie, T., Tibshirani, R., and Friedman, J. (2008). *The elements of statistical learning*. Springer-Verlag, 2nd edition.

[28] Hevner, A., March, S., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 28(1):75–105.

[29] Holland, J. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI.

[30] Hooke, R. and Jeeves, T. (1961). "direct search" solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2):212–229.

[31] Huebsch, N. and Mooney, D. (2009). Inspiration and application in the evolution of biomaterials. *Nature*, 462(7272):426–432.

[32] Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 9(1):3–12.

[33] Jin, Y. (2011). Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1:61–70.

[34] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948.

[35] Kochenberger, G. (2003). *Handbook of metaheuristics*. Springer.

[36] Kolda, T., Lewis, R., and Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. *SIAM review*, 45(3):385–482.

[37] Mitchell, T. (1997). *Machine learning*. McGraw Hill.

[38] Monson, C. and Seppi, K. (2005). Exposing origin-seeking bias in pso. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 241–248.

[39] Osman, I. and Laporte, G. (1996). Metaheuristics: A bibliography. *Annals of Operations Research*, 63(5):511–623.

[40] Pedersen, M. (2010). *Tuning & simplifying heuristical optimization*. PhD thesis, PhD thesis, University of Southampton.

[41] Rashedi, E., Nezamabadi-Pour, H., and Saryazdi, S. (2009). Gsa: a gravitational search algorithm. *Information Sciences*, 179(13):2232–2248.

[42] Rosenberg, R. (1967). *Simulation of genetic populations with biochemical properties*. PhD thesis, PhD thesis, University of Michigan.

[43] Shanno, D. (1970). Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, 24(111):647–656.

[44] Sjoberg, D., Dyba, T., and Jorgensen, M. (2007). The future of empirical methods in software engineering research. In *Future of Software Engineering, 2007*, pages 358–378.

[45] Soleymani, F. and Sharifi, M. (2011). On a general efficient class of four-step root-finding methods. *International Journal of Mathematics and Computers in Simulation*, 5:181–189.

[46] Wen, H., Zhang, S., Hapeshi, K., and Wang, X. (2008). An innovative methodology of product design from nature. *Journal of Bionic Engineering*, 5(1):75–84.

[47] Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

"Complexity makes discovery of the optimum a long, perhaps never-to-be completed task, so the best among tested options must be exploited at every step."

John Holland - 1992

2

A Fitness Granulation Approach for Large-Scale Structural Design Optimization [1]

Abstract

The complexity of large-scale mechanical optimization problems is partially due to the presence of high-dimensional design variables, the interdependence among design variables, and the high computational cost of the finite element simulations needed to evaluate the fitness of candidate solutions. Evolutionary cycles are ruled by competitive games of survival and not merely by absolute measures of fitness, as well as exploiting the robustness of evolution against uncertainties in the fitness function evaluations. This chapter takes up the complexity challenge of mechanical optimization problems by proposing a new fitness granulation approach that attempts to cope with many difficulties of fitness approximation approaches that have been reported in the literature. The approach is based on adaptive fuzzy fitness granulation having as its main aim to strike a balance between the accuracy and the utility of the computations. The adaptation algorithm adjusts the radials of influence of granules according to the perceived performance and level of convergence attained. Experimental results show that the proposed approach accelerates the convergence towards optimal solutions, when compared to the performance of other more popular approaches.

[1] This chapter is based on:

- M. Davarynejad, J. Vrancken, J. van den Berg, and C. A. Coello Coello, "A Fitness Granulation Approach for Large-Scale Mechanical Optimization Problems", In Raymond Chiong and Zbigniew Michalewicz (Eds.), *Variants of Evolutionary Algorithms for Real-World Applications*, pp. 245-280, Springer, Berlin, 2012. (ISBN 978-3-642-23423-1)

This suggests its applicability to other complex finite element-based engineering design problems.

2.1 Introduction

Since the 1960s, and due to significant developments in numerical methods and computing, the finite element analysis (FEA) became a frequent tool to solve engineering problems that arise in systems with several interacting components, complex geometries, and which are under the effect of different physical phenomena. These systems elude exact techniques, but are reasonably manageable by means of a systematic discretization approach known as the finite element method (FEM) [47]. At the same time that the FEM was developed, efficient and fast optimization algorithms have arisen for solving various kinds of mathematical and optimization problems (OPs). Both trends contributed to the development of large-scale structural design and optimization problems (SDOPs) and to the discipline of structural optimization. The aim of structural optimization is to generate automated procedures for finding the best possible structure with respect to at least one criterion (the objective), and having to satisfy a set of constraints, by selecting from a set of geometrical dimensions, material properties and/or topological parameters [43].

Structural optimization problems are often challenging due to their high computational demands [2], multi-modality, non-convexity, high dimensionality, and multi-objectivity. Because of this, many structural optimization problems are weakly amenable to conventional mathematical programming approaches, which motivates the use of alternative solution methods.

Randomized search heuristics are among the simplest and most robust strategies that are applicable to a wide range of optimization problems including structural design (SD). While they can normally provide nearly optimal solutions, they cannot guarantee convergence to the optimum. However, their computational requirements are normally high. Among the randomized search heuristics currently available, evolutionary algorithms (EAs) have become very popular in the last few years, mainly because of their ease of use and efficacy. EAs are stochastic search techniques which operate on a set of solutions (the so-called population), that are modified based on the principles of the natural evolution (i.e., the survival of the fittest) [39]. EAs have been commonly adopted for solving complex SD problems. For example, Walker and Smith [61] combined the FEM and EAs to minimize a weighted sum of the mass and deflection of fiber-reinforced structures. Similarly, Abe et al. [1] used FEM and an EA for structural optimization of the belt construction of a tire. More recently, Giger and Ermanni [21] applied FEM and EA to minimize the mass of composite fiber-reinforced plastic (CFRP) rims subject to strength and stiffness constraints. However, EAs may suffer from a slow rate of convergence towards the global optimum, which implies that they may be too (computationally) expensive for certain SD problems. Consequently, it is challenging to develop computationally efficient evolution-based search methods.

To alleviate the problem of converging time of computationally expensive optimization problems, a variety of techniques has been proposed in the literature. Perhaps the most ob-

[2] Finite element analysis is computationally costly and may require several days to complete its calculations, even for a relatively simple problem.

vious choice is to use parallelization techniques [4]. However, another alternative is to rely on fitness approximation techniques, which avoid evaluating every individual in the population of an EA. In order to do this, these approaches estimate the quality of some individuals, based on an approximate model of the fitness landscape. This is the sort of approach on which this chapter is focused. Section 2.4 provides a review of fitness approximation techniques in evolutionary computation. When using fitness approximation techniques, it is necessary to strike a balance between exact fitness evaluation and approximate fitness evaluation. In this chapter, with a view to reducing computational cost, we employ the concept of fuzzy granulation to effectively approximate the fitness function. The advantages of this approach over others is the fact that no training samples are required, and the approximate model is dynamically updated with no or negligible overhead cost.

The remainder of this chapter is organized as follows. The following section elaborates upon four SD optimization problems before explaining the genetic algorithm (GA) approach proposed here for the SD optimization task (see Section 2.3). This is followed by a review of a variety of fitness approximation approaches that have been proposed for EAs in Section 2.4. In order to accelerate the convergence speed of the GA with a minimum number of fitness function evaluations, a novel method is presented in Section 4.4. The approach is based on generating fuzzy granules via an adaptive similarity analysis. To illustrate the efficiency of the proposed method in solving the four SD problems introduced in Section 2.2, the performance results of different optimization algorithms are presented in Section 2.6. A further statistical analysis confirms that the proposed approach reduces the computational complexity of the number of fitness function evaluations by over 50% while reaching similar or even better final fitness values. Finally, in Section 3.5 we provide our conclusions.

2.2 Structural design optimization problems

Four SD optimization problems, with increasing complexity are investigated here. They are the following: (1) the design of a *3-layer composite beam* with *two* decision variables, (2) the design of an *airplane wing* with *six* decision variables, (3) the design of a *2D truss* frame with *36* decision variables, and (4) the voltage/pattern design of piezoelectric actuators. We discuss in more detail the last problem, because of its complexity. Such a problem consists of finding the best voltage and pattern arrangement for static shape control of a piezoelectric actuator with *200* design variables. Clearly, this is a more challenging and heavy optimization task from a fitness/computational perspective.

2.2.1 Easier/Smaller problems

The first three SD problems are covered in this section. The ultimate goal in these optimization problems is to maximize the first natural frequency [3] of the given structure. To allow more space for the last problem (described in Subsections 2.2.2 and 2.6.4), we limit ourselves here to a short description of the other problems.

[3] Resonance occurs when the excitation frequency is the same as the natural frequency. For the same excitation energy, the resulting vibration levels at resonance frequency is higher than other exciting frequencies. The importance of maximizing the first natural frequency is to avoid the resonance phenomenon to occur.

3-Layer composite beam

A *multi-layered composite beam* is constructed from a combination of two or more layers of dissimilar materials that are joined together to act as a unit in which the resulting combination is lighter, stronger and safer than the sum of its parts. A finite element analysis model has been developed to analyze the multi-layer composite beams and plates. The objective is to raise the first natural frequency of the beam.

Airplane wing

An *airplane wing* is an elastic structure that, in the presence of aerodynamic loads, starts to vibrate. In this study, we treated the natural frequency as the design objective since it is quite intuitive and natural to raise the natural frequencies of the wing so that it is not easily excited by undesirable disturbances.

2D truss frame

Trusses are the most commonly used structure and in comparison to heavily-built structures, they have a relatively small *dead weight*. A truss consists of *bar-elements* (members) connected by *hinged joints* to each other and supported at the base. Truss design problems belong to the class of load-supporting structure design problems that are usually finite-dimension optimization problems. The design of load-supporting structures plays a key role in engineering dynamics. The objective (fitness) here is to raise the structure's *first natural frequency*.

2.2.2 Voltage and pattern design of a piezoelectric actuator

Piezoelectric materials exhibit both direct (electric field generation as a response to mechanical strains) and converse (mechanical strain is produced as a result of an electric field) piezoelectric effects. The direct effect is used in piezoelectric sensors while the converse effect is used in piezoelectric actuators.

Apart from ultrasound applications, energy harvesting, sensor applications (e.g., strain gauges and pressure sensors), and vibration/noise control domains, piezoelectric materials are widely used as actuators in smart structures. Smart structures with integrated self-monitoring, self-diagnosis and control capabilities have practical uses ranging from MEMS, biomedical engineering, control engineering, aerospace structures, ground transportation systems and marine applications. The smart structures' technology is widely used in biomechanics, i.e., to expand obstructed blood vessels or to prevent further enlargement of blood vessels damaged by aneurysms [37] which most commonly occurs in arteries. Another apparent practical use of smart and adaptive structural systems is to properly control the undesirable motions of geometry-changing structures.

Piezoelectric actuators are also found in an enormous range of applications for distributed actuation and control of mechanical structures for shape correction and modification. One example for this is their use in flexible aircrafts where they improve the aerodynamic performance and deformation control of conformal antennas [26], through their incorporation within the structure. For instance, in [34], an optimization algorithm is used to deal with the shape control of functionally graded material (FGM) plates which are actively

controlled by piezoelectric sensor and actuator patches. A computational intelligence-based algorithm is used to derive the optimal voltage distribution, by adopting the elements of the gain control matrix as the design variables.

The optimal shape control and correction of small displacements in composite structures using piezoelectric actuators concern complex engineering problems. To achieve a predefined shape of the structure of the metal plate, in this chapter we will present a fast converging global optimization algorithm to find the optimal actuation voltages that need to be applied to the piezoelectric actuators and to the pattern of piezoelectric patches.

2.3 GAs in structural optimization problems

Genetic algorithms (GAs) are perhaps the most popular type of EAs nowadays and have been applied to a wide variety of problems [22]. The GA optimization procedure for solving SD problems begins with a set of randomly selected parents (design variables). If any of these parents does not meet all the physical constraints, they are modified until they do. In subsequent generations, each offspring's phenotype is also checked for its feasibility. Furthermore, the fitness values of the parents and their offspring are compared and the worst individuals are rejected, preserving the remaining ones as parents of the new generation (known as steady-state population treatment). This procedure is repeated until a given termination criterion is satisfied.

Due to their robustness, GAs have been frequently used in a variety of real world optimization applications including optimizing the placement of actuators on large space structures [20], the design of a low-budget lightweight motorcycle frame with superior dynamic and mechanical properties [51], and the evolution of the structural configuration for weight minimization of a space truss structure [32]. The implementation of a GA can be summarized as follows:

1. **Initialization**: Initialize P design variable $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_i, \ldots, \mathbf{x}_P\}$, where P is the population size.
2. **Constraints check**: If satisfied, continue, else modify $\mathbf{x}_i$ until the candidate solution becomes feasible.
3. **Evaluation (Analysis)**: Evaluate the fitness function $f(\mathbf{x}_i)$, $i = \{1, 2, \ldots, P\}$.
4. **Convergence check**:
   (a) **if satisfied** stop,
   (b) **else** select the next generation parent design variable, apply genetic operators (mutation, recombination) and generate the next offspring design variables $\mathbf{x}$. Go to step 2.

EAs in general are often expensive in the sense that they may require a high number of computationally costly objective function evaluations. As a result, it may be necessary to forgo an exact evaluation and use approximated fitness values that are computationally efficient. In the design of mechanical structures, for instance, each exact fitness evaluation

requires the time-consuming stage of FEA which, depending on the size of the problem, may consume from several seconds up to several days. If we assume a conventional genetic algorithm with a fixed and modest population size of 100, a maximum of 100 generations, and a very small-scale structural problem that requires 10 seconds for each fitness evaluation, the total execution of the GA would require 30 hours! This should make evident the inhibiting role of the computational complexity associated to GAs (and EAs, in general) for more non-trivial and large-scale problems.

Since one of the crucial aspects for solving large-scale SD optimization problems using EAs is the computational time required, in the following section we outline a few existing strategies that have been proposed to deal with this issue.

2.4 Fitness Approximation in Evolutionary Computation

As indicated before, one possibility to deal with time-consuming problems using a GA is to avoid evaluating every individual and estimate instead the quality of some of them based on an approximate model of the search space. Approximation techniques may estimate individuals' fitness on the basis of previously observed objective function values of *neighboring* individuals. There are many possible approximation models [24]. Next, we will briefly review some of the most commonly adopted fitness approximation methods reported in the specialized literature.

2.4.1 Fitness Inheritance

This is a very simple technique that was originally introduced by Smith et al. [60]. The mechanism works as follows: when assigning fitness to an individual, some times we evaluate the objective function as usual, but the rest of the time, we assign fitness as an average (or a weighted average) of the fitness of the parents. This fitness assignment scheme will save us one fitness function evaluation, and operates based on the assumption of similarity between an offspring and its parents. Clearly, fitness inheritance cannot be applied all the time, since we require some true fitness function values in order to obtain enough information to guide the search. This approach uses a parameter called *inheritance proportion*, which regulates how many times do we apply fitness inheritance (the rest of the time, we compute the true fitness function values). As will be seen next, several authors have reported the use of fitness inheritance.

Zheng et al. [66] used fitness inheritance for codebook design in data compression techniques. They concluded that the use of fitness inheritance did not degrade, in a significant way, the performance of their GA.

Salami et al. [55] proposed a Fast Evolutionary Strategy (FES) in which a fitness and associated reliability value were assigned to each new individual. Considering two decision variables $p_1^i = (\mathbf{x}_1^i, f_1^i, r_1^i)$ and $p_2^i = (\mathbf{x}_2^i, f_2^i, r_2^i)$ where $\mathbf{x}_1^i$ and $\mathbf{x}_2^i$ are the chromosomes 1 and 2 at generation i with fitness values f_1^i and f_2^i and reliabilities r_1^i and r_2^i, respectively. In this scheme, the true fitness function is only evaluated if the reliability value is below a certain threshold. Otherwise, the fitness of the new individual and its reliability value is calculated from:

$$f^{i+1} = \frac{S_1 r_1^i f_1^i + S_2 r_2^i f_2^i}{S_1 r_1^i + S_2 r_2^i} \tag{2.1}$$

and

$$r^{i+1} = \frac{(S_1 r_1^i)^2 + (S_2 r_2^i)^2}{S_1 r_1^i + S_2 r_2^i} \tag{2.2}$$

where S_1 is the similarity between $\mathbf{x}_1^{i+1}$ and $\mathbf{x}_1^i$ and S_2 is the similarity between $\mathbf{x}_1^{i+1}$ and $\mathbf{x}_2^i$. Also, they incorporated random evaluation and error compensation strategies. Clearly, this is another (more elaborate) form of fitness inheritance. Salami et al. reported an average reduction of 40% in the number of evaluations while obtaining similar solutions. In the same work, they presented an application of (traditional) fitness inheritance to image compression obtaining reductions ranging from 35% up to 42% of the total number of fitness function evaluations.

Pelikan et al. [44] used fitness inheritance to estimate the fitness for only part of the solutions in the Bayesian Optimization Algorithm (BOA). They concluded that fitness inheritance is a promising concept, because population-sizing requirements for building appropriate models of promising solutions lead to good fitness estimates, even if only a small proportion of candidate solutions is evaluated using the true fitness function.

Fitness inheritance has also been used for dealing with multi-objective optimization problems. Reyes-Sierra and Coello Coello [49, 50] incorporated the concept of fitness inheritance into a multi-objective particle swarm optimizer and validated it in several test problems of different degrees of difficulty. They generally reported lower computational costs, while the quality of their results improved in higher dimensional spaces. This was in contradiction with other studies (e.g., [17] as well as this chapter) that indicate that the performance of the parents may not be a good predictor for their children's fitness in sufficiently complex problems, rendering fitness inheritance inappropriate under such circumstances.

2.4.2 Surrogates

A common approach to deal with expensive objective functions is to construct an approximation function which is much cheaper to evaluate (computationally speaking). In order to build such an approximation function which will be used to predict promising new solutions, several sample points are required. The meta-model built under this scheme aims to reduce the total number of (true objective function) evaluations performed, while producing results of a reasonably good quality.

Evidently, the accuracy of the surrogate model depends on the number of samples provided (and their distribution) and on the approximation model adopted. Since throughout the course of optimization the model will be used very frequently, it is very important that the construction of such a model is computationally efficient [24]. The following are examples of the use of surrogates of different types.

Sano et al. [56] proposed a genetic algorithm for optimization of continuous noisy fitness functions. In this approach, they utilized the history of the search to reduce the number of fitness function evaluations. The fitness of a novel individual is estimated using the fitness values of the other individuals as well as the sampled fitness values for it. So, as to increase the number of individuals adopted for evaluation, they not only used the current generation

but also the whole history of the search. To utilize the history of the search, a stochastic model of the fitness function is introduced, and the maximum likelihood technique is used for estimation of the fitness function. They concluded that the proposed method outperforms a conventional GA in noisy environments.

Branke et al. [9] suggested the use of local regression for estimation, taking the fitness of neighboring individuals into account. Since in local regression it is very important to determine which solutions belong to the neighborhood of a given individual, they studied two different approaches for setting the value of the size of the local neighborhood (relative neighborhood and adaptive neighborhood). They concluded that local regression provides better estimations than previously proposed approaches. In more recent work [8], a comparison between two estimation methods, interpolation and regression, is done. They concluded that regression seems to be slightly preferable, particularly if only a very small fraction of the individuals in the population is evaluated. Their experiments also show that using fitness estimation, it is possible to either reach a better fitness level in a given time, or to reach a desired fitness level much faster (using roughly half of the original number of fitness function evaluations).

Ong et al. [41] proposed a local surrogate modeling algorithm for parallel evolutionary optimization of computationally expensive problems. The proposed algorithm combines hybrid evolutionary optimization techniques, radial basis functions, and trust-region frameworks. The main idea of the proposed approach is to use an EA combined with a feasible sequential quadratic programming solver. Each individual within an EA generation is used as an initial solution for local search, based on Lamarckian learning. They employed a trust-region framework to manage the interaction between the original objective and constraint functions and the computationally cheap surrogate models (which consist of radial basis networks constructed by using data points in the neighborhood of the initial solution), during local search. Extensive numerical studies are presented for some benchmark test functions and an aerodynamic wing design problem. They show that the proposed framework provides good designs on a limited computational budget. In more recent work, Ong et al. [42] presented a study on the effects of uncertainty in the surrogate model, using what they call Surrogate-Assisted Evolutionary Algorithms (SAEA). In particular, the focus was on both the *curse of uncertainty* (impairments due to errors in the approximation) and *blessing of uncertainty* (benefits of approximation errors). Several algorithms are tested, namelly the Surrogated-Assisted Memetic Algorithm (SAMA) proposed in [41], a standard genetic algorithm, a memetic algorithm (considered as the standard hybridization of a genetic algorithm and the feasible sequential quadratic programming solver used in [41]), and the SAMA-Perfect algorithm (which is the SAMA algorithm but using the exact fitness function as surrogate model), on three multi-modal benchmark problems (Ackley, Griewank and Rastrigin). The conclusion was that approximation errors lead to convergence at false global optima, but turns out to be beneficial in some cases, accelerating the evolutionary search.

Regis and Shoemakes [48] developed an approach for the optimization of continuous costly functions that uses a space-filling experimental design and local function approximation to reduce the number of function evaluations in an evolutionary algorithm. The proposed approach estimates the objective function value of an offspring by means of a function approximation model over the k-nearest previously evaluated points. The estimated values are used to identify the most promising offspring per function evaluation. A Symmetric Latin Hypercube Design (SLHD) is used to determine initial points for function

evaluation, and for the construction of the function approximation models. They compared the performance of an Evolution Strategy (ES) with local quadratic approximation, an ES with local cubic radial basis function interpolation, an ES whose initial parent population is obtained from a SLHD, and a conventional ES (in all cases, they used a (μ,λ)-ES with uncorrelated mutations). The algorithms were tested on a groundwater bioremediation problem and on some benchmark test functions for global optimization (including *Dixon-Szegö*, *Rastrigin* and *Ackley*). The obtained results (which include analysis of variance to provide stronger and solid claims regarding the relative performance of the algorithms) suggest that the approach that uses SLHDs together with local function approximations has potential for success in enhancing EAs for computationally expensive real-world problems. Also, the cubic radial basis function approach appears to be more promising than the quadratic approximation approach on difficult higher-dimensional problems.

Lim et al. [35] presented a Trusted Evolutionary Algorithm (TEA) for solving optimization problems with computationally expensive fitness functions. TEA is designed to maintain good trustworthiness of the surrogate models in predicting fitness improvements or controlling approximation errors throughout the evolutionary search. In this case, the most interesting part was to predict search improvement as opposed to the quality of the approximation, which is regarded as a secondary objective. TEA begins its search using the canonical EA, with only exact function evaluations. During the canonical EA search, the exact fitness values obtained are archived in a central database together with the design variables (to be used later for constructing surrogate models). After some initial search generations (specified by the user), the trust region approach takes place beginning from the best solution provided by the canonical EA. The trust region approach uses a surrogate model (radial basis neural networks) and contracts or expands the trust radius depending on the ability of the approximation model in predicting fitness improvements, until the termination conditions are reached. An empirical study was performed on two highly multimodal benchmark functions commonly used in the global optimization literature (*Ackley* and *Griewank*). Numerical comparisons to the canonical EA and the original trust region line search framework are also reported. From the obtained results, the conclusion was that TEA converges to near-optimum solutions more efficiently than the canonical evolutionary algorithm.

Kriging

A more elaborate surrogate model that has been relatively popular in engineering, is the so-called Gaussian Process Model, also known as Kriging [54]. This approach builds probability models through sample data and estimates the function values at every untested point with a Gaussian distribution.

Ratle [46] presented a new approach based on a classical real-encoded genetic algorithm for accelerating the convergence of evolutionary optimization methods. A reduction in the number of fitness function calls was obtained by means of an approximate model of the fitness landscape using Kriging interpolation. The author built a statistical model from a small number of data points obtained during one or more generations of the evolutionary method using the true fitness landscape. The model is updated each time a convergence criterion is reached.

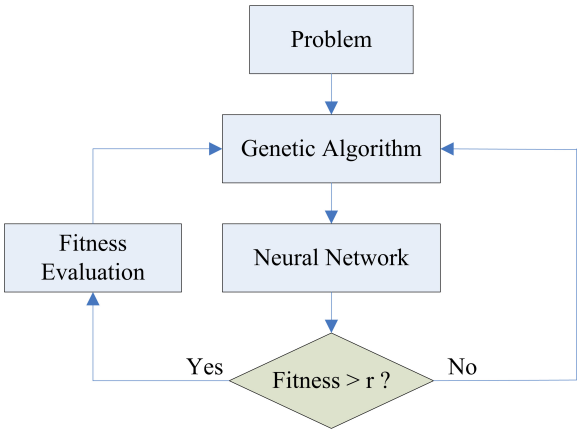

Figure 2.1: The GA-ANN algorithm that is proposed in [28]. Only if the approximate fitness of an individual is better than the maximum fitness found in the last population, its fitness is re-evaluated in order to find its real fitness value.

2.4.3 Artificial Neural Networks

In the last few years, artificial neural networks (ANNs), including multi-layer perceptrons [23] and radial basis function networks [64] have also been employed to build approximate models for design optimization. Due to their universal approximation properties, ANNs can be good fitness function estimators if provided with sufficient structural complexity and richness in their training data set. Next, some representative applications of the use of ANNs for building approximate models will be briefly reviewed.

Khorsand et al. [28] investigated structural design by a hybrid of neural network and finite element analysis. They used the neuro-estimation of the fitness value only when the individual was not deemed to be highly fit (error in estimation may not be important). The methodology used in [28] is presented in Figure 2.1 where r is considered as the maximum fitness of the individuals in the last generation. As with any other numerically driven approximation method, the performance of ANNs is closely related to the quality of the training data.

Jin et al. [25] investigated the convergence properties of an evolution strategy with neural network-based fitness evaluations. In this work, the concept of *controlled evolution* is introduced, in which the evolution proceeds using not only the approximate fitness function value, but also the true fitness function value. They also introduce two possibilities to combine the true with the approximate fitness function value: (1) the controlled individuals approach and (2) the controlled generation approach. Jin et al. defined "controlled" as evaluated with the true fitness function. Both approaches were studied and some interesting conclusions/recommendations for the correct use of such techniques are provided. A comprehensive survey of fitness approximation applied in evolutionary algorithms is presented in [58].

2.4.4 Final Remarks About Fitness Approximation

Lack of sufficient training data is the main problem in using most of the fitness approximation models currently available and hence the failure to reach a model with sufficient approximation accuracy. Since evaluation of the original fitness function is very time consuming and/or expensive, the approximate model may be of low fidelity and may even introduce false optima. Furthermore, if the training data does not cover all the domain range, large errors may occur due to extrapolation. Errors may also occur when the set of training points is not sufficiently dense and uniform. In such situations, a combination of methods may be more desirable. For example, Ong et al. [41] combined radial basis functions with transductive inference to generate local surrogate models.

Alternatively, if individuals in a population can be clustered into several groups as in [30], then only the individual that represents its cluster can be evaluated. The fitness value of other individuals in the same cluster will be estimated from the representative individual based on a distance measure. This is termed fitness imitation in contrast to fitness inheritance [24]. The idea of fitness imitation has been extended and more sophisticated estimation methods have been developed in [7]. A similarity based model is introduced in [18] and is applied to constrained and unconstrained optimization problems.

In multi-objective optimization problems (MOOP), the complexity of the problem is normally higher, compared to that of single-objective optimization problems (SOOP) [11]. In general, although the fitness approximation approaches used in SOOP can be simply extended and adapted for MOOP, such adaptation may require more elaborate mechanisms. One example of this is constraint-handling.[4] It is well-known that in real-world optimization problems there are normally constraints of different types (e.g., related to the geometry of structural elements, to completion times, etc.) that must be satisfied for a solution to be acceptable. Traditionally, penalty functions have been used with EAs to handle constraints in SOOP [10]. However, because of the several problems associated to penalty functions (e.g., the definition of appropriate penalty values is normally a difficult task that has a serious impact on the performance of the EA), some researchers have proposed alternative constraint-handling approaches that require less critical parameters and perform well across a variety of problems (see for example [10, 38, 53]). However, when dealing with MOOPs, many of these constraint-handling techniques cannot be used in a straightforward manner, since in this case, the best trade-offs among the objectives are always located in the boundary between the feasible and the feasible region. This requires the development of different approaches specially tailored for MOOPs (see for example [59, 63]). A similar problem occurs when attempting to migrate single-objective fitness approximation models to MOOPs. For more details on this topic, see [57].

While the above methods aim to reduce computational cost by approximating the fitness function, the prevalent problems with interpolation in rough surfaces remains. If the assumption of smooth continuity is not valid, interpolation may even yield values that are not physically realizable. Furthermore, we may be blinded to the optimal solutions using interpolation as interpolation assumes a pattern of behavior that may not be valid around optimal peaks. The next section addresses this problem by introducing the concept of infor-

[4] Although constraint-handling techniques are very important in real-world optimization problems, their discussion is beyond the scope of this chapter, due to space limitations. Interested readers are referred to other references for more information on this topic (see for example [38, 52]).

mation granulation.

2.5 Adaptive Fuzzy Fitness Granulation

Fuzzy granulation of information is a vehicle for handling information, as well as a lack of it (uncertainty), at a level of coarseness that can solve problems appropriately and efficiently [13]. In 1979, the concept of fuzzy information granulation was proposed by Zadeh [65] as a technique by which a class of points (objects) are partitioned into granules, with a granule being a clump of objects drawn together by indistinguishability, similarity, or functionality. The fuzziness of granules and their attributes is characteristic of the ways by which human concepts and reasoning are formed, organized and manipulated. The concept of a granule is more general than that of a cluster, potentially giving rise to several conceptual structures in various fields of science as well as mathematics.

In this chapter, with a view to reducing computational cost, the concept of fitness granulation is applied to exploit the natural tolerance of EAs in fitness function computations. Nature's *survival of the fittest* is not about exact measures of fitness; rather it is about rankings among competing peers. By exploiting this natural tolerance for imprecision, optimization performance can be preserved by computing fitness only selectively and only to keep this ranking among individuals in a given population. Also, fitness is not interpolated or estimated; rather, the similarity and indistinguishability among real solutions is exploited.

In the proposed algorithm, an adaptive pool of solutions (fuzzy granules) with an exactly computed fitness function is maintained. If a new individual is sufficiently similar to a known fuzzy granule [65], then that granules' fitness is used instead as a crude estimate. Otherwise, that individual is added to the pool as a new fuzzy granule. In this fashion, regardless of the competitions' outcome, the fitness of the new individual is always a physically realizable one, even if it is a *crude* estimate and not an exact measurement. The pool size as well as each granules' radius of influence is adaptive and will grow/shrink depending on the utility of each granule and the overall population fitness. To encourage fewer function evaluations, each granule's radius of influence is initially large and gradually shrinks at later stages of the evolutionary process. This encourages more exact fitness evaluations when competition is fierce among more similar and converging solutions. Furthermore, to prevent the pool from growing too large, not used granules are gradually replaced by new granules, once the pool reaches a certain maturity.

2.5.1 Algorithm Structure

Given the general overview in the preceding section, the concrete steps of the algorithm are as follows:

Step 1: Create a random parent population $P^1 = \{\mathbf{x}_1^1, \mathbf{x}_2^1, \ldots, \mathbf{x}_j^1, \ldots, \mathbf{x}_t^1\}$ of design variable, where, more generally, $\mathbf{x}_j^i = \{x_{j,1}^i, x_{j,2}^i, \ldots, x_{j,r}^i, \ldots, x_{j,m}^i\}$ is the jth parameter individual in the ith generation, $x_{j,r}^i \in \mathbf{R}$ the rth parameter of $\mathbf{x}_j^i$, m is the number of design variables and t is the population size.

Step 2: Define a multi-set G of fuzzy granules (C_k, σ_k, L_k) according to $G = \{(C_k, \sigma_k, L_k)|C_k \in \Re^m, \sigma_k \in \Re, L_k \in \Re, k = 1,\ldots,l\}$. G is initially empty (i.e., $l = 0$). C_k is an m-dimensional vector of centers, σ_k is the width of membership function (WMF) of the kth

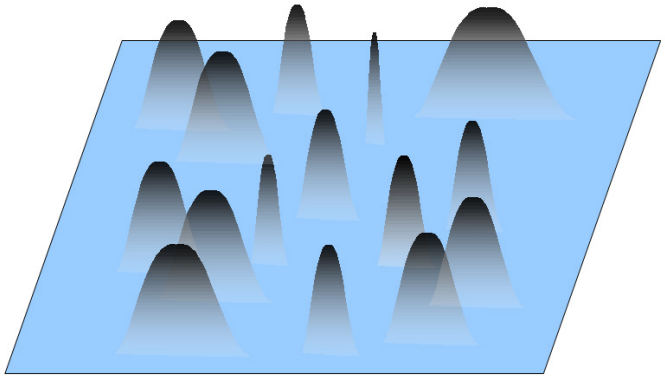

Figure 2.2: A number of gaussian granules with different widths in a 2-D solution space. Once a new individual is sufficiently similar to a granule in the granule pool, then that granules' fitness is used instead as a crude estimate. Otherwise, that individual is added to the pool as a new fuzzy granule. Each granules' radius of influence is determined based on equation (2.4).

fuzzy granule, and L_k is the granule's life index. A number of granules with different widths are shown in Figure 2.2.

Step 3: Choose the phenotype of first chromosome ($\mathbf{x}_1^1 = \{x_{1,1}^1, x_{1,2}^1, \ldots, x_{1,r}^1, \ldots, x_{1,m}^1\}$) as the center of the first granule ($C_1 = \{c_{1,1}, c_{1,2}, \ldots, c_{1,r}, \ldots, c_{1,m}\} = \mathbf{x}_1^1$).

Step 4: Define the membership $\mu_{k,r}$ of each $x_{j,r}^i$ to each granule member by a Gaussian similarity neighborhood function according to

$$\mu_{k,r}\left(x_{j,r}^i\right) = \exp\left(\frac{-\left(x_{j,r}^i - c_{k,r}\right)^2}{\left(\sigma_k\right)^2}\right),\ k = 1,2,\ldots,l\ , \tag{2.3}$$

where l is the number of fuzzy granules.

Remark: σ_k is the distance measurement parameter that controls the degree of similarity between two individuals. Like in [14], σ_k is defined based on equation (2.4). According to this definition, the granules shrink or enlarge in reverse proportion to their fitness:

$$\sigma_k = \gamma \frac{1}{\left(e^{F(C_k)}\right)^{\beta}}\ , \tag{2.4}$$

where $\beta > 0$ is an emphasis operator and γ is a proportionality constant. The problem arising here is how to determine the parameters β and γ as design parameters. The fact is that these two parameters are problem dependent and, in practice, a number of trials is needed to adjust these parameters. This trial is based on a simple rule with respect to the acceleration of the parameter optimization procedure: high speed needs to have enlargement in the granule spread and, as a consequence of this, less accuracy is obtained in the fitness approximation, and viceversa. To deal with this rule, a fuzzy controller is presented in [14].

Step 5: Compute the average similarity of every new design parameter $\mathbf{x}_j^i = \{x_{j,1}^i, x_{j,2}^i, \ldots, x_{j,r}^i, \ldots, x_{j,m}^i\}$ to each granule G_k using equation (2.5)

$$\overline{\mu}_{j,k} = \frac{\sum_{r=1}^{m} \mu_{k,r}\left(x_{j,r}^{i}\right)}{m} \tag{2.5}$$

Step 6: Either calculate the exact fitness function of $\mathbf{x}_j^i$ or estimate the fitness function value by associating it to one of the granules in the pool in case there is a granule in the pool with higher similarity to $\mathbf{x}_j^i$ than a predefined threshold, i.e.

$$f\left(\mathbf{x}_j^i\right) = \begin{cases} f\left(C_k\right) & \text{if } \max\limits_{k\in\{1,2,\ldots,l\}} \{\overline{\mu}_{j,k}\} > \theta^i \,, \\ f\left(\mathbf{x}_j^i\right) & \text{otherwise.} \end{cases} \tag{2.6}$$

where $f(C_x)$ is the fitness function value of the fuzzy granule, $f(\mathbf{x}_j^i)$ is the real fitness calculation of the individual, $\theta^i = \alpha(\max\{f(\mathbf{x}_1^{i-1}), f(\mathbf{x}_2^{i-1}), \ldots, f(\mathbf{x}_t^{i-1})\}/\overline{f}^{i-1})$, $K = \text{argmax}\{\overline{\mu}_{j,k}\}$ when $k \in \{1,2,\ldots,l\}$, $\overline{f}^i = \sum_{j=1}^{i} f(\mathbf{x}_j^i)/t$ and $\alpha > 0$ is a proportionality constant that is usually set at 0.9 unless otherwise indicated. The threshold θ^i increases as the best individual's fitness at generation i increases. As the population matures and reaches higher fitness values (i.e., while converging more), the algorithm becomes more selective and uses exact fitness calculations more often. Therefore, with this technique we can utilize the previous computational efforts during previous generations. Alternatively, if

$$\max_{k\in\{1,2,\ldots,l\}} \{\overline{\mu}_{j,k}\} < \theta^i$$

$\mathbf{x}_j^i$ is chosen as a newly created granule.

Step 7: If the population size is not completed, repeat **Steps 5 to 7**.

Step 8: Select parents using a suitable selection operator and apply the genetic operators of recombination and mutation to create a new generation.

Step 9: When termination/evolution control criteria are not met, then update σ_k using equation (2.4) and repeat **Steps 5 to 9**.

In [13] and [3], additional details on the convergence speed of the algorithm on a series of mathematical testbeds are provided along with a simple example to illustrate the competitive granule pool update.

2.5.2 How to control the length of the granule pool?

As the evolutionary procedures are applied, it is inevitable that new granules are generated and added to the pool. Depending on the complexity of the problem, the size of this pool can be excessive and become a computational burden itself. To prevent such unnecessary computational effort, a *forgetting factor* is introduced in order to appropriately decrease the size of the pool. In other words, it is better to remove granules that do not win new individuals, thereby producing a bias against individuals that have low fitness and were likely produced by a failed mutation attempt. Hence, L_k is initially set to N and subsequently updated as below,

$$L_k = \begin{cases} L_k + M & \text{if } k = K \,, \\ L_k & otherwise \,, \end{cases} \tag{2.7}$$

where M is the life reward of the granule and K is the index of the winning granule for each individual at generation i. At each table update, only the N_G granules with the highest L_k index are kept, and the others are discarded. In [16], an example has been provided to illustrate the competitive granule pool update law. Adding a new granule to the granule pool and assigning a life index to it, is a simple way of controlling the size of the granule pool, since the granules with the lowest life index will be removed from the pool. However, it may happen that the new granule is removed, even though it was just inserted into the pool. In order to prevent this, the pool is split into two parts with sizes εN_G and $(1-\varepsilon)N_G$. The first part is a FIFO (First In, First Out) queue and new granules are added to this part. If it grows above εN_G, then the top of the queue is moved to the other part. Removal from the pool takes place only in the $(1-\varepsilon)N_G$ part. In this way, new granules have a good chance to survive a number of steps. In all of the simulations that are conducted here, ε is set at 0.1.

The distance measurement parameter is completely influenced by the granule enlargement/shrinkage in the widths of the produced membership functions. As in [16], the combined effect of granule enlargement/shrinkage is in accordance with the granule fitness and it requires the fine-tuning of two parameters, namely β and γ. These parameters are problem dependent and it seems critical to have a procedure to deal with this difficulty. In [14] and [15], an auto-tuning strategy for determining the width of membership functions is presented which removes the need of exact parameter determination, without a negative influence on the convergence speed.

2.6 Numerical results

To illustrate the efficacy of the proposed granulation algorithm, the result of applying it to the problems introduced in Section 2.2 are studied and analyzed in the two following sections. The commercial FEA software ANSYS [5] is used during the analysis and numerical simulation study.

The GA routines utilize initially random populations, binary-coded chromosomes, single-point crossover for the first three problems and 15-point crossover for the piezoelectric actuator design problem, mutation, fitness scaling, and an elitist stochastic universal sampling selection strategy. Crossover rate $P_{XOVER} = 1$, $P_{MUTATION} = 0.01$ and the population size is set at 20. However, due to the number of parameters and complexity of the structural problems, the number of generations is set to 50 for the first three problems and 600 for the piezoelectric actuator design problem. These settings were determined during several trial runs to reflect the best performing set of parameters for the GA. Finally, the chromosome length varies depending on the number of variables in a given problem but each variable is still allocated 8 bits.

For performing the FES, a fitness and associated reliability value are assigned to each new individual. The reliability value, T, varies between 0 and 1 and depends on two factors: first is the reliability of parents, and second is how close parents and children are in the solution space, as explained in equation (2.2). Also, as mentioned in [55], $T = 0.7$ is used for the threshold as we empirically found that it generally produces the best results. The parameters of the GA-ANN are the same as in the GA alone. In the GA-ANN approach for solving optimization problems, a two-layer neural network is used, having as input the design variables and as outputs the fitness values.

Furthermore, due to the stochastic nature of EAs, each of the simulations was repeated ten times, and a paired Mann-Whitney U test was performed except for the last optimization problem in which, for each algorithm, it was performed only once, due to the running time needed. The significance level α represents the maximum tolerable risk of incorrectly rejecting the null hypothesis H_0, indicating that the mean of the 1^{st} population is not significantly different from the mean of the 2^{nd} population. The p-value or the observed significance level of a statistical test is the smallest value of α for which H_0 can be rejected. If the p-value is less than the pre-assigned significance level α, then the null hypothesis is rejected. Here, the significance level α was assigned, and the p-value was calculated for each of the following applications.

The results are presented in Tables 2.1, 2.2, 2.3 and 2.5, in which *FFE* stands for the number of fitness function evaluations needed to perform the optimization task and the *training data* column presents the number of initial input/output pairs needed in order to build up the approximation model. Since the most computationally expensive part of an evolutionary algorithm is usually, by far, its fitness evaluation, the convergence *time improvement* of different algorithms, compared to the standard GA, can be estimated in terms of the number of fitness evaluations. So, the *time improvement* percentage column is calculated as one minus the difference between the sum of *FFE* and *training data* divided by the number of FFE of the standard algorithm, i.e., a GA, multiplied by 100.

2.6.1 3-Layer composite beam

A 3-layer composite beam has been modeled numerically by using the ANSYS program. The composite layout are the design variables that change in the region [0 - 180]. The objective here is to raise the first natural frequency by appropriately choosing 2 composite layers' angles. In this example, the Young's modulus [19] is EX = 210 GPa, EY = 25 GPa, EZ = 25 GPa, GXY = GYZ = GXZ = 30 GPa, Poisson's ratio ν = 0.2 and density ρ = 2100 kg/m^3. There are two design variables (two degrees of freedom) for this optimization problem each varying between 0 and 180. For this case, a 2-100-1 ANN architecture is consequently chosen and used for the optimization runs. The proposed algorithm (called AFFG, for adaptive fuzzy fitness granulation) and other methods are compared in Table 2.1. Results indicate that while there is not a significant statistical difference between the three algorithms in terms of solution fitness, i.e., rigidity of the beam, the time savings provided by the proposed method is much higher than that of the GA-ANN. In particular, the proposed AFFG algorithm finds better solutions on the average with less computational time as compared with the GA-ANN. Also, while FES seems to have found better solutions, the proposed GA-AFFG used less than half as many evaluations.

2.6.2 Airplane wing

Figure 2.3(a) shows the initial design of an airplane wing. The wing is of uniform configuration along its length, and its cross-sectional area is defined to be a straight line and a spline. It is held fixed to the body of the airplane at one end and hangs up freely at the other. The objective here is to maximize the wing's *first natural frequency* by appropriately choosing three key points of the spline. The material properties are: Young's modulus = 261.820 GPa, density ρ = 11031 kg/m^3, Poisson's ratio ν = 0.3.

Table 2.1: Performance of the optimization methods (average of 10 runs) for the 3-layer composite beam, $\alpha = 0.9$, $\beta = 0.1$, $\gamma = 30$, $M = 5$, $N_G = 250$, $T = 0.7$.

	FFEs	Training data	Time improvement (%)	Optimum	*p*-value
GA	1000	Not Needed		19.3722	
FES	228.1	Not Needed	77.19	19.369	0.0211
GA-ANN	155.9	100	74.41	19.3551	0.0026
GA-AFFG	97.5	Not Needed	90.25	19.3681	0.0355

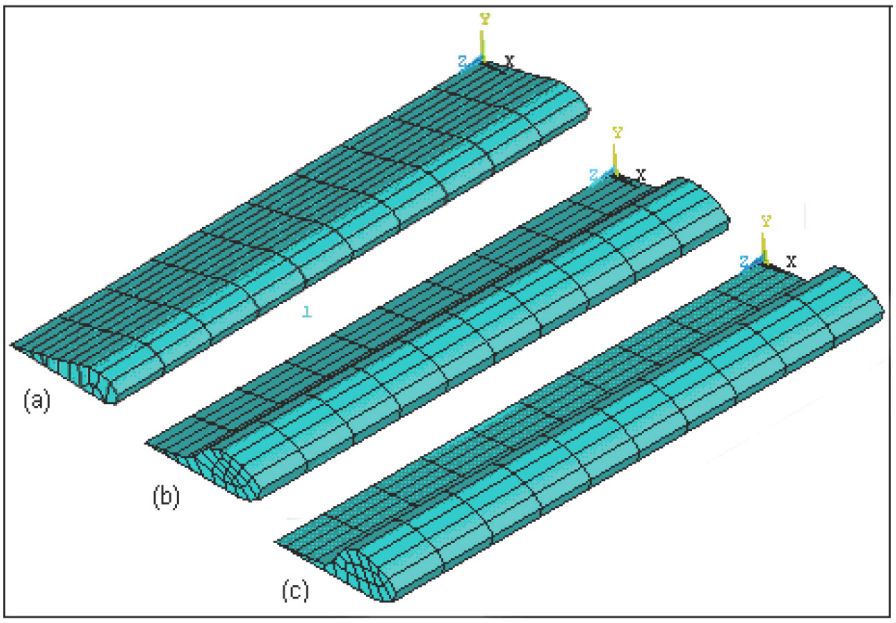

Figure 2.3: Airplane wing: (a) initial shape, (b) GA optimized shape, and (c) GA-AFFG.

The optimized shape found by a simple GA is shown in Figure 2.3(b) and that found by GA-AFFG is shown in Figure 2.3(c). A 6-100-1 architecture is chosen for the ANN used as fitness approximator. Table 2.2 illustrates that while the GA-ANN finds inferior solutions as compared with the GA, the use of the ANN significantly reduces computational time. The application of AFFG shows an improvement in the search quality while maintaining a low computational cost. Specifically, the average 10-run performance of the AFFG solutions is higher than that of any of the competing algorithms including the GA, FES and GA-ANN. Furthermore, while the Mann-Whitney U test confirms that the proposed algorithm solutions are at least as good as those produced by the GA, the proposed algorithm is over 82% faster.

Table 2.2: Performance of the optimization methods (average of 10 runs) for Airplane wing, $\alpha = 0.9$, $\beta = 0.5$, $\gamma = 1$, $M = 5$, $N_G = 250$, $T = 0.7$.

	FFEs	Training data	Time improvement (%)	Optimum	*p*-value
GA	1000			6.0006	
FES	481.6		51.84	5.9801	0.9698
GA-ANN	172.1	100	72.79	5.9386	0.4274
GA-AFFG	173.5		82.65	6.0527	0.3075

Table 2.3: Performance of the optimization methods (average of 10 runs) for the 2D truss, $\alpha = 0.9$, $\beta = 0.11$, $\gamma = 3.05$, $M = 5$, $N_G = 550$, $T = 0.7$.

	FFEs	Training data	Time improvement (%)	Optimum	*p*-value
GA	1000			12.1052	
FES	1000		0	11.8726	0.0058
GA-ANN	293	100	60.66	11.8697	0.0257
GA-AFFG	570.4		42.96	12.1160	0.9097

2.6.3 2D truss frame

A typical truss designed by an engineer is illustrated in Figure 2.4(a). The objective (fitness) here is to raise the structure's *first natural frequency* to reduce the vibration domain and to prevent the resonance phenomenon (in dynamic response) of the structure by appropriately choosing the 18 key point locations (our design variables) as illustrated in Figure 2.3(a).

In this benchmark, isotropic material properties are assumed (Young's modulus E = 210 GPa, Poisson's ratio $\nu = 0.3$ and density $\rho = 7800$ kg/m^3). The optimized shapes produced by the GA and the new proposed method AFFG are shown in Figures 2.4(b) and 2.4(c), respectively. The 36-100-1 ANN architecture is chosen and used for the optimization runs.

The search begins with an initial population. The maximum fitness in the initial population is nearly 9.32. Over several generations, the fitness gradually evolves to a higher value of 11.902. Figure 2.5 shows a plot of best, average and worst fitness vs. generation number for one run of our GA-AFFG. This performance curve shows the learning activity or adaptation associated with the algorithm. The total number of generations is 50. For a population size of twenty, this requires 1000 (50×20) fitness evaluations for the GA while the proposed GA-AFFG required only 570.4 fitness evaluations. Figure 2.6 shows the plot of the number of FEA evaluations vs. generation number corresponding to one run [13].

2.6.4 Voltage and pattern design of piezoelectric actuator

Piezoelectric materials have coupled mechanical and electrical properties making them able to generate a voltage when subjected to a force or deformation (this is termed as the direct piezoelectric effect). Conversely, they exhibit mechanical deformation when subjected to an applied electric field (this is called the converse piezoelectric effect) [3]. Various applications of piezoelectric actuators/sensors have appeared in the literature. Lin et al. [36] inves-

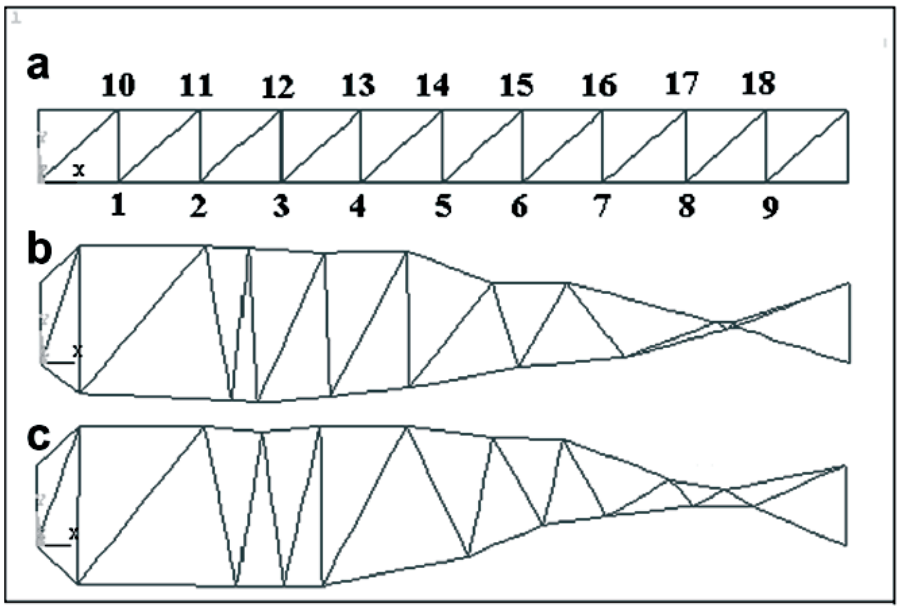

Figure 2.4: 2D truss frame: (a) initial configuration, (b) GA optimized shape, and (c) GA-AFFG optimized shape.

tigated the modeling and vibration control of a smart beam by using piezoelectric damping-modal actuators/sensors. They presented theoretical formulations based on damping-modal actuators/sensors and numerical solutions for the analysis of a laminated composite beam with integrated sensors and actuators. A proof-of-concept design of an inchworm-type piezoelectric actuator of large displacement and force for shape and vibration control of adaptive truss structures is proposed by Li et al. in [33]. The applications of such actuators include smart or adaptive structural systems for the car and aerospace industries.

A fiber composite plate with initial imperfections and under in-plane compressive loads is studied by Adali et al. [2] with a view towards minimizing its deflection and optimizing its stacking sequence by means of the piezoelectric actuators and the fiber orientations. Krommer [31] studied a method to control the deformation of a sub-section of a beam. His intention was to apply a distributed control by means of self-stresses within the sub-section to keep the sub-section in its non-deformed state. In practical applications such as deformation control of conformal antennas, this strategy is highly valuable.

Global optimization algorithms [62] along with a finite element formulation are widely used in shape control. For instance in [34], a computational intelligence based optimization algorithm along with a modified finite element formulation is used to deal with the shape control of functionally graded material (FGM) plates that contain piezoelectric sensor and actuator patches. In this study, an optimal voltage distribution or a gain control matrix are used as design variables for the shape control of smart structures. Numerical simulations have been successfully carried out on the shape control of the FGM plates by optimizing the voltage distribution for the open loop shape control or gain values for the closed loop shape control. A finite element formulation with non-rectangular shaped actuators for laminated

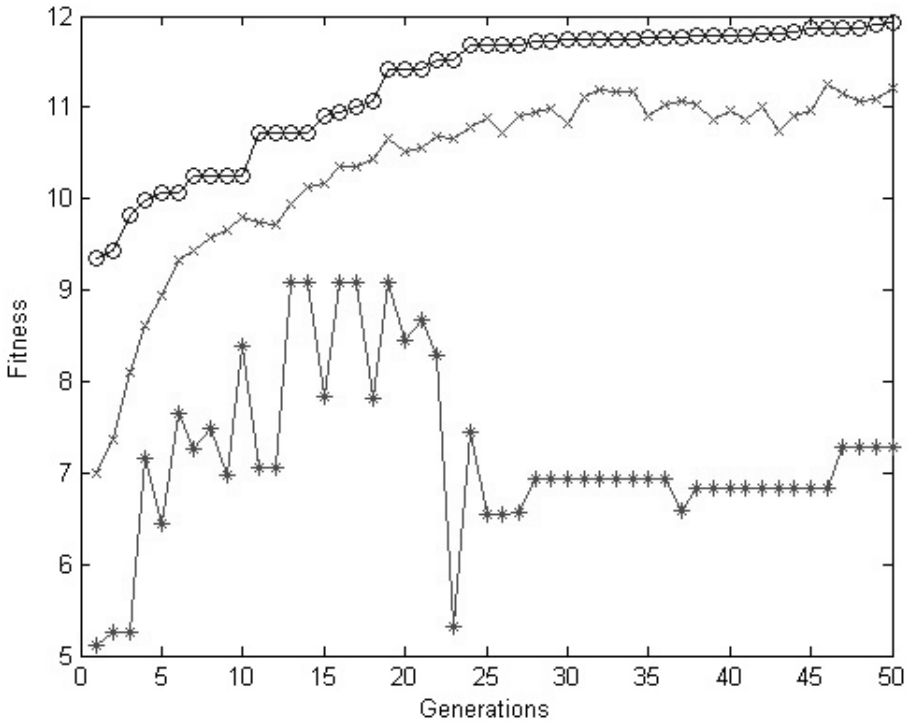

Figure 2.5: Plot of generation number vs. fitness value for the 2D truss frame using GA-AFFG: best (circle), average (cross) and worst (asterisk) individuals at each generation.

smart composite structure is studied in [40]. For smart cantilever plates, the actuated deflections are measured and are used to validate the present formulation. They also investigated the effect of actuator pattern on the optimum values of the applied voltages and the shape match factors. Numerical results shown that the actuator patterns may have an important influence on the values of the optimum voltages applied to each individual actuator and the final shape match factor.

Piezoelectric equations (constitutive equations)

In this study, the assumption is that the thermal effect is negligible. The piezoelectric constitutive relationships describe how two piezoelectric mechanical and electrical quantities (stress, strain, electric displacement, and electric field) interact and it is expressed by the direct and the converse piezoelectric equations respectively [6]:

$$\{D\} = [e]\{\varepsilon\} + [\varepsilon]\{E\}\,, \tag{2.8}$$

$$\{\sigma\} = [Q]\{\varepsilon\} + [e]^T\{E\}\,, \tag{2.9}$$

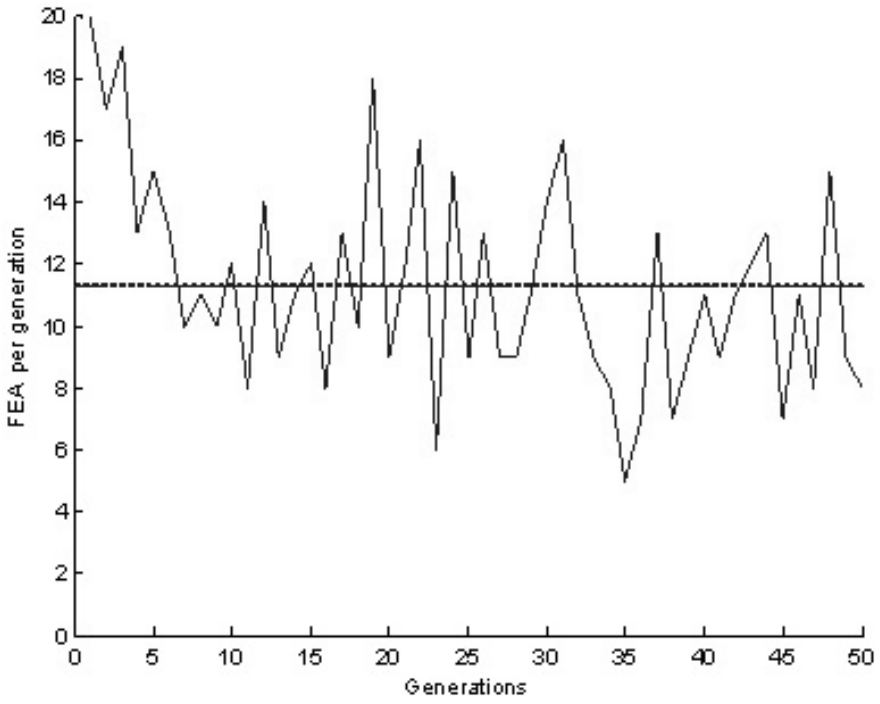

Figure 2.6: Plot of the generation number vs. number of FEA evaluations for the 2D truss frame in a single run using GA-AFFG.

where $\{\sigma\}$ is the stress vector, $[Q]$ is the elastic stiffness matrix, $\{\varepsilon\}$ is the strain vector, $[e]$ is the piezoelectric constant matrix, $\{E\} = -\nabla\varphi$ is the electric field vector. Also, φ is the electrical potential, $\{D\}$ is the electric displacement vector and $[\varepsilon]$ is the permittivity coefficient matrix. Equations (2.8) and (2.9) describe the electromechanical coupling. Assuming that a laminated beam consists of a number of layers and each layer possesses a plane of material symmetrically parallel to the x-y plane, the constitutive equations for the k^{th} layer can be written as [29]:

$$\begin{Bmatrix} D_1 \\ D_3 \end{Bmatrix}_k = \begin{bmatrix} 0 & e_{15} \\ e_{31} & 0 \end{bmatrix}_k \times \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_5 \end{Bmatrix}_k + \begin{bmatrix} \varepsilon_{11} & 0 \\ 0 & \varepsilon_{33} \end{bmatrix}_k \times \begin{Bmatrix} E_1 \\ E_3 \end{Bmatrix}_k \tag{2.10}$$

$$\begin{Bmatrix} \varepsilon_1 \\ \varepsilon_3 \end{Bmatrix}_k = \begin{bmatrix} Q_{11} & 0 \\ 0 & Q_{55} \end{bmatrix}_k \times \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_5 \end{Bmatrix}_k + \begin{bmatrix} 0 & \varepsilon_{31} \\ \varepsilon_{15} & 0 \end{bmatrix}_k \times \begin{Bmatrix} E_1 \\ E_3 \end{Bmatrix}_k \tag{2.11}$$

where

$$Q_{11} = \frac{E_{11}}{1 - v_{12}v_{21}}, Q_{55} = G_{13}$$

and are the reduced elastic constants of the k^{th} layer, E_{11} is the Young's modulus and G_{13} is the shear modulus. The piezoelectric constant matrix $[e]$ can be expressed in terms of the piezoelectric strain $[d]$ as:

$$[e] = [d][Q]$$

where

$$[d] = \begin{bmatrix} 0 & d_{15} \\ d_{31} & 0 \end{bmatrix}$$

Using the above piezoelectricity analysis and formulation, finite element model (FEM) of piezoelectric patches and metal plate [45] was built by ANSYS [5]. Also, a small deflection and thin plate theory are assumed for the FEM of the plate.

To validate the software, a clamped free aluminum plate with 4 piezoelectric patches is modeled and the results are compared with the experimental model of reference [12]. A close agreement between our model and our experimental results is observed. Also, in order to achieve an acceptable mesh density, mesh sensitivity [5] is performed.

Piezoelectric design for static shape control

The shape control problem considered here is to find the optimal actuator pattern design variable P and exiting voltage vector V as design variables. The (quasi-) static shape control problem can be defined, in the context of an optimization formulation, as follows:

Find $\mathbf{x} = [P,V]^T$ to minimize:

$$f(\mathbf{x}) = \sum_{j=1}^{N_x} \sum_{i=1}^{N_y} \frac{\left| d_{i,j}^d - d_{i,j}^f \right|}{\left| max_{i,j} \left(d_{i,j}^d \right) \right|} / \left(N_x \times N_y \right). \tag{2.12}$$

$\mathbf{x}$ is the decision variable with two components: i) the pattern variable vector P, and ii) the applied voltage variable vector V. Here, $f(\mathbf{x})$ is the objective function. P is the distribution of active piezoelectric actuator material (pattern variable) whereas the voltage variables in vector V are the electrical potentials applied across the thickness direction of each actuator. The objective function $f(\mathbf{x})$ in equation (2.12) is a weighted sum of all the absolute differences between the desired and designed shapes at all nodes. $d_{i,j}^d$ and $d_{i,j}^f$ are the desired and designed (calculated by the FE model) transversal displacements of the (i,j)-location, respectively. $max(d_{i,j}^d)$ is the maximum displacement in the desired structural shape. As the displacement is small here, there is no need to consider stress or strain constraint variables for the shape control problem.

Model description

A cantilever plate clamped at its left edge and subjected to a non-applied mechanical load is assumed here. The plate has a length of 154 mm; width of 48 mm and consists of one layer of 0.5 mm in thickness. The piezoelectric actuators (thickness of 0.3 mm each) are attached to the top surfaces of the plate (Figure 2.7). The desired pre-defined surface [12] is defined as:

$$d_{i,j}^d = \left(1.91x^2 + 0.88xy + 0.19x \right) \times 10^{-4}. \tag{2.13}$$

[5] Mesh sensitivity is performed to reduce the number of elements and nodes in the mesh while ensuring the accuracy of the finite element solution [27].

Table 2.4: Material properties for the PX5-N piezoelectric material [12].

$C_{11}^E(N\,m^{-2})$	13.11×10^{10}	$d_{15}(m\,V^{-1})$	515×10^{-12}
$C_{12}^E(N\,m^{-2})$	7.984×10^{10}	$d_{31}(m\,V^{-1})$	-215×10^{-12}
$C_{13}^E(N\,m^{-2})$	8.439×10^{10}	$d_{33}(m\,V^{-1})$	500×10^{-12}
$C_{33}^E(N\,m^{-2})$	12.31×10^{10}	$\varepsilon_{11}^t/\varepsilon_0$	1800
$C_{44}^E(N\,m^{-2})$	2.564×10^{10}	$\varepsilon_{33}^t/\varepsilon_0$	2100
$C_{66}^E(N\,m^{-2})$	2.564×10^{10}	$\rho(kg\,m^{-3})$	7800

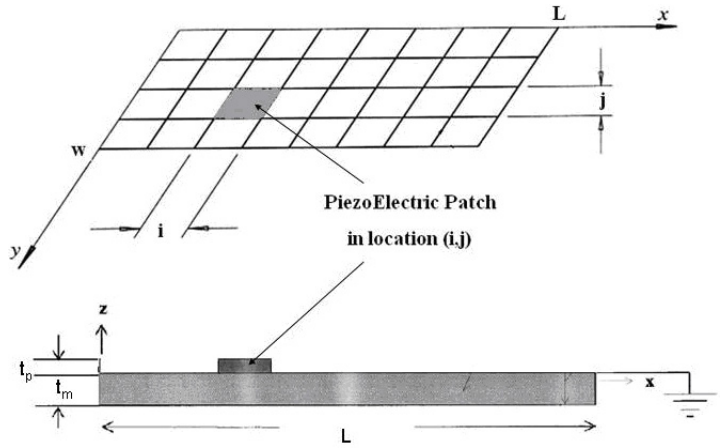

Figure 2.7: Geometrical model of the piezoelectric patch adopted here.

The piezoelectric electro-mechanical properties shown in Table 2.4 according to PX5-N from Philips Components. After a careful mesh sensitivity analysis, a FEM is built as illustrated in Figure 2.8.

For this SD and optimization problem, there are 200 design variables. Half of these

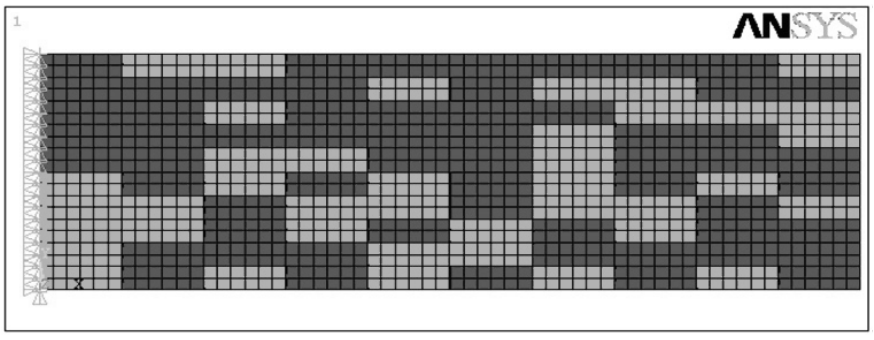

Figure 2.8: Finite element model built by ANSYS.

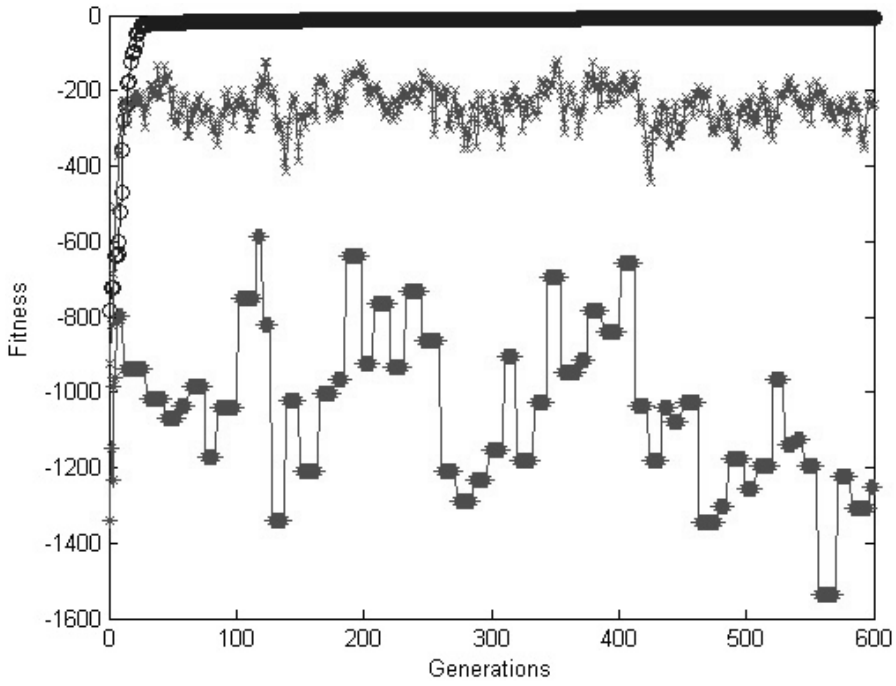

Figure 2.9: Generation number vs. fitness for the piezoelectric actuator using our proposed GA-AFFG for a single run: best (circle), average (cross) and worst (asterisk) of individuals at each generation.

Table 2.5: Piezoelectric actuator performance of the optimization methods, α *= 0.9,* β *= 0.11,* γ *= 3.05, M = 5,* N_G *= 550, T = 0.7 .*

	FFEs	Training data	Time Improved (%)	Error (%)
GA	12000			7.313
FES	12000		0	12.82
GA-ANN	2617	5000	36.52	8.093
GA-AFFG	5066	Not needed	57.64	7.141

design variables belong to actuation voltage of piezoelectric patches which vary between -10 and 20 V and the rest of the design variables are Boolean, indicating whether or not the voltage should be applied to the piezoelectric patches. When the $i^{th}(i = 1,\ldots,100)$ piezoelectric pattern variable is zero, the piezoelectric patch is not built so that there is no actuation voltage, and viceversa. Figure 2.9 shows the graph of best, average and worst fitness vs. generation number and Figure 2.10 shows the number of FEA evaluations vs. generation number for a single GA-AFFG run while Table 2.4 presents the results of the four optimization algorithms corresponding to one run.

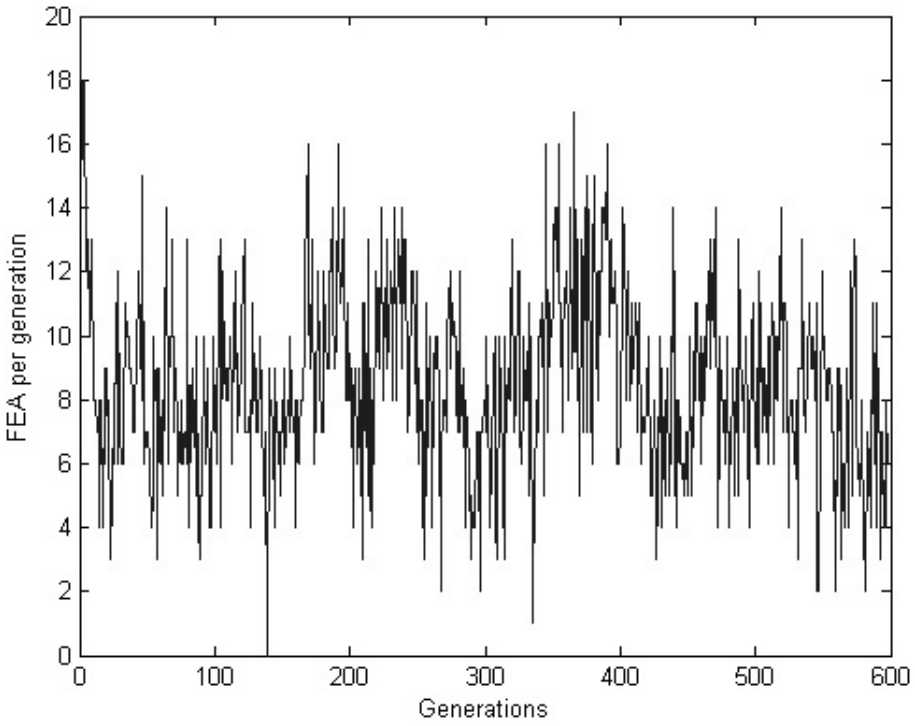

Figure 2.10: Generation number vs. number of FEA evaluations, for the piezoelectric actuator, using our proposed GA-AFFG for a single run.

2.7 Analysis of results

Tables 2.1, 2.2, 2.3 and 2.5, illustrate the performance of the proposed GA-AFFG method in comparison with a GA, FES and GA-ANN [13] in terms of computational efficiency and performance for the 3-layer composite beam, the airplane wing, and the 2D truss design problems as well as for the piezoelectric actuator problem. Due to the stochastic nature of the GA, the first three design simulations are repeated 10 times and a statistical analysis is performed. However, for the piezoelectic actuator we could not run the GA that many times, because of its high computational cost.

The second column in these tables makes a comparison of the three algorithms in terms of the number of FEA evaluations as compared to those of the GA, while the fourth column makes a comparison in terms of performance. Results indicate that our proposed GA-AFFG finds statistically equivalent solutions by using more than 90%, 82%, 42% and 57% fewer finite element evaluations. The GA-ANN also significantly reduces the number of FEA evaluations, but its average performance is inferior when compared with our proposed GA-AFFG due to the ANNs approximation error. It must be noted that the improvement in time by GA-ANN takes into account the time spent on constructing the training data set. It must be noted that the GA-ANN's improved time includes the number of initial training data.

For the piezoelectric actuator design problem, Table 2.5 illustrates a comparison of the GA, FES and GA-ANN [3] with respect to our proposed GA-AFFG in terms of computa-

Table 2.6: A Mann-Whitney U test of the number of real fitness calculations for the 3-layer composite beam (10 runs).

	Simulation results		
3-layer composite beam	Mean	Var	p-Value
FES	228.1	4601.2	6.39×10^{-05}
GA-ANN	155.9	511.9	6.34×10^{-05}
GA-AFFG	97.5	406.7	6.39×10^{-05}

Table 2.7: A Mann-Whitney U test of the number of real fitness calculations for the airplane wing (10 runs).

	Simulation results		
Airplane wing	Mean	Var	p-Value
FES	481.6	38648	6.39×10^{-05}
GA-ANN	172.1	6392.1	6.39×10^{-05}
GA-AFFG	173.5	1600.3	6.39×10^{-05}

tional efficiency and performance. The second column of this table makes a comparison of the four algorithms in terms of the number of FEA evaluations as compared with a GA, while the fifth column makes a comparison in terms of the quality of the optimal solutions. Results indicate that GA-AFFG finds at least equivalent solutions by using 57% fewer finite element evaluations as compared to GA. Also, when compared with the GA-ANN, the proposed algorithm finds better solutions while being more computationally efficient. The main difference here is ANN's need for pre-training. Trying various sizes of initial training sets and considering the 200 design parameters, the ANN required at least 5000 training data pairs for comparable performance, see Table 2.5.

Overall, when compared with a GA, the two sets of applications indicate that FES, GA-ANN and GA-AFFG improve the computational efficiency of their problem by reducing the number of exact fitness function evaluations. However, the neuro-approximation as well as fitness inheritance fail with a growing size of the input-output space. Consequently, the utility of AFFG becomes more significant in larger and more complex design problems. Furthermore, our statistical analysis confirms that fitness inheritance is more comparable in terms of performance when the size of the search space is smaller (Tables 2.1 and 2.2), but its performance deteriorates as the complexity of the problem increases (Tables 2.3 and 2.5).

A comparison of the number of exact fitness function evaluations in terms of mean and variance that presents the improved computational time is presented in Tables 2.6, 2.7 and 2.8 for the first three mechanical optimization problems described before. A Mann-Whitney U test is also performed to study the significance of lower computation cost. Since the fourth optimization problem (piezoelectic actuator design) could not be repeated due to the its FEA time consuming nature, a Mann-Whitney U test could not be performed in that case.

Table 2.8: A Mann-Whitney U test of the number of real fitness calculations for the 2D truss (10 runs).

	Simulation results		
2D truss	Mean	Var	p-Value
FES	100	0	Not available
GA-ANN	293	2394.2	6.39×10^{-05}
GA-AFFG	570.4	18477	6.39×10^{-05}

2.8 Conclusions

In this chapter, we have proposed a systematic and robust methodology for solving complex structural design and optimization problems. The proposed methodology relies on the use of finite element analysis and adaptive fuzzy fitness granulation. As we saw, adaptive fuzzy fitness granulation provides a method to selectively reduce the number of actual fitness function evaluations performed by considering the similarity/indistinguishability of an individual to a pool of fuzzy information granules. Since the proposed approach does not use approximation or online training, it is not caught in the pitfalls of such techniques such as false peaks, large approximation error due to extrapolation, and time consuming online training.

The effectiveness and functionality of the proposed approach was verified through four structural design problems. In the first three of them, the objective was to increase the first natural frequency of the structure. In the last problem, a piezoelectric actuator was considered for the purposes of shape control and/or active control for correction of static deformations. The design variables were the voltage and the actuator locations and the performance index was considered as the square root of the error between the nodal predefined displacement and the observed displacement.

References

[1] Abe, A., Kamegawa, T., and Nakajima, Y. (2003). Optimization of construction of tire reinforcement by genetic algorithm. *Optimization and Engineering*, 5(1):77–92.

[2] Adali, S., Sadek, I., Jr., J. B., and Sloss, J. (2005). Optimization of composite plates with piezoelectric stiffener-actuators under in-plane compressive loads. *Composite Structures Journal*, 71:293–301.

[3] Akbarzadeh-T, M., Davarynejad, M., and Pariz, N. (2008). Adaptive fuzzy fitness granulation for evolutionary optimization. *International Journal of Approximate Reasoning*, 49(3):523–538.

[4] Alba, E. and Tomassini, M. (2002). Parallelism and Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462.

[5] Ansys, I. (2004). ANSYS users manual. *ANSYS Inc., Southpointe*, 275.

[6] Aryana, F., Bahai, H., Mirzaeifar, R., and Yeilaghi, A. (2007). Modification of dynamic characteristics of FGM plates with integrated piezoelectric layers using first-and second-order approximations. *International Journal for Numerical Methods in Engineering*, 70(12):1409–1429.

[7] Bhattacharya, M. and Lu, G. (2003). A dynamic approximate fitness based hybrid ea for optimization problems. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 1879–1886.

[8] Branke, J. and Schmidt, C. (2005). Fast convergence by means of fitness estimation. *Soft Computing Journal*, 9(1):13–20.

[9] Branke, J., Schmidt, C., and Schmeck, H. (2001). Efficient fitness estimation in noisy environment. In et al, L. S., editor, *Proceedings of Genetic and Evolutionary Computation Conference (GECCO)*, pages 243–250, San Francisco, CA. Morgan Kaufmann.

[10] Coello, C. A. C. (2002). Theoretical and Numerical Constraint Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287.

[11] Coello Coello, C. A., Lamont, G. B., and Van Veldhuizen, D. A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, second edition. ISBN 978-0-387-33254-3.

[12] da Mota Silva, S., Ribeiro, R., Rodrigues, J. D., Vaz, M. A. P., and Monteiro, J. M. (2004). The application of genetic algorithms for shape control with piezoelectric patches-an experimental comparison. *Smart Materials and Structures*, 13:220–226.

[13] Davarynejad, M. (2007). Fuzzy Fitness Granulation in Evolutionary Algorithms for Complex Optimization. Master's thesis, Ferdowsi University of Mashhad.

[14] Davarynejad, M., Ahn, C. W., Vrancken, J. L. M., van den Berg, J., and Coello, C. A. C. (2010). Evolutionary hidden information detection by granulation-based fitness approximation. *Applied Soft Computing*, 10(3):719–729.

[15] Davarynejad, M., Akbarzadeh-T, M., and Coello, C. A. C. (2008). Auto-tuning fuzzy granulation for evolutionary optimization. In *CEC 2008, IEEE World Congress on Evolutionary Computation*, pages 3572–3579, Hong Kong.

[16] Davarynejad, M., Akbarzadeh-T, M.-R., and Pariz, N. (2007). A novel general framework for evolutionary optimization: Adaptive fuzzy fitness granulation. In *IEEE Congress on Evolutionary Computation*, pages 951–956. IEEE.

[17] Ducheyne, E. I., De Baets, B., and De Wulf, R. (2003). Is Fitness Inheritance Useful for Real-World Applications? In Fonseca, C. M., Fleming, P. J., Zitzler, E., Deb, K., and Thiele, L., editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 31–42, Faro, Portugal. Springer. Lecture Notes in Computer Science. Volume 2632.

[18] Fonseca, L. G. and Barbosa, H. J. C. (2009). A similarity-based surrogate model for enhanced performance in genetic algorithms. *OPSEARCH*, 46:89107.

[19] Freudenberger, J., Gllner, J., Heilmaier, M., Mook, G., Saage, H., Srivastava, V., and Wendt, U. (2009). Materials science and engineering. In Grote, K. H. and Antonsson, E. K., editors, *Springer Handbook of Mechanical Engineering*. Springer Berlin Heidelberg.

[20] Furuya, H. and Haftka, R. T. (1993). Locating actuators for vibration suppression on space trusses by genetic algorithms. *ASME-PUBLICATIONS-AD*, 38.

[21] Giger, M. and Ermanni, P. (2005). Development of CFRP racing motorcycle rims using a heuristic evolutionary algorithm approach. *Structural and Multidisciplinary Optimization*, 30(1):54–65.

[22] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Co., Reading, Massachusetts.

[23] Hong, Y.-S., H.Lee, and Tahk, M.-J. (2003). Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks. *Engineering Optimization*, 35(1):91–102.

[24] Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12.

[25] Jin, Y., Olhofer, M., and Sendhoff, B. (2000). On evolutionary optimization with approximate fitness functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 786–792. Morgan Kaufmann.

[26] Joseffsson, L. and Persson, P. (2006). *Conformal Array Antenna Theory and Design*. John wiley & sons.

[27] Kelly, D. W., Gago, J. P. D. S. R., Zienkiewicz, O. C., and Babuska, I. (1983). A posteriori error analysis and adaptive processes in the finite element method: Part ierror analysis. *International Journal for Numerical Methods in Engineering*, 19:1593–1619.

[28] Khorsand, A.-R. and Akbarzadeh, M. (2007). Multi-objective meta level soft computing-based evolutionary structural design. *Journal of the Franklin Institute*, pages 595–612.

[29] Khorsand, A.-R., Akbarzadeh-T, M.-R., and Moin, H. (2006). Genetic Quantum Algorithm for Voltage and Pattern Design of Piezoelectric Actuator. In *IEEE Congress on Evolutionary Computation, CEC 2006*, pages 2593–2600.

[30] Kim, H.-S. and Cho, S.-B. (2001). An efficient genetic algorithms with less fitness evaluation by clustering. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 887–894. IEEE.

[31] Krommer, M. (2005). Dynamic shape control of sub-sections of moderately thick beams. *Computers & Structures*, 83(15-16):1330–1339.

[32] Lemonge, A., Barbosa, H., and Fonseca, L. (2003). A genetic algorithm for the design of space framed structures. In *XXIV CILAMCE–Iberian Latin-American Congress on Computational Methods in Engineering, Ouro Preto, Brazil*.

[33] Li, J., Sedaghati, R., Dargahi, J., and Waechter, D. (2005). Design and development of a new piezoelectric linear Inchworm actuator. *Mechatronics Journal*, 15:651–681.

[34] Liew, K., He, X., and Ray, T. (2004). On the use of computational intelligence in the optimal shape control of functionally graded smart plates. *Computer Methods in Applied Mechanics and Engineering*, 193(42-44):4475–4492.

[35] Lim, D., Ong, Y. S., Jin, Y., and Sendhoff, B. (2006). Trusted evolutionary algorithm. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC'2006)*, pages 149–156.

[36] Lin, J. and Nien, M. (2005). Adaptive control of a composite cantilever beam with piezoelectric damping-modal actuators/sensors. *Composite Structures Journal*, 70:170–176.

[37] Mackerle, J. (2003). Smart materials and structuresa finite element approachan addendum: a bibliography (1997 2002). *Modelling and Simulation in Materials Science and Engineering*, 11(5):707–744.

[38] Mezura-Montes, E., editor (2009). *Constraint-Handling in Evolutionary Optimization*. Springer, Berlin, Germany. ISBN 978-3-642-00618-0.

[39] Michalewicz, Z. (1994). *Genetic algorithms + data structures = evolution programs*. Springer-Verlag New York, Inc., New York, NY, USA.

[40] Nguyen, Q. and Tong, L. (2004). Shape control of smart composite plate with non-rectangular piezoelectric actuators. *Composite Structures*, 66(1-4):207–214.

[41] Ong, Y., Nair, P., and Keane, A. (2003). Evolutionary optimization of computationally expensive problems via surrogate modeling. *American Institute of Aeronautics and Astronautics Journal*, 41(4):687–696.

[42] Ong, Y. S., Zhu, Z., and Lim, D. (2006). Curse and blessing of uncertainty in evolutionary algorithm using approximation. In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC'2006)*, pages 2928–2935.

[43] Papadrakakis, M., Lagaros, N., and Kokossalakis, G. (2000). Evolutionary Algorithms Applied to Structural Optimization Problems. *High Performance Computing for Computational Mechanics*, pages 207–233.

[44] Pelikan, M. and Sastry, K. (2004). Fitness inheritance in the Bayesian optimization algorithms. In *Genetic and Evolutionary Computation Conference*, pages 48–59. Springer.

[45] Piefort, V. (2001). *Finite element modelling of piezoelectric active structures*. PhD thesis, Université Libre de Bruxelles.

[46] Ratle, A. (1998). Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In Eiben, A., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature*, volume V, pages 87–96.

[47] Reddy, J. (1993). *Introduction to the Finite Element Method*. McGrawHill, New York.

[48] Regis, R. and Shoemaker, C. (2004). Local function approximation in evolutionary algorithms for the optimization of costly functions. *IEEE Transactions on Evolutionary Computation*, 8(5):490–505.

[49] Reyes Sierra, M. and Coello Coello, C. A. (2005a). Fitness Inheritance in Multi-Objective Particle Swarm Optimization. In *2005 IEEE Swarm Intelligence Symposium (SIS'05)*, pages 116–123, Pasadena, California, USA. IEEE Press.

[50] Reyes Sierra, M. and Coello Coello, C. A. (2005b). A Study of Fitness Inheritance and Approximation Techniques for Multi-Objective Particle Swarm Optimization. In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, volume 1, pages 65–72, Edinburgh, Scotland. IEEE Service Center.

[51] Rodríguez, J. E., Medaglia, A. L., and Coello, C. A. C. (2009). Design of a motorcycle frame using neuroacceleration strategies in MOEAs. *Journal of Heuristics*, 15(2):177–196.

[52] Runarsson, T. P. (2004). Constrained Evolutionary Optimization by Approximate Ranking and Surrogate Models. In Yao, X., Burke, E., Lozano, J. A., Smith, J., , Merelo-Guervós, J. J., Bullinaria, J. A., Rowe, J., Tiňo, P., Kabán, A., and Schwefel, H.-P., editors, *Proceedings of 8th Parallel Problem Solving From Nature (PPSN VIII)*, pages 401–410, Heidelberg, Germany. Birmingham, UK, Springer-Verlag. Lecture Notes in Computer Science Vol. 3242.

[53] Runarsson, T. P. and Yao, X. (2000). Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294.

[54] Sacks, J., Welch, W., Mitchell, T., and Wynn, H. (1989). Design and analysis of computer experiments (with discussion). In *Statistical Science*, volume 4, pages 409 – 435.

[55] Salami, M. and Hendtlass, T. (2003). A fast evaluation strategy for evolutionary algorithms. *Applied Soft Computing*, 2:156–173.

[56] Sano, Y. and Kita, H. (2000). Optimization of noisy fitness functions by means of genetic algorithms using history. In et al, M. S., editor, *Parallel Problem Solving from Nature (PPSN)*, volume 1917 of *Lecture Notes in Computer Science*. Springer.

[57] Santana-Quintero, L. V., Arias Montaño, A., and Coello Coello, C. A. (2010). A Review of Techniques for Handling Expensive Functions in Evolutionary Multi-Objective Optimization. In Tenne, Y. and Goh, C.-K., editors, *Computational Intelligence in Expensive Optimization Problems*, pages 29–59. Springer, Berlin, Germany. ISBN 978-3-642-10700-9.

[58] Shi, L. and Rasheed, K. (2010). A survey of fitness approximation methods applied in evolutionary algorithms. In Hiot, L. M., .Ong, Y. S., Tenne, Y., and Goh, C. K., editors, *Computational Intelligence in Expensive Optimization Problems*, volume 2 of *Adaptation Learning and Optimization*, pages 3–28. Springer Berlin Heidelberg.

[59] Singh, H. K., Ray, T., and Smith, W. (2010). C-PSA: Constrained Pareto simulated annealing for constrained multi-objective optimization. *Information Sciences*, 180(13):2499–2513.

[60] Smith, R., Dike, B., and Stegmann, S. (1995). Fitness inheritance in genetic algorithms. In *Proceedings of ACM Symposiums on Applied Computing*, pages 345–350. ACM.

[61] Walker, M. and Smith, R. E. (2003). A technique for the multiobjective optimisation of laminated composite structures using genetic algorithms and finite element analysis. *Composite Structures*, 62(1):123–128.

[62] Weise, T. (2008). Global Optimization Algorithms–Theory and Application. *URL: http://www. it-weise. de, Abrufdatum*, 1.

[63] Woldesenbet, Y. G., Yen, G. G., and Tessema, B. G. (2009). Constraint Handling in Multiobjective Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 13(3):514–525.

[64] Won, K. S., Ray, T., and Tai, K. (2003). A framework for optimization using approximate functions. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 1077–1084.

[65] Zadeh, L. A. (1979). Fuzzy sets and information granularity. *Advances in Fuzzy Set Theory and Applications*, pages 3–18.

[66] Zhang, X., Julstrom, B., and Cheng, W. (1997). Design of vector quantization codebooks using a genetic algorithm. In *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 525–529. IEEE.

3

Evolutionary Hidden Information Detection by Granulation-Based Fitness approximation [1]

Abstract

Spread spectrum audio watermarking (SSW) is one of the most powerful techniques for secure audio watermarking. SSW hides information by spreading the spectrum. The hidden information is called the "watermark" and is added to a host signal, making the latter a watermarked signal. The spreading of the spectrum is carried out by using a pseudo-noise (PN) sequence. In conventional SSW approaches, the receiver must know both the PN sequence used at the transmitter and the location of the watermark in the watermarked signal for detecting the hidden information. Detection of the PN sequence is the key issue of hidden information detection in SSW. Although the PN sequence can be reliably detected by means of heuristic approaches, due to the high computational cost of this task, such approaches tend to be too computationally expensive to be practical. Evolutionary Algorithms (EAs) belong to a class of such approaches. Most of the computational complexity involved in the use of EAs arises from fitness function evaluation that may be either very difficult to define or computationally very expensive to evaluate. This chapter proposes an approximate model, called Adaptive Fuzzy Fitness Granulation with Fuzzy Supervisor (AFFG-FS), to replace the expensive fitness function evaluation. First, an intelligent guided technique

[1] This chapter is based on:

- M. Davarynejad, C.W. Ahn, J. Vrancken, J. van den Berg, C.A. Coello Coello, "Evolutionary hidden information detection by granulation-based fitness approximation", *Applied Soft Computing*, Vol. 10(3), pp. 719-729, 2010, DOI: 10.1016/j.asoc.2009.09.001.

via an adaptive fuzzy similarity analysis for fitness granulation is used for deciding on the use of the exact fitness function and dynamically adapting the predicted model. Next, in order to avoid manually tuning parameters, a fuzzy supervisor as auto-tuning algorithm is employed. Its effectiveness is investigated with three traditional optimization benchmarks of four different choices for the dimensionality of the search space. The effect of the number of granules on the rate of convergence is also studied. The proposed method is then applied to the hidden information detection problem to recover a PN sequence with a chip period equal to 63, 127 and 255 bits. In comparison with the standard application of EAs, experimental analysis confirms that the proposed approach has an ability to considerably reduce the computational complexity of the detection problem without compromising performance. Furthermore, the auto-tuning of the fuzzy supervisor removes the need of exact parameter determination.

3.1 Introduction

In recent years, digital watermarking has received due attention from the security and cryptography research communities. Digital watermarking is a technique to hide information into an innocuous-looking media object, which is called "host", so that no one can suspect the existence of hidden information. It is intended to provide a degree of copyright protection as use of digital media mushrooms [9]. Depending on the type of the host signal to cover hidden information, watermarking is classified into *image watermarking* and *audio watermarking*. In this chapter, we focus our attention on audio watermarking but the approach can be applied to image watermarking as well.

Numerous audio watermarking techniques have been proposed and the most important ones being Least Significant Bits (LSB) [13], Phase coding [3], Echo hiding [18] and spread spectrum watermarking (SSW) [19]. The latter, SSW, is known as the most promising watermarking method due to its high robustness against noise and high perceptual transparency. The main idea of SSW is to add the spread spectrum of hidden information to the spectrum of the host signal. Spreading the spectrum of the hidden information is performed by means of a pseudo-random noise (PN) sequence.

Detection of hidden information from the received watermark signal is performed using the exact PN sequence adopted for spreading the spectrum of hidden information. Therefore, the receiver should have access to the PN sequence for detection. This essential, private knowledge results in a highly secure transmission of information against any unauthorized user who does not have access to the PN sequence and the location of the watermark. Hence, the PN sequence can be regarded as a secret key which is shared between the transmitter and the receiver.

In [26], genetic algorithms (GAs) have been presented for detecting hidden information, even though the receiver has no prior knowledge on the transmitter's spreading sequence. However, iterative fitness function evaluation for such a complex problem is often the most prohibitive and limiting segment of this approach. For the problem of recovering the PN sequence, sequences with different periods have different converging times. In the study reported in [26], it has been shown that converging time increases exponentially as the period of the PN sequence increases. So, the approach fails by losing the validity of information. The greater the PN sequence is, the more difficult is the situation for recovering the PN

sequence and the more secure SSW will result. Note hereby that a greater period of the PN sequence decreases the capacity of the SSW algorithm for embedding hidden information. To alleviate the problem of exponentially increasing converging times, a variety of techniques for constructing approximation models - often referred to as *metamodels* - have been proposed [8, 14]. For computationally expensive optimization problems such as the detection of hidden information, it may be necessary to strike a balance between exact fitness evaluation and approximate fitness evaluation. A popular subclass of fitness function approximation methods is fitness inheritance where fitness is simply inherited [8]. A similar approach named *Fast Evolutionary Strategy* (FES) has also been suggested in [25], in which the fitness of a child individual is the weighted sum of its parents. In that approach, fitness and associated reliability values are assigned to each new individual, and then the actual fitness function is only evaluated when the reliability value is below a certain threshold. Further, Reyes Sierra and Coello Coello [24] incorporated the concept of fitness inheritance into a multi-objective particle swarm optimizer to reduce the number of fitness evaluations. In [23], they tested their approach on a well-known test suite of multi-objective optimization problems. They generally reported lower computation cost, while the quality of their results improved in higher dimensional spaces. However, as also shown in [12] as well as in this chapter, the performance of parents may not be a good predictor of their children for sufficiently complex and multiobjective problems, rendering fitness inheritance inappropriate under such circumstances.

Other common approaches based on learning and interpolation from known fitness values of a small population, (e.g. low-order polynomials and least square estimations [21], artificial neural networks (ANN) including multi-layer perceptrons [16] and radial basis function networks [29], support vector machines (SVM) [14, 28], etc.) can also be employed.

In 1979, Zadeh [31] developed fuzzy information granulation as a technique by which a class of points (objects) is partitioned into granules, with a granule being a clump of objects drawn together by indistinguishability, similarity, and/or functionality. The fuzziness of granules and their attributes is characteristic of the ways by which human concepts and reasoning are formed, organized and manipulated. The concept of a granule is more general than that of a cluster, potentially giving rise to various conceptual structures in various fields of science as well as in mathematics.

In this chapter, with a view to reducing computational cost, we employ the concept of fuzzy granulation to effectively approximate the fitness function in evolutionary algorithms (EAs). In other words, the concept of fitness granulation is applied to exploit the natural tolerance of EAs in fitness function computations. Nature's "survival of the fittest" does not necessarily mean exact measures of fitness; rather it is about rankings among competing peers [17]. By exploiting this natural tolerance for imprecision, optimization performance can be preserved through computing fitness only selectively based on the ranking among individuals in a given population. Unlike existing approaches, the fitness values are not interpolated or estimated; rather the similarity and indistinguishability among real solutions is exploited.

In the proposed algorithm, as explained in detail in [11], an adaptive pool of solutions (fuzzy granules) with an exactly computed fitness function is maintained. If a new individual is sufficiently similar to a known fuzzy granule, then that granule's fitness is used instead as a crude estimate. Otherwise, the individual is added to the pool as a new fuzzy

granule. In this fashion, regardless of the competition's outcome, fitness of the new individual is always a physically realizable one, even if it is a "crude" estimate and not an exact measurement. The pool size as well as each granule's radius of influence self-adaptively grow or shrink depending on the utility of each granule and the overall population fitness. To encourage fewer function evaluations, each granule's radius of influence is initially large and then gradually shrinks in the course of evolution. This encourages more exact fitness evaluations when competition is fierce among more similar and converging solutions. Furthermore, to prevent the pool from growing too large, granules that are not used are gradually eliminated. This fuzzy granulation scheme is applied here as a type of fuzzy approximation model to efficiently detect hidden information from spread spectrum watermarked signals. Finally, a fuzzy supervisor is developed for adaptively, automatically adjusting system parameters. The chapter is organized as follows: Section 3.2 presents the framework of adaptive fuzzy fitness granulation (AFFG). An auto-tuning strategy for determining width of membership functions (MFs) is also presented in the section; by which the need of exact parameter setting is eliminated, without affecting the rate of convergence. This approach is called *adaptive fuzzy fitness granulation with fuzzy supervisory* (AFFG-FS). In Section 3.3, the proposed algorithm is tested on three traditional optimization benchmarks with four different dimensions. In Section 3.4, the recovery of the PN sequence from a received watermarked signal using the proposed approach is illustrated. Some supporting simulation results and discussion thereof are also presented in the section. Finally, conclusions are drawn in Section 3.5.

3.2 The AFFG Framework

Adaptive fuzzy fitness granulation (AFFG) was first proposed in [11]. It includes a global model of a genetic algorithm (GA) which is hybridized with a fuzzy granulation (FG) tool (see Figure 3.1). The expensive fitness evaluation of individuals required in traditional GA, can be partially replaced by an approximation model. Explicit control strategies are used for evolution control, leading to a considerable speedup without compromising heavily on the solution accuracy. While the approximation techniques themselves are widely known for accelerating the iterative optimization process, the focus of AFFG lies in promoting controlled speedup in view of avoiding detrimental effects of the approximation. The following section presents the main elements of the AFFG framework.

3.2.1 Basic Idea

The proposed adaptive fuzzy fitness granulation aims to minimize the number of exact fitness function (FF) evaluations by maintaining a pool of solutions (fuzzy granules) by which can be used to approximate solutions in further stages of the evolutionary process. The algorithm uses Fuzzy Similarity Analysis (FSA) to produce and update an adaptive competitive pool of dissimilar solutions (granules). When a new solution is introduced to this pool, granules compete by a measure of similarity to win the new solution and thereby to prolong their lives in the pool. In turn, the new individual simply assumes fitness of the winning (most similar) individual in this pool. If none of the granules are sufficiently similar to the new individual (i.e., if their similarity is below a certain threshold), the new individual is

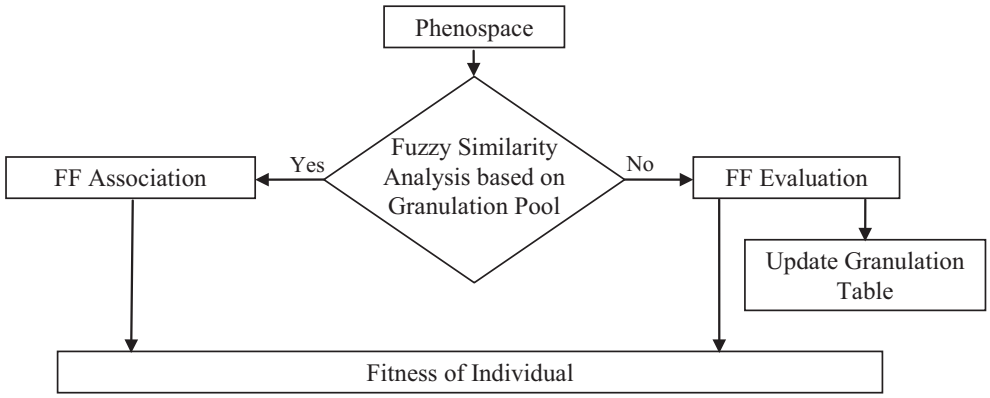

Figure 3.1: The architecture of the proposed algorithm.

instead added to the pool after its exact fitness is evaluated by the actual fitness function. Finally, granules that cannot win new individuals are gradually eliminated in order to avoid consistent growth of the pool. The basic idea of the proposed algorithm is graphically shown in Figure 3.1 and is discussed in more detail in the next section. For even more details, we refer to [11] and [2].

3.2.2 Basic Algorithm Structure

Step 1: Create a random parent population $P^1 = \{\mathbf{x}_1^1, \mathbf{x}_2^1, \ldots, \mathbf{x}_j^1, \ldots, \mathbf{x}_t^1\}$ of design variable vector, where, more generally, $\mathbf{x}_j^i = \{x_{j,1}^i, x_{j,2}^i, \ldots, x_{j,r}^i, \ldots, x_{j,m}^i\}$ is the jth parameter individual in the ith generation, $x_{j,r}^i$ the rth parameter of $\mathbf{x}_j^i \in \mathbf{R}$, m is the number of design variables and t is the population size.

Step 2: Define a multi-set G of fuzzy granules (C_k, σ_k, L_k) according to $G = \{(C_k, \sigma_k, L_k)|C_k \in \mathbf{R}^m, \sigma_k \in \mathbf{R}, L_k \in \mathbf{R}, k = 1,\ldots,l\}$. G is initially empty (i.e., $l = 0$). C_k is an m-dimensional vector of centers, σ_k is the width of membership function (WMF) of the kth fuzzy granule, and L_k is the granule's life index.

Step 3: Choose the phenotype of first chromosome $(\mathbf{x}_1^1 = \{x_{1,1}^1, x_{1,2}^1, \ldots, x_{1,r}^1, \ldots, x_{1,m}^1\})$ as the center of the first granule $(C_1 = \{c_{1,1}, c_{1,2}, \ldots, c_{1,r}, \ldots, c_{1,m}\} = \mathbf{x}_1^1)$.

Step 4: Define the fuzzy membership $\mu_{k,r}$ of each $x_{j,r}^i$ to each granule member by a Gaussian similarity neighborhood function according to

$$\mu_{k,r}\left(x_{j,r}^i\right) = \exp\left(\frac{-\left(x_{j,r}^i - c_{k,r}\right)^2}{\left(\sigma_k\right)^2}\right),\ k = 1,2,\ldots,l\ , \tag{3.1}$$

where l is the number of fuzzy granules.

Remark: σ_k is the distance measurement parameter that controls the degree of similarity between two individuals. Like in [10], σ_k is defined based on equation (3.2). According to

this definition, the granules shrink or enlarge in reverse proportion to their fitness:

$$\sigma_k = \gamma \frac{1}{\left(e^{F(C_k)}\right)^{\beta}}\ , \tag{3.2}$$

where $\beta > 0$ is an emphasis operator and γ is a proportionality constant. The problem arising here is how to determine the parameters β and γ as design parameters. The fact is that these two parameters are problem dependent and, in practice, a number of trials is needed to adjust these parameters. This trial is based on a simple rule with respect to the acceleration of the parameter optimization procedure: high speed needs to have enlargement in the granule spread and, as a consequence of this, less accuracy is obtained in the fitness approximation, and viceversa. To deal with this rule, a fuzzy controller is presented in [10].

Step 5: Compute the average similarity of every new design parameter $\mathbf{x}_j^i = \{x_{j,1}^i, x_{j,2}^i, \ldots, x_{j,r}^i, \ldots, x_{j,m}^i\}$ to each granule G_k using equation (3.3)

$$\overline{\mu}_{j,k} = \frac{\sum_{r=1}^{m} \mu_{k,r}\left(x_{j,r}^i\right)}{m} \tag{3.3}$$

Step 6: Either calculate the exact fitness function of $\mathbf{x}_j^i$ or estimate the fitness function value by associating it to one of the granules in the pool in case there is a granule in the pool with higher similarity to $\mathbf{x}_j^i$ than a predefined threshold, i.e.

$$f\left(\mathbf{x}_j^i\right) = \begin{cases} f\left(C_k\right) & \text{if } \max\limits_{k\in\{1,2,\ldots,l\}} \{\overline{\mu}_{j,k}\} > \theta^i\ , \\ f\left(\mathbf{x}_j^i\right) & \text{otherwise.} \end{cases} \tag{3.4}$$

where $f(C_x)$ is the fitness function value of the fuzzy granule, $f(\mathbf{x}_j^i)$ is the real fitness calculation of the individual, $\theta^i = \alpha(\max\{f(\mathbf{x}_1^{i-1}), f(\mathbf{x}_2^{i-1}), \ldots, f(\mathbf{x}_t^{i-1})\}/\overline{f}^{i-1})$, $K = \text{argmax}\{\overline{\mu}_{j,k}\}$ when $k \in \{1,2,\ldots,l\}$, $\overline{f}^i = \sum_{j=1}^{i} f(\mathbf{x}_j^i)/t$ and $\alpha > 0$ is a proportionality constant that is usually set at 0.9 unless otherwise indicated. The threshold θ^i increases as the best individual's fitness at generation i increases. As the population matures and reaches higher fitness values (i.e., while converging more), the algorithm becomes more selective and uses exact fitness calculations more often. Therefore, with this technique we can utilize the previous computational efforts during previous generations. Alternatively, if

$$\max_{k\in\{1,2,\ldots,l\}} \{\overline{\mu}_{j,k}\} < \theta^i$$

$\mathbf{x}_j^i$ is chosen as a newly created granule.

Step 7: If the population size is not completed, repeat **Steps 5 to 7**.

Step 8: Select parents using a suitable selection operator and apply the genetic operators of recombination and mutation to create a new generation.

Step 9: When termination/evolution control criteria are not met, then update σ_k using equation (3.2) and repeat **Steps 5 to 9**.

3.2.3 How to control the size of the granule pool?

As the evolutionary procedures proceed, it is inevitable that new granules are generated and added to the pool. Depending on complexity of the problem, the size of this pool can become excessive and become a computational burden itself. To prevent such unnecessary computational effort, a *forgetting factor* is introduced in order to appropriately decrease the size of the pool. In other words, it is better to remove granules that do not win new individuals, thereby producing a bias against individuals that have low fitness and were likely produced by a failed mutation attempt. Hence, L_k is initially set to N and subsequently updated as below,

$$L_k = \begin{cases} L_k + M & \text{if } k = K\,, \\ L_k & otherwise\,, \end{cases} \tag{3.5}$$

where M is the life reward of the granule and K is the index of the winning granule for each individual at generation i. At each table update, only the N_G granules with the highest L_k index are kept, and the others are discarded. In [2], an example has been provided to illustrate the competitive granule pool update law.

Adding a new granule to the granule pool and assigning a life index to it, is a simple way of controlling the size of the granule pool, since the granules with the lowest life index will be removed from the pool. However, it may happen that the new granule is removed, even though it was just inserted into the pool. In order to prevent this, the pool is split into two parts with sizes εN_G and $(1-\varepsilon)N_G$. The first part is a FIFO (First In, First Out) queue and new granules are added to this part. If it grows above εN_G, then the top of the queue is moved to the other part. Removal from the pool takes place only in the $(1-\varepsilon)N_G$ part. In this way, new granules have a good chance to survive a number of steps. In all of the simulations that are conducted here, ε is set at 0.1.

The distance measurement parameter is completely influenced by the granule enlargement/shrinkage in the widths of the produced membership functions. As in [11], the combined effect of granule enlargement/shrinkage is in accordance with the granule fitness and it requires the fine-tuning of two parameters, namely β and γ. These parameters are problem dependent and it seems critical to have a procedure to deal with this difficulty. The next section presents an auto-tuning strategy for determining the width of MFs which removes the need of exact parameter determination, without negative influence on the convergence speed.

3.2.4 How to Determine the Width of the Membership Functions

It is crucial to have accurate estimations of the fitness function of the individuals in the finishing generations. In the proposed method, it can be accomplished by controlling the width of the produced MFs. At early steps of evolution, by choosing relatively large WMFs, the algorithm accepts individuals with less degree of similarity as similar individual. Therefore at the early stages of the search, the fitness function is more often estimated. As the individuals mature and reach better fitness values, the width decreases and the similarity between individuals should increase in order to be accepted as similar individuals. This prompts higher selectivity for granule associability and a higher threshold for estimation. In short, in later generations, the degree of similarity between two individuals must be larger than that

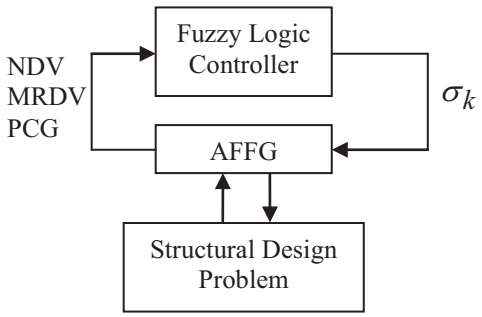

Figure 3.2: Flow-diagram of Adaptive Fuzzy Controller.

in the early generations, to be accepted as similar individuals. This procedure ensures a fast convergence rate due to rapid computation at the early phase and accurate fitness estimation at the later stage.

To achieve these desiderata, a fuzzy supervisor with three inputs is employed. During the AFFG search, the fuzzy logic controller observes the Number of Design Variables (NDV), the Maximum Range of Design Variables (MRDV) and the percentage of completed trials, and specifies the WMFs. The first input is the NDV and the Range of the input variables (RIV) is the second one. Large values of the NDV and MRDV need big width in the MFs, *vice versa*. The Percent Completed Generations (PCG) is the third input, which takes a number in the range [0, 1], where "1" signifies exhaustion of all allowed trials. This concerns the maturity level of search, given a fixed amount of resources. The combined effect of granule enlargement/shrinkage in accordance to PCG is to realize both rapid computation and accurate fitness estimation.

The architecture for adaptive fuzzy control of the WMFs is visualized in Figure 3.2. Gaussian MFs are used for specification of the knowledge base of the fuzzy logic controller. The knowledge base for controlling the WMFs based on the above architecture has a large number of rules and the extraction of these rules is very difficult. Consequently, a new architecture (as shown in Figure 3.3) is proposed, in which the controller is separated in two controllers to diminish the complexity of the system and to reduce the number of rules. The first controller has two inputs (with three MFs in each, Zero(0, 0.3), Small(0.5, 0.3), Big(1.0, 0.3), the first number is the center and the second one is the spread), and the second controller has only one input. As shown in Figure 3.3, the spread of the granules is provided by the multiple output of the controllers. The knowledge base for the first controller is shown in Table 3.1. The Gaussian MFs with equal width in each (0.3) are used for output. The second controller has just one Gaussian MF in which 0 and 1.25 are its center and spread, respectively. The fuzzy system (that employs singleton fuzzifier, products inference engine, and center average defuzzifier) adjusts σ_k after each generation.

3.3 Benchmark problems and numerical results

To illustrate the efficacy of the proposed granulation techniques, a set of 3 traditional optimization benchmarks (shown in Table (3.2)) are chosen namely: Ackley, Griewangk and

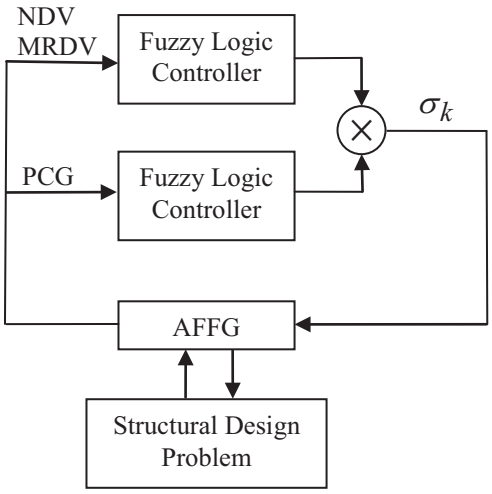

Figure 3.3: Flow-diagram of Proposed Fuzzy Controller.

Table 3.1: Fuzzy Rules of the First Controller.

	NDV		
MRDV	Zero	Small	Big
Zero	228.1	4601.2	6.39×10^{-05}
Small	155.9	511.9	6.34×10^{-05}
Big	97.5	406.7	6.39×10^{-05}

Rastrigin. These benchmark functions are scalable and are commonly used to assess optimization algorithms. They have some intriguing features which most optimization algorithms find hard to deal with.

The Ackley function [1, 7] has an exponential term by which numerous local minima are produced. Analyzing a wider region helps to cross the valley along local optima, thereby achieving better solutions. The global optimum is always $f(\mathbf{x}) = 0$, which is obtained at

Table 3.2: Benchmark problems used in the experiments.

Function name	Mathematical Representation	Original Search space
Ackley	$20+e-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)$ $-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)$	$[-32.768, 32.768]^D$
Griewank	$1+\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	$[-600, 600]^D$
Rastrigin	$10D+\sum_{i=1}^{D}\left(x_i^2-10\cos(2\pi x_i)\right)$	$[-5.112, 5.112]^D$

Table 3.3: Benchmark problems used in the experiments.

Function	β	γ
Ackley	0.02	0.25
Griewank	0.00012	190.0
Rastrigin	0.004	0.15

$x_i = 0, \forall i$.

The Griewangk function [6] is also highly multimodal. Unlike Ackley and Rastrigin functions, it has a product term that introduces interdependence among variables. It is hard to find the optimal solution without some information on the variables' dependencies. Regardless of its dimensionality, the global optimum is $f(\mathbf{x}) = 0$ which occurs at $x_i = 0, \forall i$.

The Rastrigin function [22] is created by adding a cosine modulation term to the Sphere function. It consists of a large number of local minima whose values increase in receding from the global minimum. The global optimum is $f(\mathbf{x}) = 0$ which occurs at $x_i = 0, \forall i$.

The aim of the empirical study consists of investigating the search capability, as a function optimizer, of the proposed granulation technique (AFFG-FS), compared to the conventional GA, FES and AFFG techniques. The parameters are summarized in Table 3.4.

The GA routine utilizes random initial populations, binary-coded chromosomes, single-point crossover, bit-wise mutation, fitness scaling, and an elitist stochastic universal sampling selection strategy. Moreover, crossover and mutation probabilities are $P_{XOVER} = 1$ and $P_{MUTATION} = 0.01$ respectively, the population size is 20, and the maximum number of generations is 100. Finally chromosome length varies depending on the number of variables in a given problem, but each variable's length is set to 8 bits. The total number of generations as well as the termination criterion is determined during several trial runs to ensure the convergence of the algorithm on the three benchmark problems.

AFFG and AFFG-FS uses all of the above evolutionary parameters as in a GA to establish analysis only from the perspective of granulation and in order to keep track of the best solution found. Ten independent runs of each experiment were executed.

As to FES, a fitness and an associated reliability values are assigned to each new individual. The fitness is actually evaluated if the reliability value is below a certain threshold. The reliability value varies between 0 and 1 and depends on two factors: the first one is the reliability of parents, and the second one is the closeness of parents and children in the solution space. Three different levels for T , i.e., 0.5, 0.7 and 0.9, have been used here which equal to ones proposed in [25].

In this experiment, four sets of dimensions are considered for each test function; namely n = 5, 10, 20 and 30. As for both the AFFG and AFFG-FS, N_G changes and is set at 20, 20, 40 and 80 respectively. The reported results were obtained by achieving the same level of fitness evaluations for both the proposed method (AFFG-FS) and the comparative references (GA, FES and AFFG), namely 500 for 5-D (dimension), 1000 for 10-D, 2000 for 20-D and 3000 for 30-D.

The average convergence trends of the standard GA, FES, AFFG and AFFG-FG are summarized in Figures 3.4 to 3.6. All the results presented were averaged over 10 runs. The y-axis in these figures denotes the (average) fitness value in common logarithmic scale, and

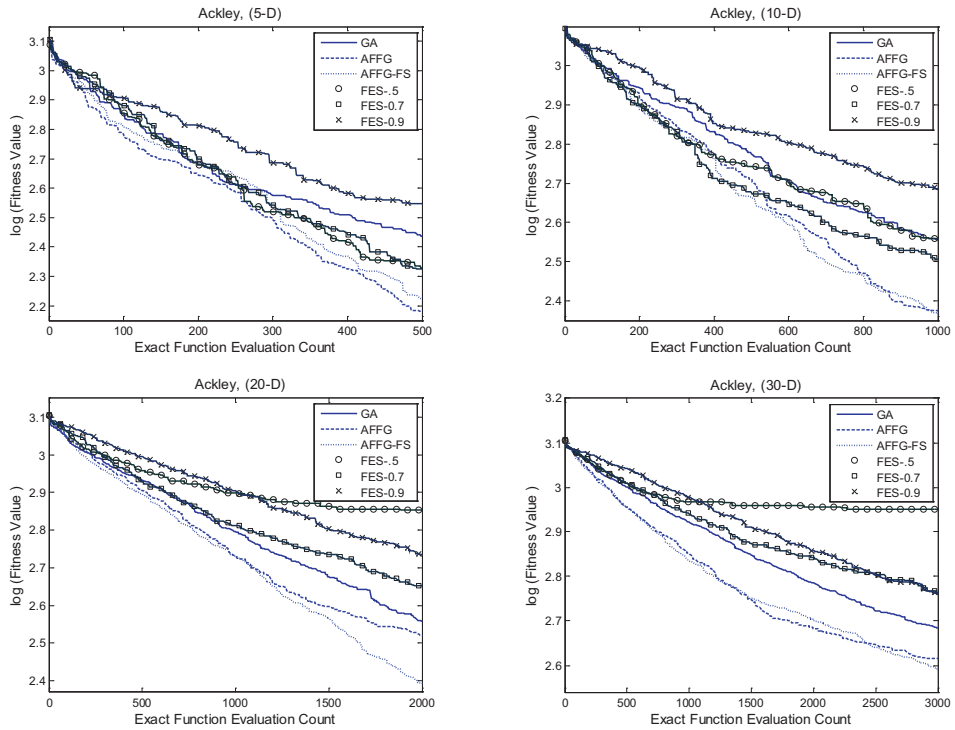

Figure 3.4: Comparisons of convergence curve on the Ackley function.

the x-axis denotes the number of exact function evaluation.

As shown in Figures 3.4 to 3.6, the search performance of AFFG and AFFG-FS are superior to GA and FES, even with a small number of granules in the granule pool. The results also illustrate that fitness inheritance method (i.e., FES), albeit being comparable in smaller dimensions, deteriorates as the problem size increases.

We also studied the effect of varying the number of granules N_G on the convergence behavior of AFFG and AFFG-FS. The comparison can be made by the results obtained in Figure 3.8. The good news from the results is that AFFG and AFFG-FS are not so sensitive to N_G . However, further increase of N_G slows down the rate of convergence due to the imposed computational complexity.

3.4 Spread Spectrum Watermarking (SSW)

This section bears out the effectiveness of the proposed granulation technique in real world applications. We consider a hidden information detection problem such that the correct (pseudorandom noise) PN sequence must be recovered from a spread spectrum watermarked signal. Spread spectrum watermarking (SSW) has been perceived to be a powerful watermarking scheme that offers high robustness (surviving hidden information after noise addition), high transparency (high quality of watermarked signal after addition of hidden information) and high security (against unauthorized users) to hide the bits of information. SSW uses the idea of spread spectrum communication to embed bits of information into a

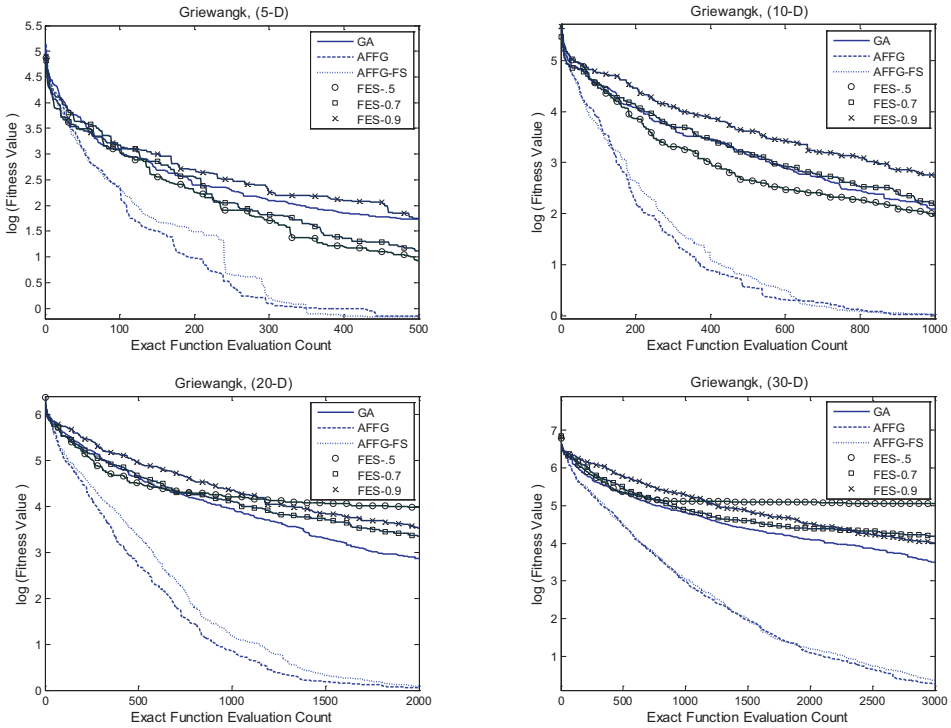

Figure 3.5: Comparisons of convergence curve on the Griewangk function.

host signal. Spreading the spectrum of the hidden information is carried out by a pseudo-random noise sequence. A PN sequence is a zero mean, periodic binary sequence with a noise-like waveform whose bits are equal to +1 or -1 [15]. To embed each bit of hidden information $m(i), i = 1,2\ldots,$ into a host signal, the embedder conducts the following steps.

- Generates one period of the PN sequence by a PN sequence generator.
- Multiply $m(i)$ by all the bits of the generated PN sequence to generate a watermark signal as follows:

$$w(i) = p(n)m(i), n = 1,2,\ldots,N \tag{3.6}$$

  where $p(n)$ is the nth bit of the PN sequence and $w(i)$ is the ith block of the watermark signal.

- Produces a watermarked signal $s(w,x)$ as follows:

$$S(w,x) = \lambda w(n) + x(n) \tag{3.7}$$

Then the watermarked signal $S(w,x)$ is sent to the receiver.

Extraction of hidden information from a received watermarked signal at the detector can be done using the correlation property of the PN sequence. Cross correlation $C(.,.)$ between two PN sequences p_a and p_b is given as 3.8 [20].

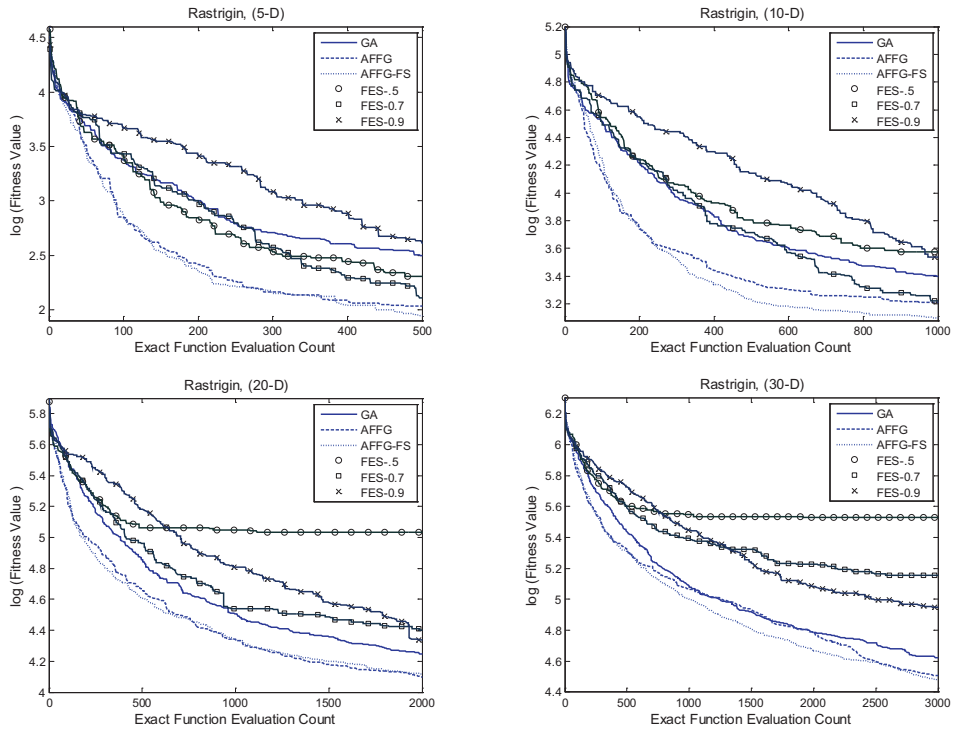

Figure 3.6: Comparisons of convergence curve on the Rastrigin function.

$$C(P_a,P_b)=\frac{1}{N}\sum_{i=0}^{N-1}(P_a(i)P_b(i))=\begin{cases}i, & \text{if } a=b\\ \frac{-1}{N}, & \text{otherwise.}\end{cases} \tag{3.8}$$

Hence, cross correlation between a watermarked signal and a PN sequence can be written a the following.

$$C(S,p')=C(w,p')+C(m.p,p')=\begin{cases}C(w,p')+m, & \text{if } p=p'\\ C(w,p')-\frac{m}{N}, & \text{otherwise.}\end{cases} \tag{3.9}$$

Equation (3.9) expresses that the bit of hidden information can be determined by calculating the correlations between the received watermarked signal and the PN sequence employed at the transmitter, and comparing the result with a threshold.

3.4.1 Recovering the PN sequence

In general, it is very hard to recover the PN sequence from a spread spectrum watermarked signal where no information about the PN sequence or its location is known. The reason is that there are vast regions for the solution sets of possible PN sequences. For instance, to recover a PN sequence with a period equal to 63 bits, 2^{63} PN sequences must be generated.

To make the problem of recovering the PN sequence more tractable, we assume that the exact location of the watermark in the watermarked signal is known. In [27], an approach

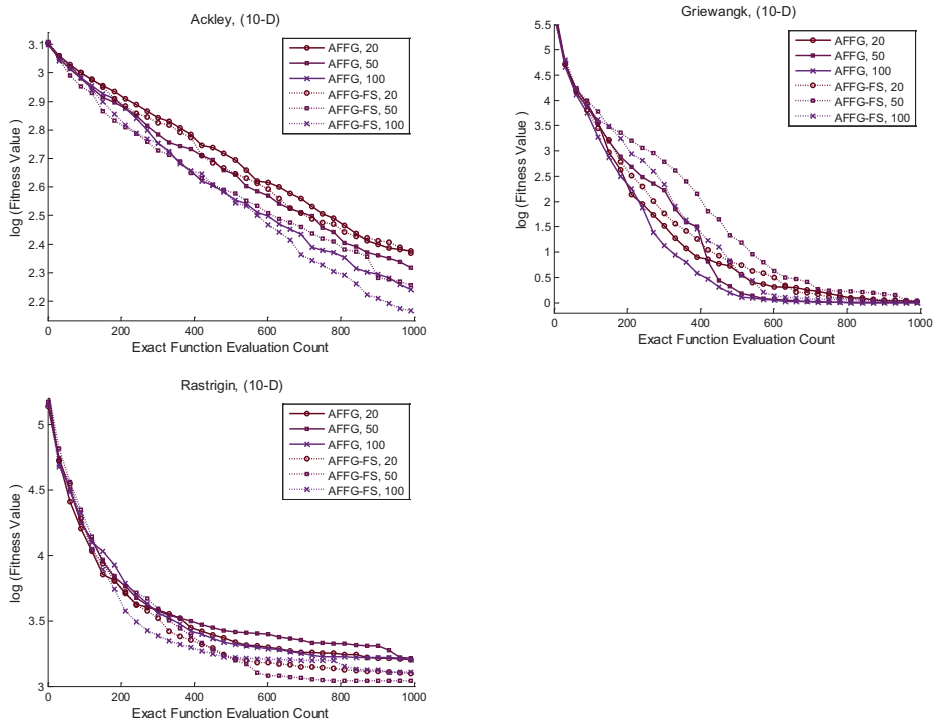

Figure 3.7: Effect of Varying $N_G \in \{20, 50, 100\}$ on convergence trend of studied benchmark optimization problems when D is set at 10.

for detecting hidden information from an image spread spectrum signal has been proposed. This approach detects abrupt jumps in the statistics of the watermarked signal to recover the PN sequence. However, the algorithm which is based on hypothesis tests for detection of abrupt jump in the statistics is very complicated and its performance suffers from low frequency embedding.

Our approach to recover the PN sequence is based on unconstrained optimization. We have a set of feasible solutions available in order to find the global minimum of a cost function. The feasible solutions are sequences with the period length of the PN sequence and elements of +1 and -1. A cost function for this problem can be defined by a exploring a very useful property of SSW (in detection), namely the correlation property of the PN sequence. Thus, the proper cost (fitness) function is the cross correlation between the generated sequence and the watermarked signal as is defined in Equation (3.9).

In [5], an interesting method for recovering the PN sequence of the spread spectrum signal with a predefined SNR has been proposed. The approach uses a GA approach with a fitness function based on the cross correlation between the estimated PN sequence and the spread spectrum. However, spread spectrum watermarking is more complicated than a single spread spectrum signal since, in SSW, the spread spectrum hidden information is like a white Gaussian noise for the host signal.

We observe here that the computation of the cross correlation between the sequences of possible solutions' set and the watermarked signal for only one block of the SSW signal

would not converge to the PN sequence used at the transmitter. This is because the energy of host signal is at least 12 dB more than the energy of the watermark, and that has a strong effect on maximizing the cross correlation (i.e., the optimization algorithm converges to a sequence that maximizes the correlation with the host). As a solution to this problem, several consequence blocks of the watermark (i.e. several bits of hidden information) should be considered in the computation of the cross correlation. In this case, the watermark signal has a stronger effect than the host signal on maximizing the cross correlation function.

Carrying out the global optimization by searching over the entire solution set, as mentioned above, is the subject of deterministic methods such as covering methods, tunneling methods, zooming methods, etc. Such methods discover the global minimum by an exhaustive search over the entire solution set. For instance, the basic idea is to cover all the feasible solutions by evaluating the objective function at all points [4]. Although these schemes have high reliability and accuracy is always guaranteed, they are not practical due to their poor convergence [30].

Since the solution set is vast, we need an efficient optimization algorithm with high reliability and fast converging rate. Many stochastic optimization algorithms have been proposed such as GA, simulated annealing, ant colony, etc. However, the GA approach has been perceived to be promising in a wide range of applications. Moreover, it is apt to strike an attractive balance between reliability and converging rate. In this regard, we have chosen the GA framework for the global optimization task. In order to further enhance the search capability, we employ the proposed AFFG-FS with a view to reduce the number of expensive fitness evaluations by incorporating an approximate model.

Empirical results for recovering PN sequence

This empirical study focuses on performance improvement of the proposed granulation technique (AFFG-FS) in comparison with conventional GA approaches. In Section (3.3), it has been exhibited that the fuzzy supervisory part of AFFG-FS gets rid of the need of exact parameter determination of AFFG, and their performances are comparable to each other. Moreover, it has also been shown that FES is much worse than the granulation techniques. As such, we did not take into account the original AFFG and FES as comparative references in this experiment.

In order to reasonably keep track of the best solution found, the GA uses roulette-wheel selection with elitism. Moreover, one-point crossover and bit-wise mutation are implemented. Crossover and mutation probabilities used are 1.0 and 0.01, respectively. The population size is set to 20 with the elite size of 2.

For AFFG-FS, the number of individuals in the granule pool varies between 10, 20 and 50. The reported results were obtained by achieving the same level of fitness evaluations for both a canonical GA and the proposed AFFG-FS. In this experiment, all results were averaged over 10 runs.

The average of convergence performance of GA and AFFG-FG is depicted in Figure (3.8) and is summarized in Table 3.4. It is seen that cross correlation values returned by AFFG with $N_G = \{10, 20, 50\}$ are much better than that of GA. It is also observed that the cross correlation increases, albeit insensitive, with the number of granules. However, the increase of N_G slows down the rate of convergence due to its imposed computational complexity. Moreover, Table 3.4 exhibits that the rate of convergence of AFFG-FS is, on

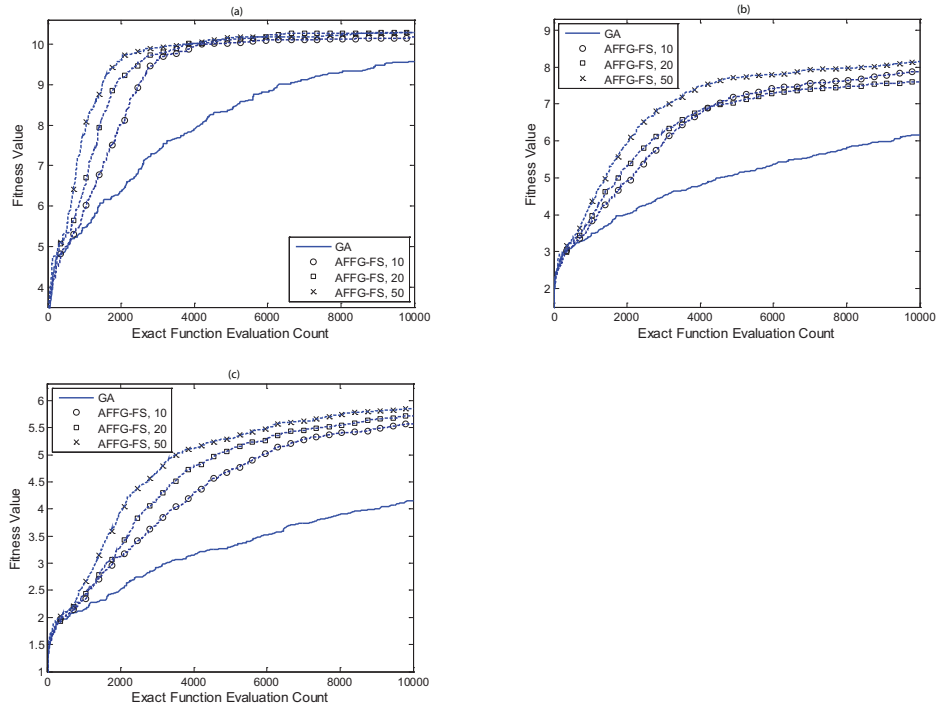

Figure 3.8: Cross correlation between the estimated PN sequence and the watermarked signal when a) PN sequence has a period of 63 chips, b) PN sequence has a period of 127 chips, c) PN sequence has a period of 255 chips, .

average, 3.5 times faster than that of GA. It is noted that the performance gain is not so dependent on the chip length of the PN sequence (i.e., problem size). From the results, it can be concluded that the search performance of AFFG-FS is superior to that of the GA, even with the small number of individuals in the granule pool.

3.5 Concluding Remarks

An intelligent guided technique via an adaptive fuzzy similarity analysis for fitness granulation, called adaptive fuzzy fitness granulation with fuzzy supervisory (AFFG-FS), has been presented. The aim was to decide on the use of expensive function evaluations and adapt the predictive model in a dynamic manner. A fuzzy supervisor as an auto-tuning strategy has also been proposed in order to avoid the tuning of parameters. Empirical evidence on its effectiveness over existing approaches (i.e., GA and FES) was adduced with widely-known benchmark functions. In detail, numerical results showed that the proposed technique is capable of optimizing functions of varied complexity efficiently. It was seen that AFFG and AFFG-FS are not much sensitive to the number of granules N_G, and smaller values of N_G still lead to good results. Moreover, the auto-tuning of fuzzy supervisor eliminated the need for exact parameter determination without compromising convergence performance.

The proposed AFFG-FS has been further applied to the problem of detecting hidden

Table 3.4: Performance comparison of GA and AFFG-FS when $N_G = \{10, 20, 50\}$.

Chip length	Criteria-I [a]	Criteria-II [b]	Criteria-III [c]
63			
GA	10.17	9.57	9965
AFFG-FS, 10	10.29	10.18	2978
AFFG-FS, 20	10.36	10.29	2547
AFFG-FS, 50	10.39	10.28	1904
127			
GA	4.51	4.16	9934
AFFG-FS, 10	5.90	5.58	3817
AFFG-FS, 20	6.10	5.72	2969
AFFG-FS, 50	6.19	5.86	2156
255			
GA	4.51	4.16	9934
AFFG-FS, 10	5.90	5.58	3817
AFFG-FS, 20	6.10	5.72	2969
AFFG-FS, 50	6.19	5.86	2156

[a] The best cross correlation of population at the last generation.
[b] The average cross correlation of population at the last generation.
[c] The average number of fitness evaluations until the same cross correlation value is reached (the values are equal to the average cross correlation of population achieved by GA at the last generation); 4.16 for 255 chips, 6.16 for 127 chips, 9.57 for 63 chips.

information from a spread spectrum watermarked signal. Under the assumption of knowing the location of hidden information, the knowledge necessary for detecting hidden information at the receiver (that is the PN sequence used at the transmitter) could be detected. Experimental studies demonstrated that AFFG-FS is capable of rapidly detecting hidden information.

Acknowledgments

This research received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. INFSO-ICT-223844, the Next Generation Infrastructures Research Program of Delft University of Technology and the Mexican CONACyT project No. 45683-Y.

References

[1] Ackley, D. (1987). An empirical study of bit vector function optimization. *Genetic algorithms and simulated annealing*, 1:170–204.

[2] Akbarzadeh-T, M., Davarynejad, M., and Pariz, N. (2008). Adaptive fuzzy fitness granulation for evolutionary optimization. *International Journal of Approximate Reasoning*, 49(3):523–538.

[3] Ansari, R., Malik, H., and Khokhar, A. (2004). Data-hiding in audio using frequency-selective phase alteration. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'04)*, volume 5, pages 389–392.

[4] Arora, J., Elwakeil, O., Chahande, A., and Hsieh, C. (1995). Global optimization methods for engineering applications: a review. *Structural and Multidisciplinary Optimization*, 9(3):137–159.

[5] Asghari, V. and Ardebilipour, M. (2004). Spread spectrum code estimation by genetic algorithm. *International Journal of signal processing*, 1(4):301–304.

[6] Back, T., Fogel, D., and Michalewicz, Z. (1997). *Handbook of evolutionary computation*. New York: Oxford Univ. Press and Institute of Physics.

[7] Bäck, T. and Schwefel, H. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation*, 1(1):1–23.

[8] Chen, J., Goldberg, D., Ho, S., and Sastry, K. (2002). Fitness inheritance in multiobjective optimization. In *Genetic and Evolutionary Computation Conference (GECCO'02)*, pages 319–326.

[9] Cvejic, N. (2004). *Algorithms for audio watermarking and steganography*. PhD thesis, PhD thesis, Oulu University of Technology.

[10] Davarynejad, M., Ahn, C. W., Vrancken, J. L. M., van den Berg, J., and Coello, C. A. C. (2010). Evolutionary hidden information detection by granulation-based fitness approximation. *Applied Soft Computing*, 10(3):719–729.

[11] Davarynejad, M., Vrancken, J., van den Berg, J., and Coello Coello, C. (2012). A Fitness Granulation Approach for Large-Scale Structural Design Optimization. In Chiong, R., Weise, T., and Michalewicz, Z., editors, *Variants of Evolutionary Algorithms for Real-World Applications*, pages 245–280. Springer-Verlag, Berlin.

[12] Ducheyne, E., Baets, B. D., and Wulf, R. D. (2003). Is fitness inheritance useful for real-world applications? In *Evolutionary Multi-Criterion Optimization*, pages 31–42.

[13] Gopalan, K. (2003). Audio steganography using bit modification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '03)*, pages 421–424.

[14] Gunn, S. (1998). Support vector machines for classification and regression. Technical report, Technical Report, School of Electronics and Computer Science, University of Southampton.

[15] Haykin, S. (2001). *Communication systems*. John Wiley & Sons, Inc.

[16] Hong, Y., Lee, H., and Tahk, M. (2003). Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks. *Engineering Optimization*, 35(1):91–102.

[17] Hüsken, M., Jin, Y., and Sendhoff, B. (2005). Structure optimization of neural networks for evolutionary design optimization. *Soft Computing*, 9(1):21–28.

[18] Kim, H. and Choi, Y. (2003). A novel echo-hiding scheme with backward and forward kernels. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(8):885–889.

[19] Kirovski, D. and Malvar, H. (2003). Spread-spectrum watermarking of audio signals. *IEEE Transactions on Signal Processing*, 51(4):1020–1033.

[20] Liu, Z., Kobayashi, Y., Sawato, S., and Inoue, A. (2002). A robust audio watermarking method using sine function patterns based on pseudorandom sequences. In *Pacific Rim Workshop on Digital Steganography*, pages 167–173.

[21] Myers, R., Montgomery, D., and Anderson-Cook, C. (2009). *Response surface methodology: process and product optimization using designed experiments*. John Wiley & Sons.

[22] Rastrigin, L. (1974). Extremal control systems. *Theoretical Foundations of engineering cybernetics series*.

[23] Reyes-Sierra, M. and Coello, C. C. (2005). A study of fitness inheritance and approximation techniques for multi-objective particle swarm optimization. In *IEEE Congress on Evolutionary Computation (CEC'05)*, volume 1, pages 65–72.

[24] Reyes-Sierra, M. and Coello, C. C. (2006). Dynamic fitness inheritance proportion for multi-objective particle swarm optimization. In *Genetic and Evolutionary Computation Conference (GECCO'06)*, pages 89–90.

[25] Salami, M. and Hendtlass, T. (2003). A fast evaluation strategy for evolutionary algorithms. *Applied Soft Computing*, 2(3):156–173.

[26] Sedghi, S., Mashhadi, H., and Khademi, M. (2006). Detecting hidden information from a spread spectrum watermarked signal by genetic algorithm. In *IEEE Congress on Evolutionary Computation (CEC'06)*, pages 173–178.

[27] Trivedi, S. and Chandramouli, R. (2005). Secret key estimation in sequential steganography. *IEEE Transactions on Signal Processing*, 53(2):746–757.

[28] Vapnik, V. (2000). *The nature of statistical learning theory*. Springer-Verlag New York Incorporated.

[29] Won, K. and Ray, T. (2005). A framework for design optimization using surrogates. *Engineering optimization*, 37(7):685–703.

[30] Yen, K. and Hanzo, L. (2001). Genetic algorithm assisted joint multiuser symbol detection and fading channel estimation for synchronous cdma systems. *IEEE Journal on Selected Areas in Communications*, 19(6):985–998.

[31] Zadeh, L. (1979). Fuzzy sets and information granularity. In Gupta, M., Ragade, R., and Yager, R., editors, *Advances in Fuzzy Set Theory and Applications, North-Holland, Amsterdam*, Adaptation Learning and Optimization, pages 3–18. Springer Berlin Heidelberg.

4

Accelerating Convergence Towards the Optimal Pareto Front [1]

Abstract

Evolutionary algorithms have been very popular optimization methods for a wide variety of applications. However, in spite of their advantages, their computational cost is still a prohibitive factor in certain real-world applications involving expensive (computationally speaking) fitness function evaluations. In this chapter, we adopt the observation that nature's survival of the fittest is not about exact measures of fitness; rather it is about rankings among competing peers. Thus, by exploiting this natural tolerance for imprecision, we propose here a new, fuzzy granules-based approach for reducing the number of necessary function calls involving time consuming real-world problems. Our proposed approach is compared with respect to the standard NSGA-II, using the *Set Coverage*, *Hypervolume* and *Generational Distance* performance measures. Our results indicate that our proposed approach is a very promising alternative for dealing with multi-objective optimization problems involving expensive fitness function evaluations.

[1] This chapter is based on:

- M. Davarynejad, J. Rezaei, J. Vrancken, J. van den Berg and Carlos A. Coello Coello, "Accelerating Convergence Towards the Optimal Pareto Front", in 2011 Congress on Evolutionary Computation (CEC'2011), New Orleans, pp. 2107-2114, 2011.

4.1 Introduction

Optimization using metaheuristics has become a very popular research topic in the last few years. Real-world problems, however, frequently have two or more (possibly conflicting) objectives that we aim to optimize at the same time. Such problems are called *multi-objective* and have been intensively studied using metaheuristics (particularly, evolutionary algorithms) in the last few years [2].

As opposed to single-objective optimization problems in which we aim to find a single optimum solution, in multi-objective optimization problems (MOOPs) the notion of *optimality* changes, since there is normally no single solution that is the best for all the criteria. The aim in this case is to find a set of solutions for which no objective can be improved without worsening another. This set of solutions is known as the *Pareto optimal set* and their vectors are said to be non-dominated. When plotted in objective function space, these solutions are collectively known as the *Pareto front*.

A wide variety of multi-objective evolutionary algorithms (MOEAs) have been proposed since the inception of this field in the mid-1980s [2, 7]. However, MOEAs are known to be computationally expensive, since they normally require a high number of objective function evaluations in order to produce a reasonably good approximation of the Pareto front of the problem being solved. Nevertheless, relatively little research has been reported so far on the development of techniques that reduce the computational cost of MOEAs (see [25]). This chapter seeks to contribute to this area by introducing a fuzzy granules-based approach for reducing the number of objective function evaluations required by a MOEA.

The remainder of this chapter is organized as follows. Section 4.2 provides some basic multi-objective optimization concepts. The previous related work is discussed in Section 4.3. Section 4.4 presents the approach proposed in this chapter. To illustrate the efficiency of the proposed method, the performance results on ZDT1-6 test problem is presented in Section 6.5. The final section draws conclusions and considers implications for future research.

4.2 Basic Concepts

We are interested in solving problems of the type[2]:

$$\text{minimize } \mathbf{f}(\mathbf{x}) := [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_n(\mathbf{x})] \tag{4.1}$$

subject to:

$$g_i(\mathbf{x}) \leq 0 \quad i = 1, 2, \ldots, q \tag{4.2}$$

$$h_j(\mathbf{x}) = 0 \quad j = 1, 2, \ldots, p \tag{4.3}$$

where $\mathbf{x}$ is a vector of decision variables, $f_i : \mathbf{R}^m \rightarrow \mathbf{R}$, $i = 1, ..., n$ are the objective functions and $g_i, h_j : \mathbf{R}^m \rightarrow \mathbf{R}$, $i = 1, ..., q$, $j = 1, ..., p$ are the constraints of the problem.

To describe the concept of optimality, a few definitions are introduced.

[2] Without loss of generality, we will assume only minimization problems.

Definition 1. Given two vectors $\mathbf{x}, \mathbf{x} \in \mathbf{R}^m$, $\mathbf{x}$ **dominates** $\mathbf{x}$ (denoted by $\mathbf{x} \prec \mathbf{x}$) if $f_i(\mathbf{x}) \leq f_i(\mathbf{x})$ for $i = 1, \ldots, n$, and that $\mathbf{x} \neq \mathbf{x}$.

Definition 2. A vector of decision variables $\mathbf{x} \in \mathcal{X} \subset \mathbf{R}^m$ is **nondominated** with respect to $\mathcal{X}$, if there does not exist another $\mathbf{x}' \in \mathcal{X}$ such that $\mathbf{x}' \prec \mathbf{x}$.

Definition 3. A vector of decision variables $\mathbf{x}^* \in \mathcal{F} \subset \mathbf{R}^m$ ($\mathcal{F}$ is the feasible region) is **Pareto-optimal** if it is nondominated with respect to $\mathcal{F}$.

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:

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

The problem is to find the Pareto optimal set from the set $\mathcal{F}$ of all the decision variable vectors that satisfy (4.2) and (4.3). Note however that in practice, not all the Pareto optimal set is normally desirable (e.g., it may not be desirable to have different solutions that map to the same values in objective function space) or achievable.

4.3 Previous Related Work

Evolutionary algorithms usually require such a large number of function calls that this frequently makes them computationally prohibitive in some real-world applications. When dealing with MOOPs, this issue becomes more critical, because more objectives are involved and this multiplies the computational cost, while also making the search more difficult. For dealing with expensive objective functions, it is relatively common to rely on approximate models that allow us to simplify the representation of real-world complex behaviors. [3] Approximation techniques may estimate each of the individuals' fitness value on the basis of previously observed objective function values of *neighboring* individuals. A wide range of approximation and meta-model techniques have been adopted in combination with evolutionary algorithms, including Kriging [24], artificial neural networks [26], and radial-basis-function networks [18]. Other authors have adopted fitness inheritance [21], cultural algorithms [15] and other fitness function approximation techniques [14] for the same purpose. Next, we will briefly review the most representative work on the use of mechanisms for handling expensive objective functions with MOEAs reported in specialized literature.

Fitness inheritance, a popular class of fitness approximation method, was originally introduced by Smith et al. [27] and is a very simple technique that works as follows: when assigning fitness to an individual, some times the objective function is evaluated as usual, but the rest of the time, the fitness assigned to the individual is the average (or a weighted average) of the fitness of its parents. This fitness assignment scheme operates based on the

[3] This is based on the assumption that approximate models require small computational resources compared to the cost of complex simulations, which is normally the case when considering real-world problems.

assumption of similarity between an offspring and its parents. Clearly, fitness inheritance cannot be applied all the time, since some true fitness function values are required in order to obtain enough information to guide the search. This approach uses a parameter called *inheritance proportion*, which regulates how many times the fitness has to be approximated. Very few authors have reported the use of fitness inheritance in MOOPs. Ducheyne et al. [11] tested the performance of both average and weighted average fitness inheritance approaches and concluded that the usefulness of this technique was limited to cases in which the Pareto front is convex and continuous. Ducheyne et al. [10] also concluded that for non-convex Pareto fronts, fitness inheritance produces a slower convergence to the true Pareto front than when the approach is not adopted. Other authors, however, have successfully applied fitness inheritance to more complicated test problems having non-convex Pareto fronts (see [21]).

Another approach for dealing with expensive objective functions is based on learning and interpolation from representative small datasets of the *true* objective functions values in the desired design space which is known as *functional approximation* [14]. Function approximation methods provide a mapping between design space and objective functions space that may be multi-dimensional. The accuracy of these models depends greatly on the number of sample data points used and their location in the multi-dimensional space. Some examples of this sort of approach are the following: the response surface methodology that uses low-order polynomials and the least square estimations [12, 13, 17] and Gaussian processes (also known as Kriging) that build probability models by exploiting information recorded and use them to estimate the function values of new candidate solutions [3].

Artificial Neural networks (ANNs) can also be used for dealing with expensive objective functions. In fact, ANNs can be considered one of the best approaches to approximate a generic $\mathbf{R}^m \Rightarrow \mathbf{R}^n$ function[4], where m and n represent the number of decision variables and number of objectives, respectively. Although nonlinear interpolation can be used, it is shown that with a number of decision variables higher than 10, the interpolation problem becomes almost not tractable [20]. ANNs are successfully used for building approximate models in a number of complex multiplicative optimization problems. As an example, in [1], a generic supersonic aircraft configuration with two main goals (maximization of the total range of the aircraft and minimization of the ground sonic boom) and a number of buildability and mission constraints (such as structural integrity of the aircraft, take-off and landing field length) is optimized using ANNs to generate inexpensive surrogates. The approximation is used only where this is warranted. Using Latin Hypercube Sampling (LHS), 300 sample data were generate via CFD (Computational Fluid Dynamics) simulation are fitted using a single hidden layer perceptron with sigmoid activation functions to provide a general nonlinear higher fidelity model. In another study, Poloni et al. [20] used a combination of GAs and ANNs with a modified backpropagation algorithm, and a local search method to optimize the design of a sailing yacht fin keel which is a complex design problem in fluid dynamics. The ANN acted as a model for 3D Navier-Stokes simulation of the fin keel while cruising.

For more information on approaches for dealing with expensive objective functions in the context of multi-objective optimization, interested readers should refer to [25].

[4] If they are provided with sufficient structural complexity and a rich training data set.

4.3.1 Final Remarks on Fitness Approximation

In most of the fitness approximation models currently available, the main problem is the lack of sufficient training data and hence the failure to reach a model with sufficient approximation accuracy. Since the evaluation of the original fitness function, in many practical problems, is obtained by some sort of analysis (i.e., fluid mechanics analysis, thermodynamic analysis) that is computationally expensive, the approximate model may be of low fidelity. Furthermore, if the training data does not cover the full domain range, large errors may occur due to extrapolation. Errors may also occur when the set of training points is not sufficiently dense and uniform. Here, we adopt the concept of information granulation as an attempt to address these difficulties.

4.4 Adaptive Fuzzy Fitness Granulation (AFFG)

Granular computing is regarded as the processing of granules of information that are aggregated due to their indistinguishability, similarity, proximity or functionality in some context [29]. It is a vehicle for handling information, as well as a lack of it (uncertainty), at a level of coarseness that can solve problems appropriately and efficiently [5]. In problems with incomplete, uncertain or vague information, the *practical necessity*; and in problems with huge detailed information, the *simplicity* are the main reasons of popularity, respectively, of granular computing. It is widely used in many fields including interval analysis, Dempster-Shafer theory of belief functions, cluster analysis, optimization and problem solving [23], machine learning, bioinformatics, among other fields [19]. The concept of information granulation was proposed by Zadeh [30] (in the context of fuzzy set theory) as a technique by which a class of points (objects) is partitioned into granules. The fuzziness of granules and their attributes is characteristic of the ways by which human concepts and reasoning are formed, organized and manipulated. The concept of a granule is more general than that of a cluster, potentially giving rise to several conceptual structures in various fields of science as well as mathematics.

In the present chapter, with the aim to reducing the computational cost of MOOPs, the concept of information granulation and approximation in the context of rough set theory is studied to exploit the natural tolerance of EAs in fitness function computations. Nature's *survival of the fittest* is not about exact measures of fitness; rather it is about rankings among competing peers. By exploiting this natural tolerance for imprecision and aiming to exploit this uncertainty [16], optimization performance can be preserved by computing fitness only selectively and only to keep this ranking among individuals in a given population.

In the proposed algorithm, a pool of solutions with exact fitness values are maintained. Based on the maximum similarity of a new candidate solution to this pool, the fitness of individuals will be either approximated or calculated explicitly. If a new individual is sufficiently similar to a known fuzzy granule, then that granules' fitness is used instead as a crude estimate. Otherwise, that individual is added to the pool as a new granule. In this fashion, regardless of the competitions' outcome, the fitness of the new individual is always a physically realizable one, even if it is a *crude* estimate and not an exact measurement. The pool size as well as each granules' radius of influence depends on the utility of each granule [6].Furthermore, to prevent the pool from growing too large, pool members are competing for survival and members with lower *life index* are gradually replaced by new

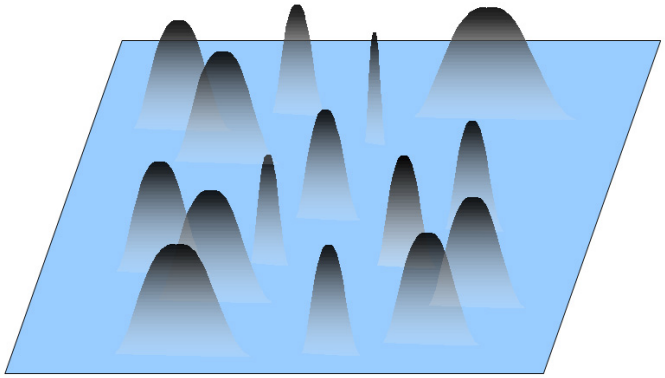

Figure 4.1: A number of gaussian granules with different widths in a 2-D solution space. Once a new individual is sufficiently similar to a granule in the granule pool, then that granules' fitness is used instead as a crude estimate. Otherwise, that individual is added to the pool as a new fuzzy granule. Each granules' radius of influence is determined based on equation (4.7).

granules. By splitting up the pool into two parts, the new granules are given a chance to survive a number of steps [4].

4.4.1 Algorithm's Structure

The preceding section provided a general overview of our approach. Going in more detail now, the algorithm's computation steps are as follows:

Step 1: Create a random parent population $P^1 = \{\mathbf{x}_1^1, \mathbf{x}_2^1, \ldots, \mathbf{x}_j^1, \ldots, \mathbf{x}_t^1\}$ of decision vectors, where, $\mathbf{x}_j^i = \{x_{j,1}^i, x_{j,2}^i, \ldots, x_{j,r}^i, \ldots, x_{j,m}^i\}$ is the jth individual in the ith generation, $x_{j,r}^i$ the rth component of $\mathbf{x}_j^i$, m the number of components of decision vector and t is the population size.

Step 2: Define a multi-set G of fuzzy granules (C_k, σ_k, L_k) according to $G = \{(C_k, \sigma_k, L_k)|C_k \in \mathbf{R}^m, \sigma_k \in \mathbf{R}, L_k \in \mathbf{R}, k = 1, \ldots, N_G\}$. G is initially empty. C_k is an m-dimensional vector of centers, σ_k is the width of membership function (WMF) of the kth fuzzy granule, and L_k is the granule's life index. A number of granules with different widths are shown in Figure 4.1.

Step 3:

- Choose the phenotype of chromosomes, $\mathbf{x}_j^i$, as the center of granules, C_k.
- Rank P^1 and goto **step 8**.

Step 4: Define the membership $\mu_{k,r}$ of each $x_{j,r}^i$ to each granule member by a Gaussian similarity neighborhood function according to

$$\mu_{k,r}\left(x_{j,r}^i\right) = \exp\left(\frac{-\left(x_{j,r}^i - c_{k,r}\right)^2}{\left(\sigma_k\right)^2}\right), \; k = 1, 2, \ldots, N_G \,, \tag{4.4}$$

where N_G is the number of fuzzy granules.

Step 5: Compute the average similarity of the new decision vector $\mathbf{x}_j^i = \{x_{j,1}^i, x_{j,2}^i, \ldots, x_{j,r}^i, \ldots, x_{j,m}^i\}$ to each granule G_k using equation (4.5)

$$\overline{\mu}_{j,k} = \frac{\sum_{r=1}^{m} \mu_{k,r}\left(x_{j,r}^i\right)}{m} \tag{4.5}$$

Step 6: Either calculate the exact fitness value of $\mathbf{x}_j^i$ or estimate it by associating it to one of the granules in the pool in case there is a granule in the pool with similarity value higher than a predefined threshold, i.e.,

$$f\left(\mathbf{x}_j^i\right) = \begin{cases} f(C_k) & \text{if } \max\limits_{k\in\{1,2,\ldots,N_G\}} \{\overline{\mu}_{j,k}\} > \theta^i \,, \\ f\left(\mathbf{x}_j^i\right) & \text{otherwise.} \end{cases} \tag{4.6}$$

where $f(C_x)$ is the fitness function value of the fuzzy granule and $f(\mathbf{x}_j^i)$ is the real fitness calculation of the individual.

Remark: θ^i is a predefined (time-varying) threshold that controls the minimum similarity a solution has to have with a pool member to be approximated. Here, θ^i is considered as a constant value for all simulations, and is set to 0.9. In general, as the population matures steadily, the algorithm needs to be more selective (to calculate the exact fitness more often), suggesting the need for a gradual increase of θ^i. Alternatively, if

$$\max_{k\in\{1,2,\ldots,N_G\}} \{\overline{\mu}_{j,k}\} < \theta^i$$

$\mathbf{x}_j^i$ is chosen as a newly created granule.

Step 7: If the population size is not completed, repeat **Steps 4 to 7**.

Step 8: When termination/evolution control criteria are not met:

- Create offspring population.
- Rank the granule pool.
- Assign σ_k based on equation (4.7).

$$\sigma_k = \sigma_{min} * ((1 - gr_\sigma) + gr_\sigma * \text{rank}(k)) \tag{4.7}$$

  where $\text{rank}(k)$ is the rank of the granule k among the granule set, and $\sigma_{min} \in \mathbb{R}_{>0}$ is a proportional constant that defines the minimum spread of granules. σ_{min} is a problem dependent design parameter.

  Remark: σ_k, the distance measurement parameter that controls the degree of similarity between two individuals, controls the radius of influence of each granule. Instead of drawing the radius directly from the fitness (as in the single-objective optimization case [4]), as objectives are often non-commensurable and conflicting, dominance-based ranking is used. The spread of granules grow as their rank among granule members increases, with a rate of gr_σ. Here, gr_σ is set to 0.1 and $\sigma_{min} \in \{2^n | n \in \mathbb{Z}\}$.

- Goto **step 4**.

4.4.2 Controlling the size of the granule pool and protecting new pool members through speciation

As the evolutionary procedures are applied, it is inevitable that new granules are generated and added to the pool. Depending on the complexity of the problem, the size of this pool can be excessive and become a computational burden itself. To prevent such unnecessary computational effort, a *life index* is introduced in order to appropriately decrease the size of the pool. In other words, it is better to remove granules that do not win new individuals, thereby producing a bias against individuals that have low fitness and were likely produced by a failed mutation attempt. L_k is initially set to 0 and subsequently updated as below,

$$L_k = \begin{cases} L_k + M & \text{if } k = K\,, \\ L_k & otherwise\,, \end{cases} \tag{4.8}$$

where M is the life reward of the granule and K is the index of the winning granule for each individual at generation i. Here, M is set at 1. At each table update, only the N_G granules with the highest L_k index are kept, and the others are discarded. In [5], an example has been provided that illustrates the competitive granule pool update law. Adding a new granule to the granule pool and assigning a life index to it, is a simple way of controlling the size of the granule pool, since the granules with the lowest life index will be removed from the pool. However, it may happen that the new granule is removed, even though it was just inserted into the pool. In order to prevent this, the pool is split into two parts with sizes εN_G and $(1-\varepsilon)N_G$. The first part is a FIFO (First In, First Out) queue and new granules are added to this part. If it grows above εN_G, then the top of the queue is moved to the other part. Removal from the pool takes place only in the $(1-\varepsilon)N_G$ part. In this way, new granules have a good chance to survive a number of steps. In all of the simulations that are conducted here, ε is set to 0.1.

4.5 Numerical results

In order to validate our proposed approach, we adopted the Zitzler-Deb-Thiele (ZDT) test problems [32] and compared our results with respect to those obtained with the standard NSGA-II [8]. The following parameters were adopted for our experiments:

- Population size = 50.
- Crossover rate = 0.9 (SBX).
- Binary tournament selection.
- Mutation rate of $1/m$, m = number of decision variables.
- Distribution indices for crossover η_c and mutation η_m: $\eta_c = 20$ and $\eta_m = 20$.

For assessing our results we adopted three performance measures: (1) Generational Distance (GD) [28], which measures how far the given solutions are, on average, from the true Pareto front, (2) the Hypervolume indicator I_H (also known as Lebesgue measure or S-metric) [31], which measures the volume of the dominated portion of the objective space

Table 4.1: AFFG-NSGA-II utilized parameter values and reference points used for calculating I_H.

Problem	σ_{min}	N_G	Reference point
ZDT1	2^{-4}	100	$[1.1, 3.5]$
ZDT2	2^{-5}	100	$[1.1, 5.0]$
ZDT3	2^{-5}	100	$[1.1, 6.0]$
ZDT4	2^{-6}	100	$[1.1, 140]$
ZDT6	2^{-5}	100	$[1.1, 9.0]$

Table 4.2: Mean and standard deviation of the GD performance measure.

Problem	AFFG-NSGA-II mean, σ	NSGA-II mean, σ
ZDT1	0.010165, 0.005744	0.102095, 0.029859
ZDT2	0.018143, 0.008509	0.716683, 0.365823
ZDT3	0.098656, 0.022421	0.236176, 0.048486
ZDT4	11.160124, 4.239201	20.191547, 11.658247
ZDT6	0.768217, 0.143028	1.328310, 0.224595

which is enclosed by the reference set and (3) Set Coverage (SC) [32], which measures the percentage of solutions from one algorithm that are covered by the solutions of the other. To measure the Hypervolume, a single reference point, $R = r \in \mathbf{R}^m$ was considered in all cases. This point corresponds to the worst value in each dimension of the fronts. The reference values we used here are given in Table 4.1.

The performance measures to assess the results are presented in Tables 4.2, 4.3 and 4.4. The measures are evaluated by conducting 30 independent runs per test problem per algorithm. Each run is restricted to 1,000 fitness function evaluations. Each table displays the average and standard deviation of each of the performance measures.

Table 4.3: Mean and standard deviation of the I_H performance measure.

Problem	AFFG-NSGA-II mean, σ	NSGA-II mean, σ
ZDT1	3.408204, 0.052768	2.689226, 0.164173
ZDT2	4.524421, 0.110119	2.227951, 0.350130
ZDT3	6.106243, 0.198963	4.516725, 0.267211
ZDT4	108.878924, 10.460062	100.619288, 9.466605
ZDT6	3.229885, 0.896935	1.178803, 0.176150

Table 4.4: Mean and standard deviation of the SC performance measure.

Problem	AFFG-NSGA-II mean, σ	NSGA-II-AFFG mean, σ
ZDT1	1.000000, 0.000000	0.000000, 0.000000
ZDT2	1.000000, 0.000000	0.000000, 0.000000
ZDT3	0.995745, 0.023307	0.003401, 0.018630
ZDT4	0.613805, 0.455574	0.324147, 0.427815
ZDT6	0.891819, 0.134759	0.033209, 0.081798

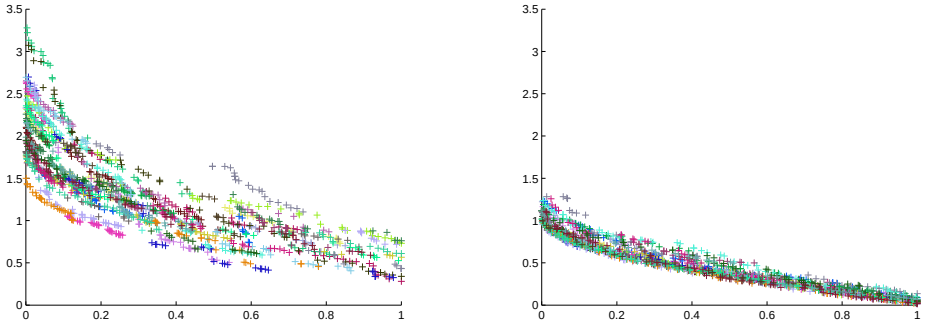

Figure 4.2: 30 independent runs of the NSGA-II (left panel) and AFFG-NSGA-II (right panel) for the ZDT1 test problem using 1,000 real fitness function evaluations.

Figures 4.2 to 4.6 present results of 30 independent runs of the standard NSGA-II and the AFFG-NSGA-II, adopting the test problems ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 (ZDT5 is a binary problem and was, therefore, omitted here), with a budget of only 1,000 fitness function evaluations. Each color corresponds to a single run.

The results clearly show that the proposed AFFG approach outperforms the standard NSGA-II. According to the Wilcoxon rank-sum test, the results of our proposed approach

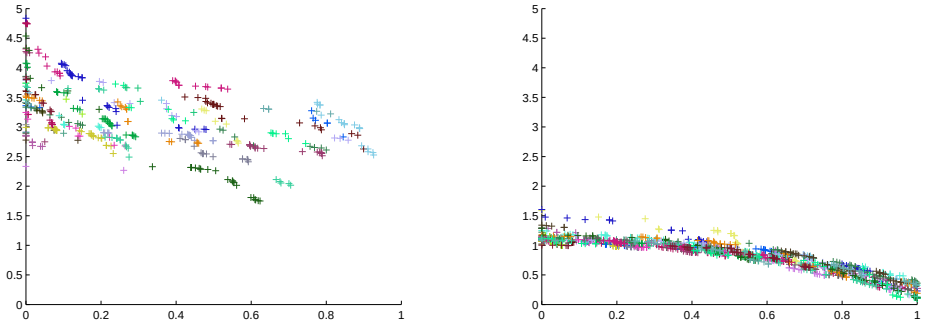

Figure 4.3: 30 independent runs of the NSGA-II (left panel) and AFFG-NSGA-II (right panel) for the ZDT2 test problem using 1,000 real fitness function evaluations.

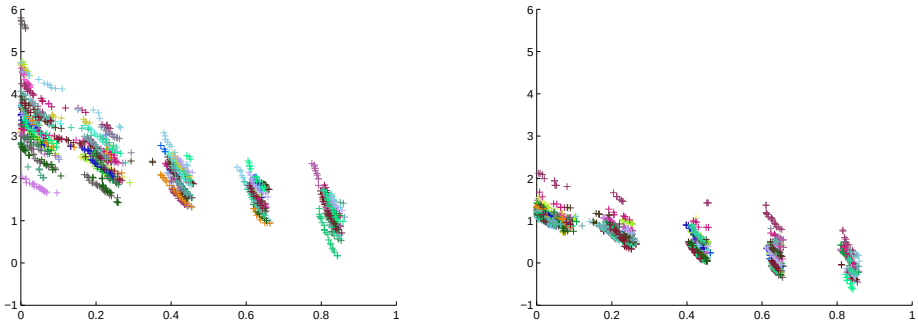

Figure 4.4: 30 independent runs of the NSGA-II (left panel) and AFFG-NSGA-II (right panel) for the ZDT3 test problem using 1,000 real fitness function evaluations.

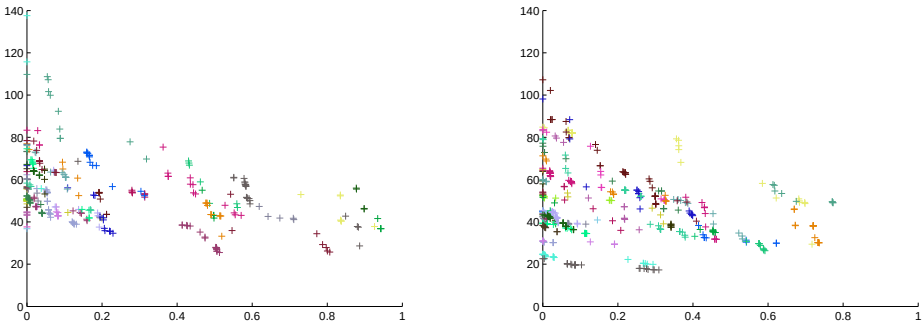

Figure 4.5: 30 independent runs of the NSGA-II (left panel) and AFFG-NSGA-II (right panel) for the ZDT4 test problem using 1,000 real fitness function evaluations.

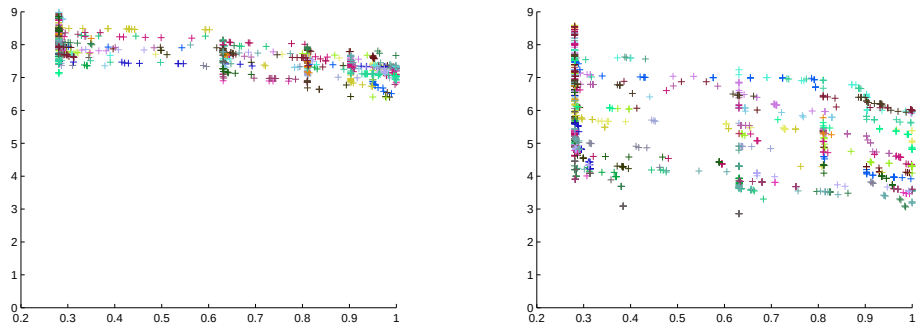

Figure 4.6: 30 independent runs of the NSGA-II (left panel) and AFFG-NSGA-II (right panel) for the ZDT6 test problem using 1,000 real fitness function evaluations.

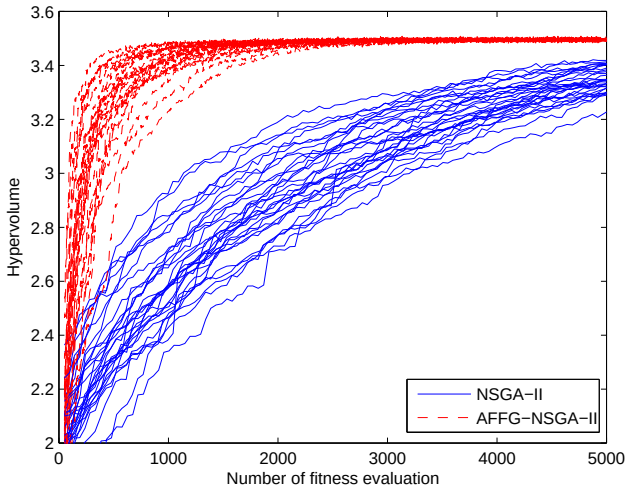

Figure 4.7: Convergence of the hypevolume metric for the ZDT1 problem (30 distinct runs).

are better with a significance level of 5%. To further investigate the convergence speed of the proposed approach, in Figure 4.7, the changes in hypervolume metric is plotted against the number of fitness function evaluations, for the ZDT1 problem.

4.6 Conclusions and Future Work

By combining the concepts of survival of the fittest and fuzzy granulation, which enables a faster convergence without degrading the estimated set of solutions, this chapter presents an approach to speed up convergence towards the Pareto optimal front of multi-objective optimization problems. With the proposed approach, we can exploit the information obtained from our previous objective function evaluations. Our results indicate that the proposed approach is very promising, since it can achieve a faster convergence than the standard NSGA-II in the test problems adopted. However, a more thorough validation is still required (adopting other problems such as the DTLZ test problems [9]). It is also desirable to perform comparisons with respect to other fitness approximation methods such as curve fitting, fitness inheritance and artificial neural networks. As part of our future work, we are interested in studying the effect of the number of granules on the convergence rate. Additionally, in order to further test the robustness of our proposed approach, we want to study its sensitivity to its parameters and its scalability when increasing the number of decision variables and objectives. Adaptively changing θ^i and being more selective as the population matures (to calculate the exact fitness more often), is indeed part of our ongoing research. Finally, we wish to apply our proposed approach to real-world problems in the field of supplier selections [22].

Acknowledgment

This research received funding from the European Community's Seventh Framework Programme within the "Control for Coordination of Distributed Systems" (Con4Coord - FP7/2007-2013 under grant agreement no. INFSO-ICT-223844), the Next Generation Infrastructures Research Program of Delft University of Technology and the Mexican CONACyT Project No. 103570.

References

[1] Alonso, J., LeGresley, P., and Pereyra, V. (2009). Aircraft design optimization. *Mathematics and Computers in Simulation*, 79(6):1948–1958.

[2] Coello Coello, C., Lamont, G., and Van Veldhuizen, D. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, second edition. ISBN 978-0-387-33254-3.

[3] D'Angelo, S. and Minisci, E. (2005). Multi-objective evolutionary optimization of subsonic airfoils by kriging approximation and evolutionary control. In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, volume 2, pages 1262–1267, Edinburg, Scotland.

[4] Davarynejad, M., Ahn, C., Vrancken, J., van den Berg, J., and Coello Coello, C. (2010). Evolutionary hidden information detection by granulation-based fitness approximation. *Applied Soft Computing*, 10(3):719–729.

[5] Davarynejad, M., Akbarzadeh-T, M.-R., and Pariz, N. (2007). A novel general framework for evolutionary optimization: Adaptive fuzzy fitness granulation. In *IEEE Congress on Evolutionary Computation*, pages 951–956. IEEE.

[6] Davarynejad, M., Vrancken, J., van den Berg, J., and Coello Coello, C. (2012). A Fitness Granulation Approach for Large-Scale Structural Design Optimization. In Chiong, R., Weise, T., and Michalewicz, Z., editors, *Variants of Evolutionary Algorithms for Real-World Applications*, pages 245–280. Springer-Verlag, Berlin.

[7] Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK. ISBN 0-471-87339-X.

[8] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

[9] Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2005). Scalable Test Problems for Evolutionary Multiobjective Optimization. In Abraham, A., Jain, L., and Goldberg, R., editors, *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pages 105–145. Springer, USA.

[10] Ducheyne, E., Baets, B. D., and Wulf, R. D. (2008). Fitness inheritance in multiple objective evolutionary algorithms: A test bench and real-world evaluation. *Applied Soft Computing*, 8(1):337–349.

[11] Ducheyne, E., De Baets, B., and De Wulf, R. (2003). Is Fitness Inheritance Useful for Real-World Applications? In Fonseca, C. M., Fleming, P. J., Zitzler, E., Deb, K., and Thiele, L., editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 31–42, Faro, Portugal. Springer. Lecture Notes in Computer Science. Volume 2632.

[12] Goel, T., Haftka, R., Shyy, W., Queipo, N., Vaidyanathan, R., and Tucker, K. (2007). Response surface approximation of pareto optimal front in multi-objective optimization. *Computer Methods in Applied Mechanics and Engineering*, 196(4-6):879–893.

[13] Goel, T., Vaidyanathan, R., Haftka, R., Shyy, W., Queipo, N., and Tucker, K. (2004). Response surface approximation of pareto optimal front in multiobjective optimization. Technical Report 2004-4501, AIAA.

[14] Jin, Y. (2005). A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12.

[15] Landa Becerra, R. and Coello Coello, C. (2006). Solving Hard Multiobjective Optimization Problems Using ε-Constraint with Cultured Differential Evolution. In Runarsson, T. P., Beyer, H.-G., Burke, E., Merelo-Guervós, J. J., Whitley, L. D., and Yao, X., editors, *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference*, pages 543–552. Springer. Lecture Notes in Computer Science Vol. 4193, Reykjavik, Iceland.

[16] Lim, D., Jin, Y., Ong, Y., and Sendhoff, B. (2010). Generalizing surrogate-assisted evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 14(3):329–355.

[17] Madsen, J., Shyy, W., and Haftka, R. (2000). Response surface techniques for diffuser shape optimization. *AIAA journal*, 38(9):1512–1518.

[18] Nakayama, H., Arakawa, M., and Washino, K. (2003). Optimization for black-box objective functions. In Pardalos, P. M., Tseveendorj, I., and Enkhbat, R., editors, *Optimization and Optimal Control*, pages 185–210. World Scientific, Singapore.

[19] Pedrycz, W., Skowron, A., and Kreinovich, V. (2008). *Handbook of granular computing*. Wiley-Interscience New York, NY, USA.

[20] Poloni, C., Giurgevich, A., Onesti, L., and Pediroda, V. (2000). Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):403–420.

[21] Reyes Sierra, M. and Coello Coello, C. (2005). A Study of Fitness Inheritance and Approximation Techniques for Multi-Objective Particle Swarm Optimization. In *2005 IEEE Congress on Evolutionary Computation (CEC'2005)*, volume 1, pages 65–72, Edinburgh, Scotland. IEEE Service Center.

[22] Rezaei, J. and Davoodi, M. (2011). Multi-objective models for lot-sizing with supplier selection. *International Journal of Production Economics*, 130(1):77–86.

[23] Rowhanimanesh, A. and Akbarzadeh-T, M.-R. (2010). Perception-based evolutionary optimization: Outline of a novel approach to optimization and problem solving. In *IEEE International Conference on Systems Man and Cybernetics*, pages 4270–4275. IEEE Press.

[24] Sacks, J., Welch, W., Mitchell, T., and Wynn, H. (1989). Design and analysis of computer experiments. *Statistical science*, 4(4):409–423.

[25] Santana-Quintero, L., Arias Montaño, A., and Coello Coello, C. (2010). A Review of Techniques for Handling Expensive Functions in Evolutionary Multi-Objective Optimization. In Tenne, Y. and Goh, C.-K., editors, *Computational Intelligence in Expensive Optimization Problems*, pages 29–59. Springer, Berlin, Germany. ISBN 978-3-642-10700-9.

[26] Smith, M. (1993). *Neural Networks for Statistical Modeling*. von Nostrand, Reinhold, New York, USA.

[27] Smith, R., Dike, B. A., and Stegmann, S. A. (1995). Fitness inheritance in genetic algorithms. In *SAC '95: Proceedings of the 1995 ACM symposium on Applied computing*, pages 345–350, New York, NY, USA. ACM Press.

[28] Veldhuizen, D. V. (1999). *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio.

[29] Yao, Y. (2001). Information granulation and rough set approximation. *International Journal of Intelligent Systems*, 16(1):87–104.

[30] Zadeh, L. A. (1979). Fuzzy sets and information granularity. In *Advances in Fuzzy Set Theory and Applications*, pages 3–18. North Holland, New York, USA.

[31] Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland.

[32] Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195.

"Nature is my teacher!"

Antoni Gaudi

5

Simulated Big Bounce: A continuous space global optimizer [1]

Abstract

That most complex of systems, Nature has field-tested solutions to many problems, and has furnished the inspiration for many successful problem-solving techniques, including metaheuristics. Metaheuristics applies a search strategy that balances exploration and exploitation in an algorithm-specific way. It is observed that metaheuristic algorithms in practice often find local minima, sometimes of low quality, meaning that the chosen balance is inadequate to the problem at stake. For example, due to an algorithm's search bias, too great an emphasis may be placed on the exploitation of solutions found, while little attention is paid to the further exploration of the search space as a whole. Based on these observations, and inspired by the Big Bounce theory (a cosmological oscillatory model of the Universe), we developed a Simulated Big Bounce (SBB) algorithm that, next to exploitation, applies robust exploration in order to escape local minima. This paper presents the design of this new algorithm and shows the results of a series of comparative experiments in which the performance of SBB on a set of high-dimensional mathematical benchmarks is compared to that of five other popular metaheuristics. The results obtained indicate that the proposed algorithm (*i*) is competitive with (and in most cases surpasses) other population-based optimization algorithms, and (*ii*) substantially decreases the number of fitness function evaluations needed to find equally good solutions. Although SBB has features in common with

[1] This chapter is based on:

- M. Davarynejad and J. van den Berg, "Simulated big bounce: a continuous space global optimizer", Under second-round review at *Information Sciences*.

existing optimization methods, such as particle swarm optimization (PSO), it possesses additional unique features. These owe to the diverse kinetic energy of particles, and enable the algorithm to escape from local minima. Furthermore, the experimental outcomes provide evidence that the characteristic of robust exploration which marks SBB underlies the superior performance observed.

5.1 Introduction

Assume a *search scenario* in a finite continuous search space $E \subset \mathbf{R}^D$ defined as

$$E = \bigotimes_{d=1}^{D} [\mathrm{L}_x^d, \mathrm{U}_x^d], \tag{5.1}$$

with the objective of locating $\mathbf{x}^* \in E$, where $f(\mathbf{x}^*)$ is the extremum of a function $f(\mathbf{x}) : E \to \mathbf{R}$ and where L_x^d and U_x^d are respectively the lower and upper bound of the search domain at dimension d. The optimization problem may be in any engineering, business, medicine, etc. area [8] where it is assumed that samples of the search space and their associated objective function values are the only available information to locate $\mathbf{x}^*$. Without loss of generality, a minimization problem is considered here.

Metaheuristics [26] are strategies to guide the search process. They provide a trade-off between the cost of exploring new possibilities (diversification) and exploiting old certainties (intensification). Many nature-inspired algorithms make use of a certain metaheuristic. The evolution of life on Earth has been such a source of inspiration, the research efforts related to which resulted in the development of evolutionary algorithms [21], a well-known class of population-based stochastic search that falls into the territory of metaheuristics. Not only is a life-supporting planet like the Earth a great inspiration for humans in the designing of metaheuristics, but so is the Universe as a whole.

All population-based algorithms find common ground in (a) the availability of a measure to discriminate solutions and (b) the capability to modify solutions and, using a set of operators, direct the solutions towards promising search regions. Two distinct classes of nature-inspired population-based optimization algorithms are evolutionary algorithms (EAs) and swarm intelligence (SI). A popular member of the former is the class of genetic algorithms (GAs) [23]. The metaphor underlying EAs is natural selection [10], which simulates the natural evolution phenomenon. One possible explanation for the success of genetic algorithms is the building block hypothesis [21, 22].

Swarm intelligence, which refers to the collective problem-solving behavior of multiple agents (exampled by ant colonies [19], bee colonies [24], and the immune system [6]), is also a popular source of inspiration in the design of optimization heuristics. Particle swarm optimization (PSO) [25] and the gravitational search algorithm (GSA) [34] are among popular swarm-intelligence-based algorithms, where the former mimics the social behavior of fish-schooling or bird-flocking, and the later mimics the law of gravity and mass interactions. In swarm intelligence, those solutions which perform best are used to construct a new set of solutions and guide the swarm towards promising regions of the search space.

A competitive optimization algorithm, in order that it quickly and accurately converge on the global optimal solution, has to overcome a number of difficulties, including local optimal solutions, isolation of optimum, non-improving regions of objective landscapes, etc.

This requires a good balance between exploring new search directions (through a "healthy" preservation of diversity) and exploiting the best current search direction [16]. In practice, metaheuristic algorithms have been shown frequently to find local minima, meaning that the chosen balance between exploration and exploitation during the search is inadequate to the problem at hand. For example, because of the search bias of an algorithm, much emphasis may be placed on the exploitation of solutions found, while, after some time, little attention is paid to the further exploration of the whole search space.

Based on these observations, and inspired by the Big Bounce theory (the theory of the development of the Universe), we developed a Simulated Big Bounce (SBB) algorithm that, besides exploitation, applies robust exploration in order to escape local minima. We first provide a general presentation of this new nature-inspired algorithm. We then compare and contrast the SBB with other well-established population-based optimization methods. This second task is achieved by looking at the differences and commonalities from the algorithmic point-of-view as well as an experimental point of view; by comparing their performances on a set of several widely used high-diminution mathematical benchmark problems with various optimization characteristics.

The remainder of this paper is organized as follows. In Section 5.2, GA, PSO and GSA algorithms are briefly summarized. The Big Bounce theory, followed by the key terminology of the new SBB, is provided in Section 5.3. Next, a brief tour of the SBB algorithm is given in Section 5.4. The experimental setup adopted for evaluation and comparison, followed by the major observations made in the experiments, is presented in Section 5.5. The final section draws conclusions and considers implications for future research.

5.2 A review of some popular heuristic algorithms

While SBB shares similarities with other existing population-based algorithms, there are, however, some substantial differences. To help clarify these similarities and differences, and to establish the notation used in SBB, this section outlines the evolutionary algorithms [5, 23], the particle swarm algorithm as proposed in [9] and the gravitational search algorithm [34]. This set of well-established metaheuristics is adopted in order to both specify and to compare the performance of SBB.

5.2.1 Evolutionary algorithms

Evolutionary algorithms (EAs) are stochastic population-based search methods inspired by the scheme of natural selection [22]. EAs includes method of genetic algorithms (GAs) [5, 23], evolution strategies (ESs) and some other derivatives, including genetic programming [27]. Although they are different in some details, they are loosely similar. They all maintain a population of solutions that is randomly generated in the initialization phase. To guide the evolution, by discriminating good solutions from bad ones, a fitness is assigned to every individual. Selection and genetic operators iteratively evolve the population. Recombination (to combine pieces of parental solutions to form offspring) and mutation (whereby random changes are introduced into a solution, reducing the risk of the search algorithm becoming trapped locally) are genetic operators that are often used.

5.2.2 Particle swarm optimization

Particle swarm optimization (PSO) [25] is a successful instance of a nature-inspired algorithm used to solve global optimization problems. A number of advantages have been attributed to PSO, making it a choice candidate for benchmark algorithm. The PSO algorithm is suited to handle nonlinear, nonconvex optimization problems with fast convergence characteristics. PSO is therefore a reasonable choice for comparison as it shares some similarities with SBB proposed in this paper. For that reason, and in order to illustrate the similarities and differences between the PSO and SBB, the PSO is introduced more formally.

In classical PSO, every particle is a solution moving in a D-dimensional search space. A collection of particles is known as a *swarm*. Each particle i has a position $\mathbf{x}_i \in \mathbf{R}^D$, a velocity $\mathbf{v}_i \in \mathbf{R}^D$ and the best position found so far $\mathbf{p}_i \in \mathbf{R}^D$. In contrast to *global best* PSO (GPSO), where each particle shares the details of the best position it has found with the whole swarm, in *local best* PSO (LPSO) each particle informs only k other randomly chosen particles, known as *information links* or *neighbors*. Neighbors are often defined once at the beginning of the search process [4]. While the LPSO typically has a slower convergence compared to that of GPSO, it is less prone to becoming trapped in a local minima. Hence, for complex and high-dimensional problems, LPSO is preferred over GPSO.

Under the global best topology (GPSO) setting, the movement equations of every particle $i \in 1,2,\ldots,S$ are given by expressions (5.2) and (5.3).

$$\mathbf{v}_i = w\mathbf{v}_i + C_1\mathbf{R}_1\left(\mathbf{p_i} - \mathbf{x}_i\right) + C_2\mathbf{R}_2\left(\mathbf{g}_i - \mathbf{x}_i\right), \tag{5.2}$$

$$\mathbf{x}_i = \mathbf{x}_i + \mathbf{v}_i, \tag{5.3}$$

PSO uses two independent random vectors, $\mathbf{R}_1, \mathbf{R}_2$ that are arrays of size D corresponding to each element in vectors $(\mathbf{p}_i - \mathbf{x}_i)$ and $(\mathbf{g}_i - \mathbf{x}_i)$ respectively. $\mathbf{R}_1$ and $\mathbf{R}_2$, with components drawn from a uniform distribution $\mathcal{U}(0,1)$, are used to maintain the population diversity. The scaling constants C_1 and C_2 are known as *learning rates* and they influence the maximum step size a particle can take in a single iteration. These two scaling constants represent, respectively, the confidence of a particle in its best performance and that of the global best. w is a predefined constant representing the confidence of a particle in its own movements. $\mathbf{p}_i$ and $\mathbf{g}_i$ are those positions found which represent personal best and global best respectively. Finally S is the number of particles in the swarm.

To ensure convergence by avoiding *explosion*, Clerc et al. [9] introduce the *constriction factor* and modify the velocity update equation as follows:

$$\mathbf{v}_i = \chi\left(\mathbf{v}_i + C_1\mathbf{R}_1\left(\mathbf{p}_i - \mathbf{x}_i\right) + C_2\mathbf{R}_2\left(\mathbf{g}_i - \mathbf{x}_i\right)\right), \tag{5.4}$$

where $\chi = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|}$ and $\varphi = C_1 + C_2$, $\varphi > 4$.

In the local best setup, the global best position $\mathbf{g}_i$ may be replaced by the local best position $\mathbf{l}_i$ in equations (5.2) and (5.4).

5.2.3 A Brief Tour of the GSA

The Gravitational search algorithm (GSA) [34] is a relatively new technique that has been empirically shown to perform well on many function optimization problems [1, 7, 20, 28, 29, 32, 33, 35, 37]. In its original version, GSA scatters particles in a feasible region of the search space, where they interact with each other under Newtons gravitational force and move within the search area, seeking for an optimal solution.

In GSA, each candidate solution is a particle with a mass M_i. The attractive gravitational force governs the movement of the particles in the search space. The quantity of the resulting force is determined by Newtons gravitational law, which together with particle's current position $\mathbf{x}_i(t)$, particles updated position $\mathbf{x}_i(t+1)$ is determined.

In the original GSA [34], the mass of particles is assigned as follows:

$$M_i = \frac{m_i}{\sum_{j=1}^{S} m_j}, i = 1,2,\ldots,S \tag{5.5}$$

where

$$m_i = \frac{f(\mathbf{x}_i) - \max_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j)}{\min_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j) - \max_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j)}, \tag{5.6}$$

and S is the number of particles. The resulting gravitational force acting on particle i in direction d is determined using

$$F_i^d = \sum_{j\in Kbest} r_j F_{ij}^d, \tag{5.7}$$

where $Kbest$ is a set of particles with the highest mass, $r_j \sim \mathcal{U}(0,1)$ and F_{ij}^d is the gravitational force exerted by particle j on particle i. To provide a better exploration in the early iterations $|Kbest|$ is set at S in the beginning, i.e. $K_0 = S$; however the exploration must be decreased gradually. Therefore choosing a decremented function for $|Kbest|$ increases the exploitation of the algorithm when the number of iterations increases.

The force exerted by particle j acting on particle i is defined as:

$$F_{ij}^d = G_t \frac{M_i \times M_j}{R_{ij} + \varepsilon} \left(x_j^d - x_i^d\right) \tag{5.8}$$

where R_{ij} is Euclidian distance between particles i and j, and G_t, the gravitational constant initialized at G_0 is determined using equation (5.9) as:

$$G_t = G_0 e^{-\alpha \frac{t}{T}}, \tag{5.9}$$

where α is algorithmic tuning parameter and T is the maximum number of cycles.

The equations of motion of particle i are as follows.

$$\mathbf{v}_i(t+1) = \mathbf{R} \times \mathbf{v}_i(t) + \frac{\mathbf{F}_i}{M_i}.\Delta t, \tag{5.10}$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1).\Delta t, \tag{5.11}$$

where $\Delta t = 1$, $\mathbf{R}$ is an array of size D drawn from a uniform distribution $\mathcal{U}(0,1)$.

5.3 Simulated Big Bounce (SBB)

5.3.1 Elements of Big-Bang (BB) Theory: Back to the beginning

Life on Earth is thought to be a consequence of the evolution of the cosmos. Our understanding of the origin of life on Earth is connected to a proper understanding of the origin of the Universe. Cosmology is the study of the Universe as a whole, its origin and evolution, from early times down to the future [39]. The Big Bang, known as the standard cosmological model of the Universe, is a widely accepted explanation for the origin of the Universe.

According to this model, the Universe was born about 13.7 billion years ago in an unimaginably hot and dense single point known as the Big Bang singularity[2], and started to expand. In time, as the Universe cooled down to the point where particles and atoms could form, gravity pulled particles together and organized them into cosmic structures - the planets, stars, galaxies and associated objects we observe today. The observed evolution of organized structures, clumps of galaxies and stars, from a nearly homogenous hot gas in thermal equilibrium[3], is the inspirational base for the SBB. Even though there is no "sufficiently broad" basis for the theoretical justification of BB theory, there are several facts which are very much in favour of the validity of this theory.

The discovery of the cosmic microwave background (CMB) in 1965 by two American radio astronomers performing a routine noise analysis of a large antenna, led, some 10 years later, to the establishment of the hot Big Bang model, a revolution in the winding path of cosmology. The CMB (Figure 5.1), on average nearly 2.725 K (-270.425C) [18] deviation from a perfect blackbody spectrum, is generally interpreted as a thermal relic of an earlier phase of the Universe, e.g. left over from the Big Bang. The BB model states that the Universe, in its early phase, started hot and dense. As the evolution of the Universe proceeds, it is expanded, thinned out and cooled. When the temperature dropped below 3,000 K, the photons and matter that were tightly coupled in the early Universe were released in the late Universe, travelled freely, and made up the CMB we see today.

The second piece of evidence in favour of BB is known as Hubble's Law of red-shift, which states that distant galaxies are receding from us with a speed proportional to their distance from us. Red-shift, interpreted as a Doppler phenomenon, is the increase in the measured wavelength of radiation from remote space matter, caused by their recession (Figure 5.2). Red-shift was first discovered by Slipher and Hubble and suggests that the expansion of the Universe traces back to a time when all the matter was closer together than it is today.

5.3.2 The Big Bounce Theory explains the Universe preceding the Big Bang and after.

According to Einstein's general theory of relativity, the Big Bang began in a singularity, a state where general relativity is invalid. To evade the singularity problem, the possibility of

[2] A state in which a certain parameter increases without bound is referred to as a singularity.

[3] In thermal equilibrium, matter and radiation continuously undergo reactions and are able, based on the Special Theory of Relativity, to freely convert back and forth.

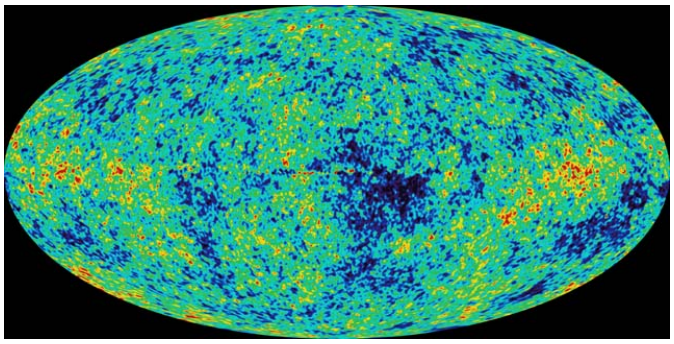

Figure 5.1: Cosmic microwave background (CMB). CMB anisotropies as observed by Wilkinson Microwave Anisotropy Probe (WMAP). Different colors representing different temperature with fluctuations of the order of μK, too small to have been observed in the 1960s. [Figure courtesy NASA/WMAP Science Team]

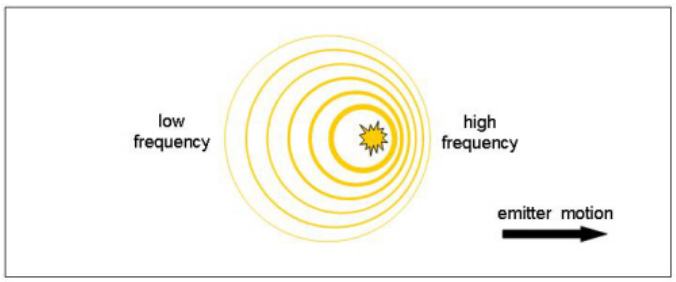

Figure 5.2: Doppler effect (The observed radiation of light sources moving toward us shifts to shorter wavelengths (blue-shifted) and those moving away from us have their wavelength increased (red-shifted)). Some studies shown that the Hubble's Law do not need to be related to the notion of Doppler redshifts of the observed light from receding Galaxies. As the distance between the light source and the observer expands, the wavelength of emitted light expanders by the same proportion and that's the source of the redshift.

a cyclic Universe is proposed in [38] and [3], in which the Universe undergoes a "bounce" at a minimum non-zero volume into an expanding one (Figure 5.3).

The Big Bounce theory extends the history of the Universe even further back beyond the Big Bang. It asserts that the initial high-density state arose as a result of the collapse of an existing Universe under the attractive force of gravity. Gravity switches and becomes repulsive as a result of high density [3]. An expansion phase, with all cosmic energy in mass format, begins. The expansion phase of each cycle proceeds by converting the mass resolved in the preceding phase back into energy. Gravity dilutes the accelerated expansion phase and initiates the reconversion of the energy back into mass. Contraction occurs when the mass that fuels the cosmic expansion is nearly depleted.

Simulated Big Bounce (SBB) is an optimization algorithm that is inspired by the Big Bounce Theory of the evolution of the Universe. This oscillation of the Universe (a cy-

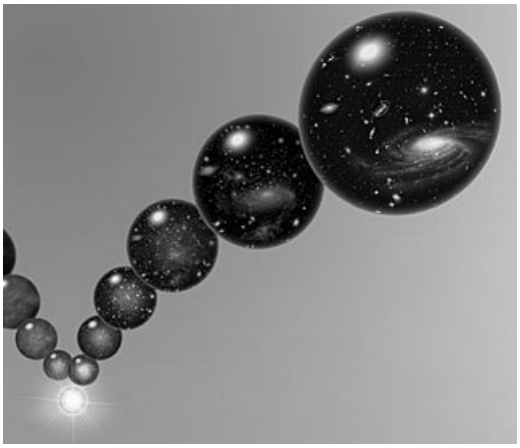

Figure 5.3: Big Bounce is a cosmological oscillatory model of Universe with an endless sequence of cycles of extraction and contraction[3].

cle that begins with a Big Bang, followed by expansion, contraction and a Big Crunch) is the metaphor that endows the SBB, a search algorithm that is explained in detail in Section 5.3.3, and more extensively in Section 5.4.

5.3.3 SBB algorithm

SBB models each candidate solution as a particle, each with a position, velocity and mass. Each particle travels in the D-dimensional search space E under the effect of a dynamically changing force field. The presence of the force field and the particle's kinetic energy, induced by its velocity and mass, together with its current position, determine the particle's next position in the search space. Both a) the diverse kinetic energy accumulated by the moving particles and b) the oscillatory force acting on them, enable the swarm to escape from the local minima, thereby making SBB a successful optimizer in complex multi-modal optimization problems. A description of the working principles of the SBB algorithm follows.

In SBB the Universe undergoes two phases, the *Small-bang* and the *Small-crunch*.

Small-bang (Explosion phase)

In this phase, also known as the *Explosion* phase, all particles are exposed to a repulsive force $\mathbf{F}_i^r \in \mathbf{R}^D$, $i = 1,2,\ldots,S$, whose direction opposes the BaryCenter $\mathbf{x}_{BC}$, tending to preserve diversity in the swarm. The BaryCenter $\mathbf{x}_{BC}$ is the center of mass of all the particles. The external applied force affects the speed and the direction of motion of the particles and changes their position in the search space. In this phase, the particles are scattered throughout the search space. This phase will be selected by a small probability p_{SB}.

Small-crunch (Contraction phase)

In this phase, also known as the *Contraction* phase, the particles are getting closer to the best performing particle, $\mathbf{x}_{DM}$, that is, the particle with the highest mass, U_M. The best performing particle is referred to as "Dark Matter" with position $\mathbf{x}_{DM}$. The probability of performing Small-crunch is p_{SC}.

The search begins with a force applied to the static particles uniformly distributed within the search space E. This force may bring the particles closer to the Dark Matter (the best solution found so far) or it may spread the particles out in order to preserve the diversity of the swarm, depending on whether the Universe is in an explosion or contraction phase. The fitness of the particles in the next time step defines the mass of each particle. A superior solution is analogous to a particle with a high mass, and a poor solution represents a particle with a small mass. Particles with high mass resist position change more than those with low mass, and tend to have a higher impact on the Barycenter X_{BC}, thereby sharing their good features (better fitness) with low quality solutions. Poor solutions with low mass are, however, subject to a massive change which may raise the quality of those solutions found.

Particles with a diverse range of mass and different resistances to position change prevent premature convergence and stagnation. This is the first built-in mechanism for preserving the diversity of solutions throughout the course of the search process in SBB. The second mechanism engineered to prevent premature convergence is the Small-bang phase, where particles are scattered throughout the search space by a force acting on them with a random strength, and which tends to preserve diversity in the swarm. Moreover, the randomness of the strength of the force also boosts the particles exploration of the search space. To ensure the convergence of the swarm, both the p_{SB} and this upper-bound for the randomized strength of the force decreases with lapse of time.

5.4 A Brief Tour of the SBB Algorithm

The search begins by scattering a predefined number S of particles randomly distributed in the search space E, with positions $\mathbf{x}_i$, $i = 1,\ldots,S$ and velocities $\mathbf{v}_i$, $i = 1,\ldots,S$ all set at zero. To guide the population in the search space, some measure of discrimination is needed, referred here to as the "fitness" $f(\mathbf{x}_i)$ of each candidate solution. The fitness value associated with each point in the search space also defines the mass M_i, $i = 1,\ldots,S$ of a particle in that position.

In the next step, depending on whether the search is in the contraction or expansion phase, an attractive or repulsive force is applied to the particles. For the sake of notation simplification, we will bundle the lower and upper bounds of decision variables into the arrays

$$\mathbf{L}_x := (L_x^1|\ldots|L_x^d|\ldots|L_x^D)$$

and

$$\mathbf{U}_x := (U_x^1|\ldots|U_x^d|\ldots|U_x^D)$$

respectively. If the search is in the contraction phase, the external force, with a random strength bounded between $[0,\, C_1\,(\mathbf{U}_x - \mathbf{L}_x)]$, has a direction towards the Dark Matter $\mathbf{x}_{DM}$. Otherwise the strength is bounded between $[0,\, C_2\,(\mathbf{U}_x - \mathbf{L}_x)]$ with a direction opposing the Barycenter X_{BC} of the swarm (5.18). With probability p_{SB} the swarm undergoes expansion,

and with probability $p_{\mathrm{SC}} = 1 - p_{\mathrm{SB}}$ the swarm undergoes contraction. In the contraction phase, the particles are getting closer to the best performing solution, i.e., that with the highest mass. This oscillatory process preserves the diversity of the particles in the swarm throughout the whole search process.

In order to achieve a solid compromise between exploration and exploitation, different strategies are embedded in the search process of SBB. The parameter p_{SB} controls the frequency of the expansion phase the Universe undergoes. With lapse of time, in order to provide enough iterations for particles to converge, p_{SB} decreases according to (5.12). The same holds for the parameters that control the strength of the attractive and repulsive forces exerted on particles, C_1 (5.13) and C_2 (5.14).

$$p_{\mathrm{SB}}(t+1) = p_{\mathrm{SB}}(t) * \frac{T-t}{T} \tag{5.12}$$

$$C_1(t+1) = C_1(t) * \frac{T-t}{T} \tag{5.13}$$

$$C_2(t+1) = C_2(t) * \frac{T-t}{T} \tag{5.14}$$

The behavior of particles under the attractive $\mathbf{F}_i^a \in \mathbf{R}^D$ and repulsive $\mathbf{F}_i^r \in \mathbf{R}^D$ forces and their associated mass M_i and velocity $\mathbf{v}_i$ is governed according to (5.15) and (5.16), also known as "equations of motion". The kinetic energy of the particle is a form of "memory", giving it the possibility to "steer" its movement under the influence of both its past behavior and the current external forces acting on it. The force field and the kinetic energy induced from the particles's velocity and mass, together with its current position, determine the particle's next position in the search space.

$$\mathbf{v}_i(t+1) = \rho * \mathbf{v}_i(t) + \frac{\mathbf{F}_i}{M_i}.\Delta t \tag{5.15}$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1).\Delta t \tag{5.16}$$

where Δt is the time interval and is usually set as one unit; $\mathbf{v}_i = (v_{i1}, \ldots, v_{id}, \ldots, v_{iD})$, $v_{id} \in [-U_v^d, U_v^d]$, and U_v^d is a designated velocity upper bound and is determined according to (5.17) where C_3 is velocity coefficient. In reality, as a particle accelerates under the effect of the force field, it is also attenuated by a friction force. In equation (5.15), ρ is friction coefficient.

$$U_v^d = C_3(U_x^d - L_x^d) \tag{5.17}$$

The termination criterion for iterations is determined according to whether a maximum number of cycles T or a designated value of the fitness has been reached.

The barycenter (center of mass) that is $\mathbf{x}_{BC}$ is given by:

$$\mathbf{x}_{BC} = \frac{\sum_{i=1}^{S} M_i \mathbf{x}_i}{\sum_{i=1}^{S} M_i} \tag{5.18}$$

5.4.1 Mass Assignment

Based on the fitness of each particle, a mass is assigned to it in the range of $[L_M, U_M]$. g, the function that maps the non-negative fitness values $f(\mathbf{x}_i)$ to the mass $g : \mathbf{R} \to \mathbf{R}$, $f(\mathbf{x}_i) \mapsto g(f(\mathbf{x}_i))$, $\forall \mathbf{x}_i \in E$ may be any arbitrary (and possibly time varying) monotonically nondecreasing, usually real-valued function. We take g as a linear time-invariant strictly increasing function according to (5.19).

$$M_i = g(f(\mathbf{x}_i)) = \\ L_M + (U_M - L_M) \frac{f(\mathbf{x}_i) - \max\limits_{j \in \{1,\ldots,S\}} f(\mathbf{x}_j)}{\min\limits_{j \in \{1,\ldots,S\}} f(\mathbf{x}_j) - \max\limits_{j \in \{1,\ldots,S\}} f(\mathbf{x}_j)}. \tag{5.19}$$

Although the mass assignment in SBB (equation 5.19) and GSA (equation 5.5) share similarities, they are different in definition. The mass assignment in GSA has two steps. In the first, particles are assigned a mass m_i according to their fitness (equation 5.6). Then M_i is calculated by normalizing m_i. The sum of m_i's at every iteration changes. Now, consider that the position of particle k, $\mathbf{x}_k$ is the same in two consecutive iterations. Since the sum of m_i's over all particles in these two iterations is most likely different, the mass assigned to particle k changes surprisingly, although its position $\mathbf{x}_k$ has not changed.

The basic steps of the SBB algorithm are summarized in pseudocode shown in Algorithm 5.1.

We also observed that in GSA, a change in the number of particles changes the mass assigned to them. This is a result of an increase in the denominator of the equation (5.5). This increase of the denominator smoothes out the difference between the mass of the particles, making them, in absolute terms, more equal in exerting an attractive force, and equally resistant to change in their position as a result of the applied gravitational force. The swarm, then, can be seen as a group of particles with a loosely uniform mass distribution. Under the Newtonian gravitational force, this brings the particles closer to the center of the swarm, resulting in an increase in the density of swarm. As a result, they move towards the center of the search space. As we will see in Section 5.5, GSA has a poor performance when the optimal solution is not at the center of the search space, an observation that may be explained by the "center-seeking bias" of the GSA [14]. Neither of these two observations are to be found in the mass assignment procedure of SBB.

5.5 Experimental Setup and Numerical results

Typical mathematical optimization problems are crafted to hold the optima at, or near, the origin of the search space. This is the case with most of the test functions adopted in this study. When comparing nature-inspired metaheuristic algorithms, a symmetric search space is possibly misleading [12, 14]. It may positively influence the performance of certain metaheuristics. To prevent predisposed results in favour of *origin-seeking* bias of studied optimization algorithms, an "asymmetric search space" has been adopted here.

In addition, most real-world optimization problems of today are encountered in environments that undergo continual change. These are referred to as "dynamic optimization problems". When the global optimal solution changes, the population members have to

Algorithm 5.1 Pseudocode of Simulated Big Bounce (SBB)

Input: Search space E, Maximum number of cycles T, Population size S, Fitness function f, Friction coefficient ρ, Force coefficients C_1 and C_2, Velocity coefficient C_3, p_{SB}, L_M, U_M.

Calculate U_v^d ▷ According to (5.17)
Initialize particles location, $X = (\mathbf{x}_1, \ldots, \mathbf{x}_S)^T$
Fitness calculation
$t \leftarrow 1$ ▷ t is the number of iterations
$M_i \leftarrow 1$, $i = 1, \ldots, S$
$\mathbf{v_i} \leftarrow 0$, $i = 1, \ldots, S$
procedure EXPLOSION($\mathbf{x}$) ▷ For Exploration
    Update $\mathbf{x}_{BC}$ according to (5.18)
    Update repulsive force $\mathbf{F}_i^r$, $\forall i = 1, \ldots, S$
end procedure
procedure CONTRACTION($\mathbf{x}$) ▷ For Exploitation
    $\mathbf{x}_{DM} \leftarrow \arg\min_{\mathbf{x}_i} M_i(\mathbf{x}_i)$
    Update attractive force $\mathbf{F}_i^t$, $\forall i = 1, \ldots, S$
end procedure
while $t < T$ **do**
    Update $\mathbf{v}_i$, $\forall i = 1, \ldots, S$ ▷ According to (5.15)
    Update $\mathbf{x}_i$, $\forall i = 1, \ldots, S$ ▷ According to (5.16)
    With probability p_{SB} do **Explosion** else do **Contraction**
    Fitness calculation
    Update M_i, $\forall i = 1, \ldots, S$ ▷ According to (5.19)
    $t{+}{+}$ ▷ t is the number of iterations
    Update C_1 and C_2 ▷ According to (5.13) and (5.13)
    Update p_{SB} ▷ According to (5.12)
end while

Output: x^* and $f(x^*)$

move along an extended path, often with many local optima. To test the sensitivity of the different algorithms studied on the search initialization, as well as their ability to move from the initial search space to more promising regions, "asymmetric initialization" is adopted as well [2]. E.g. the algorithms are deliberately initialized in a portion of the search space that does not include the global optimum. A notable example of algorithms suffering from insufficient generation of offspring outside of a given initial population is GA with Unimodal Normal Distribution Crossover (UNDX) [31].

When comparing the effectiveness of different optimization methods, a standard performance measure is the best fitness value a certain algorithm can attain within a predefined number of function evaluations. This is based on the assumption that fitness evaluation is the dominant factor in the overall computational cost. Such an assumption is usually valid for complex optimization tasks of interest in real-world problems [13, 15] where the budget allocated, in terms of time and resources, is limited. For that reason, the algorithms in this work are compared with one another solely based on fitness values that can be achieved within a predefined number of function evaluations, and not based on their algorithm-related

computation time.

From the test beds studied in [34, 40], those with varying dimensions are used in this study in addition to those studied in [30]. The test beds, along with their characteristics, are listed in Table 5.1[4].

In Table 5.1, D is the dimension of the search space. The optimal solution $f(\mathbf{x}^*)$ for all the adopted test functions is located at $[0, \ldots, 0]$, with the exception of Dixon-Price, the optimal solution for which is located at $2^{-\frac{2^d-2}{2^d}}$ for $d = 1, 2, \ldots, D$, as well as the Levy and Rosenbrock, where the optimal solution sits at $[1, \ldots, 1]$. Of the 14 adopted test studies, half are unimodal, while the other half are multimodal. The set contains five separable[5] and nine non-separable functions.

Table 5.1: Test problems used in the experiments. U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable

Function name	Characteristic	Search space	Initialization range
Ackley	MN	$[-10, 30]^D$	$[20, 30]^D$
Dixon-Price	UN	$[-10, 2]^D$	$[-10, -8]^D$
Griewank	MN	$[-100, 600]^D$	$[500, 600]^D$
Levy	MN	$[-10, 2]^D$	$[-10, -8]^D$
Penalized1	MN	$[-50, 10]^D$	$[-50, -40]^D$
Penalized2	MN	$[-50, 10]^D$	$[-50, -40]^D$
Quartic	US	$[-1.28, 0.28]^D$	$[-1.28, -1.18]^D$
Rastrigin	MS	$[-1.12, 5.12]^D$	$[4.12, 5.12]^D$
Rosenbrock	MN	$[-10, 50]^D$	$[40, 50]^D$
Schwefel	MS	$[-512, 512]^D$	$[-512, -412]^D$
Schwefel P2.22	UN	$[-2, 10]^D$	$[8, 10]^D$
Schwefel P1.2	UN	$[-50, 100]^D$	$[50, 100]^D$
Schwefel P2.21	US	$[-50, 100]^D$	$[50, 100]^D$
Sphere	US	$[-10, 100]^D$	$[90, 100]^D$
Step	US	$[-100, 10]^D$	$[-100, -80]^D$

As the studied optimization techniques are stochastic in nature, 30 independent runs of each of the algorithms were executed, and the median and the best of the fitness values are reported in Tables 5.2 and 5.3 along with the results of the Mann-Whitney U-test for statistical significance.

5.5.1 Parameter Settings

In all experiments described in this section, the common parameters used in each algorithm, such as population size and the number of fitness evaluations, are the same. Unless

[4]Note that, in [34], the Rosenbrock function (also known as the Banana problem) is treated as a unimodal test function when D is set at 30, while it is multimodal when the problem dimensions are more than three [17, 36].

[5]A separable function can be decomposed into D one-dimensional functions.

otherwise mentioned, the population size is set at 50 and the maximum number of fitness evaluations is set at 100,000.

GA settings

A real-coded genetic algorithm with roulette wheel selection was used. Two different types of crossover, namely single point (GA1) and uniform (GAU), were tried. In both cases, the crossover rate was set at 1 and the mutation rate at 0.05. In order to ensure that the best-performing chromosomes always intact, the elitism parameter was set at 2.

PSO settings

For PSO, local learning was used, where each particle is influenced by its topological neighbors. Both standard PSO and PSO with a constriction factor were considered. For standard PSO (PSO-local) the inertial constant w was set at 0.9 and linearly decreased to 0.4. Both the cognitive constant C_1 and social constant C_2 were set at 2.0. For PSO with a constriction factor (PSO-cf-local), w was set at 0.729 and $C_1 = C_2 = 1.49$. This is equivalent to setting $\chi = 0.729$ and $\varphi = 4.1$. k, the number of neighbors was set at 3.

GSA settings

Apart from the common parameters, namely, the number of the population and the maximum number of the fitness evaluation, in the case of GSA, k was set at the population size S and was linearly decreased to 1. The gravitational constant G_0 was set at 100 and α was set at 20.

SBB settings

In the case of the SBB algorithm, The friction coefficient ρ was set at .1, $C_1 = 1$, $C_2 = 10 * C_1 = 10$, $C_3 = .2$, $p_{SB} = .1$ and L_M and U_M were set at 1 and 10 respectively.

5.5.2 Results

The *medians* of the best-of-run of the studied algorithms are normalized and shown in Table 5.2. This normalization is obtained by comparison with the best performing algorithm on each benchmark. Therefore, the best solution found usually has a value of 1.00 (this holds for all the functions except the last). As the results of Table 5.2 demonstrate, the median performance of the SBB, when averaged over 30 independent simulation runs, is competitive to the studied optimization algorithms. In this table, asterisk symbols are used to denote solutions that are, statistically, significantly better than their contenders. The test caries out using MannWhitney U test (a non-parametric statistical test that is often interpreted as a comparison of medians).

As shown in Figures (5.4), in roughly most of the cases SBB began with a steep convergence, and in 6 cases, located better solutions substantially faster than other algorithms studied. PSO with a constriction factor is the second best performing algorithm, and finds significantly better solutions in 5 of the benchmarks. GSA is significantly superior to the

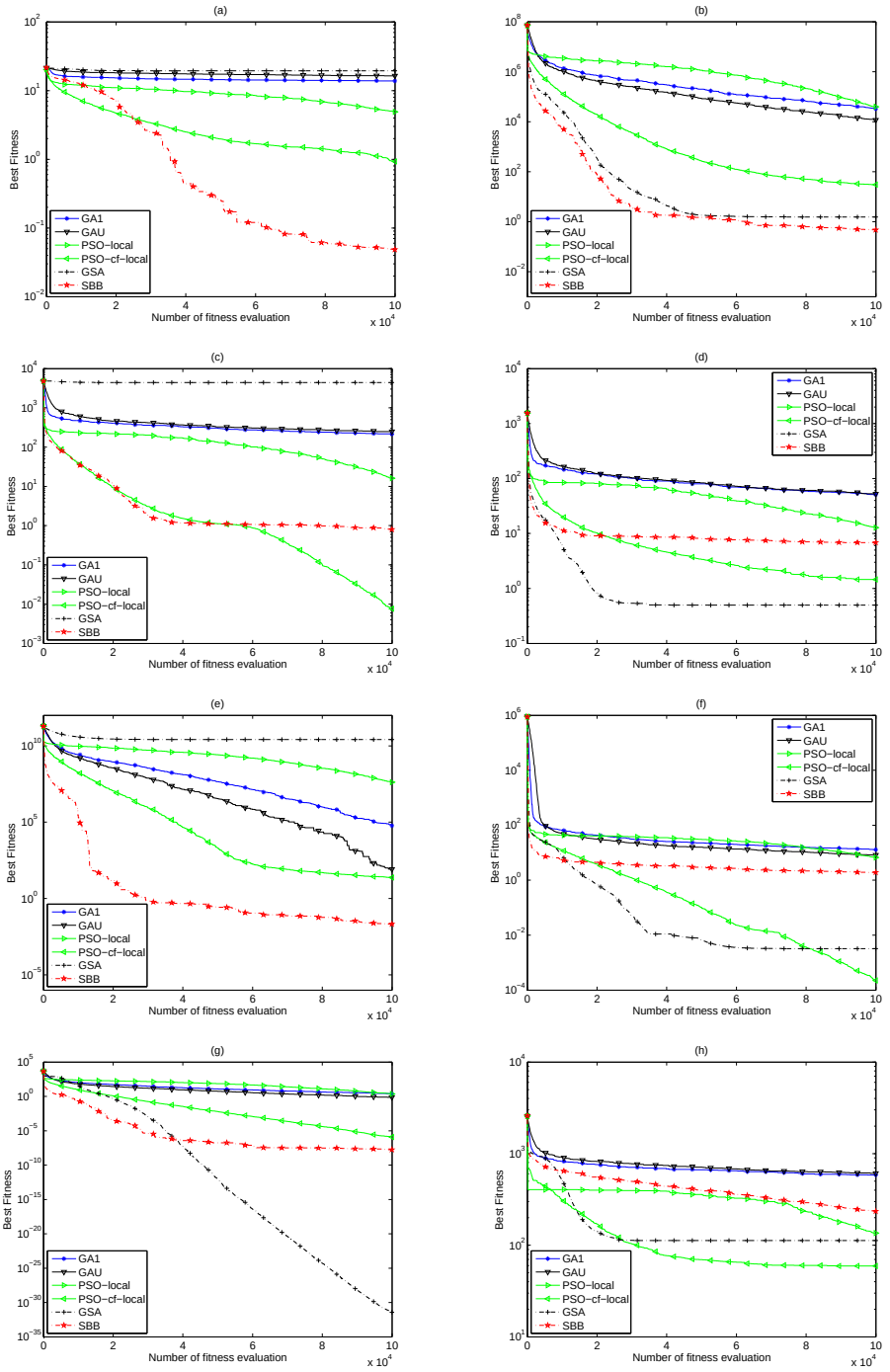

Figure 5.4: Median of 30 independent runs on a) Ackley, b) Dixon-Price, c) Griewank, d) Levy, e) Penalty 1, f) Penalty 2, g) Quartic, h) Rastrigin,

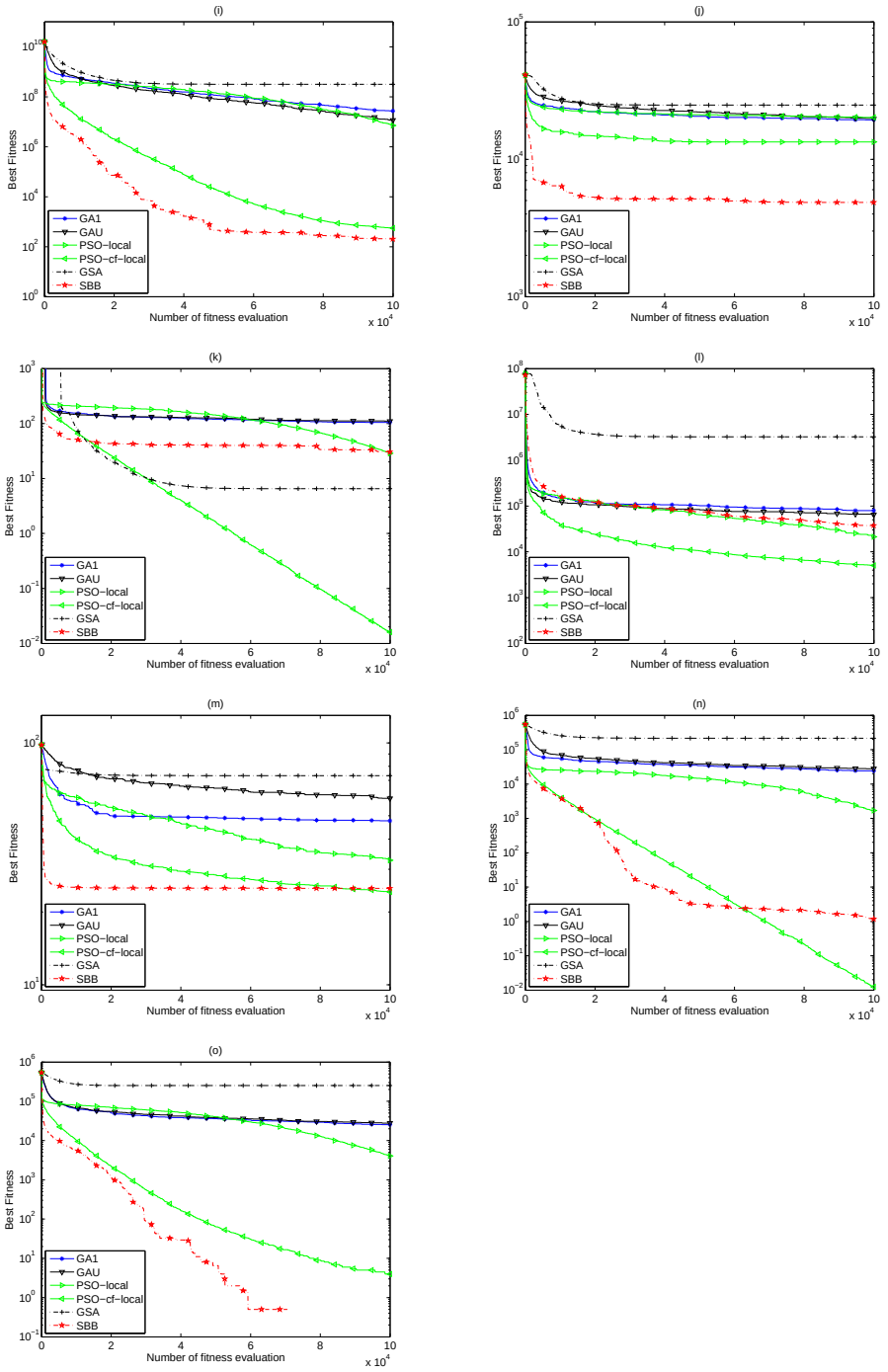

Figure 5.4: (con't) i) Rosenbrock, j) Schwefel, k) Schwefel P2.22, l) Schwefel P1.2, m) Schwefel P1.21, n) Sphere, o) Step.

others in only 2 of the cases. As shown in our other studies [12, 14], the poor mass assignment procedure in GSA is, in part, responsible for its poor performance. Its performance is flattened out dramatically in almost every case tested. With the *Step* function, SBB reaches upto a true minimum with value zero, resulting in a "NaN" score for this algorithm in Table 5.2 and an "Inf" of infinity for the others.

We also provide a comparison to find, amongst those metaheuristics studied, which is best at finding the most fitted solution when multiple runs are made (Table 5.3). SBB places first by performing best in 6 out of 14 benchmarks (the Step function is excluded). PSO-cf placed second, performing best in 5 benchmarks, followed by GSA, which performed best in three benchmarks. With regards to the Step function, the optimal solution, 0, is attainable under a wide range of design values. Although both PSO-cf-local and SBB found the optimal solution, SBB had a better convergence characteristics (Figure 5.4.o).

It is notable that, in this work, the settings are taken from the original works. However, it must be noted that changing the algorithmic parameter settings and stopping criteria, the benchmark functions, and even the grading criteria, may change the results and conclusions. In spite of these caveats, these preliminary results are a promising indication of the success of the proposed SBB on a wide range of optimization problems.

Table 5.2: Median of normalized optimization results of the competing optimization algorithms on the studied benchmark functions averaged over 30 independent simulation runs.

Function name	GA1	GAU	PSO-Local	PSO-cf-local	GSA	SBB
Ackley	283.91	337.04	101.29	19.21	399.72	**1***
Dixon-Price	6.97E4	2.49E4	8.43E4	64.21	3.28	**1***
Griewank	2.83E4	3.23E4	2.10E3	**1***	5.75E5	104.01
Levy	102.15	105.21	25.66	2.92	**1***	13.63
Penalty 1	2.86E6	3.97E3	1.99E9	1.1E3	1.23E12	**1***
Penalty 2	5.77E4	3.66E4	2.96E4	**1**	14.32	8.45E3
Quartic	6.43E31	1.93E31	6.07E31	3.32E25	**1***	4.51E23
Rastrigin	9.86	10.28	2.29	**1***	1.89	3.95
Rosenbrock	1.34E5	5.63E4	3.51E4	2.73	1.57E6	**1***
Schwefel	3.99	4.05	2.76	4.15	5.11	**1***
Schwefel P2.22	6.51E3	6.86E3	1.79E3	**1***	407.66	1.90E3
Schwefel P1.2	15.84	12.93	4.26	**1***	647.02	7.431
Schwefel P1.21	1.97	2.43	1.35	**1**	3.03	1.03
Sphere	1.96E6	2.23E6	1.36E5	**1***	1.73E7	94.10
Step	Inf	Inf	Inf	Inf	Inf	NaN*

Table 5.3: Best of normalized optimization results of the competing optimization algorithms on the studied benchmark functions averaged over 30 independent simulation runs.

Function name	GA1	GAU	PSO-Local	PSO-cf-local	GSA	SBB
Ackley	5.6E3	6.89E3	1.78E3	6.62	8.53E3	**1**
Dixon-Price	8.78E4	1.90E4	1.35E5	36.86	3.78	**1**
Griewank	5.24E4	6.50E4	4.09E4	**1**	1.57E6	18.57
Levy	3.90E17	4.20E17	1.25E17	1.97E13	**1**	1.82E13
Penalty 1	1.54E5	1.08E4	3.75E9	3.55E3	5.80E12	**1**
Penalty 2	7.94E17	4.63E17	2.75E17	1.73E12	**1**	1.34E15
Quartic	4.18E31	1.53E31	8.93E31	2.03E25	**1**	7.14E19
Rastrigin	17.29	17.51	2.33	**1**	2.71	2.70
Rosenbrock	1.98E5	2.94E5	1.76E5	20.33	1.12E7	**1**
Schwefel	19.50	21.42	13.56	20.71	27.22	**1**
Schwefel P2.22	7.96E3	8.45E3	2.23E3	**1**	3.14	957.28
Schwefel P1.2	20.73	14.48	3.49	**1**	79.87	8.13
Schwefel P1.21	8.27E3	9.37E3	5.56E3	3.26E3	1.38E4	**1**
Sphere	3.67E6	3.59E6	1.97E5	**1**	4.38E7	2.3694
Step	Inf	Inf	Inf	NaN	Inf	NaN

5.6 A comparative discussion on evolutionary computing paradigms vs. SBB

In the previous sections, the comparative results of GA, PSO, GSA and SBB were presented. In this section, a thorough comparative analysis is offered.

Two very important aspects of evolutionary computing paradigms are exploration (also known as diversification; that is, the ability to search for new possibilities) and exploitation (also known as intensification; that is, the ability to find better solutions in the neighborhood of good solutions found so far). Different metaheuristics employ different operators to balance exploration and exploitation throughout the optimization process. The design of the SBB scheme is strongly influenced by the idea that exploration should be continued in a robust way in order that it be able to escape from local minima discovered in all phases of the search. The key feature distinguishing SBB from prior work is its unique machinery to maintaining a healthy diversity of particles.

In GAs, the mutation operator is responsible for maintaining the population diversity by modifying a randomly selected part of each (new) population member. In SBB meanwhile, the repulsive force applied to particles preserves diversity. Unlike PSO, where solutions are mostly clumped together for information flow control (in order to prevent premature convergence, although their topology is not necessarily fixed), both SBB and EAs solutions do not necessarily have a built-in tendency to cluster solutions. They already possess certain mechanisms for maintaining population diversity. PSO and GSA control population diversity by

inserting random parameters when updating solutions. SBB maintains population diversity though two mechanisms: a) it diversifies particle's by updating their mass as the optimization process progresses; b) it applies repulsive external force to particles with a bounded random strength acting in a direction opposite to the center of gravity of the swarm.

The fitness-based probabilistic selection scheme used in GAs does not exist in SBB. The selection mechanism is the main difference between GA and other metaheuristics adopted in this study. Moreover, in contrast to EAs where solutions "die" at the end of each generation, in PSO, GSA and SBB, solutions survive through the course of the optimization process, providing a substantial source of information for the population searching for the global optimum.

In PSO, each particle's position vector is decoded by a function of the particles current position, the best solution found by itself so far, and the swarms best solution. In SBB, a new solution vector is calculated, depending on the state of the Universe, using the best solution found so far, or using the position of the Barycenter. In SBB, the speed of DM is set at zero. This improves the exploitation of the area surrounding the best solution so far, while the population continues to enjoy robust exploration due to the particles of light mass with small inertia. In this way, the information belonging to the fittest member of the population is distributed among other members without jeopardizing exploration. In PSO, the inertia of all particles is the same, while in the case of SBB, the inertia of each particle is directly connected to its mass, and is thus not unique to the entire population.

In GSA, a new position vector is calculated using Newtons attractive gravitational force, while in SBB, the force applied may be attractive or repulsive with a stochastic variation in strength. This is in complete contrast to GSA. In early iterations, in the case of GSA, objects are further apart compared to later iterations. As a result of the gravitational force, particles are pulled closer together and so the gravitational force strengthens as time elapses. This leads to shallow exploration in early search iterations and strong exploitation in the final search stages, which making GSA susceptible to be trapped in a local optimum [12]. In GSA, the mass assigned to a particle changes as a result of an increase or decrease in swarm size. As swarm size increases, the mass assigned to particles becomes, in absolute terms, loosely equal. Particles thus exert an equal force on each other, and demonstrate an equivalent resistance to position change, a phenomenon that is considered undesirable. In contrast, the dispersed mass-assignment procedure in SBB helps maintaining exploration-exploitation balance in both the early and the late stages of the search process.

5.7 Conclusions and Future Work

Existing scientific literature evidences that the information sharing mechanism of stochastic population-based metaheuristics accounts for their robust search capabilities as well as their small inclination to deceit. Nonetheless, for complex high-dimensional optimization problems, they have been shown to be susceptible to premature convergence, due to a poor balance between exploration and exploitation. The extent to which information is shared between agents, and the mechanism designed to update the agents, should work together to balance the exploration-exploitation trade-off.

A new metaheuristic, based on the theory of Big Bounce, was therefore designed, one that uses robust exploration in order to escape from local extrema. The proposed Simulated

Big Bounce (SBB) showed a capability for the robust exploration of the search space. We hypothesize that this is due to the built-in "diversification of mass" which supports both continued exploitation and robust exploration. The algorithm has been tested by comparing its performance with the performance of five other variations of metaheuristics. The results obtained show that the SBB's performance is competitive with existing metaheuristics.

We would like to further test this hypothesis in future research. In addition, we observe that the proposed optimization strategy can be easily extended to multi-objective optimization problems. Further studies may also focus on sensitivity analysis and parameter studies, and their relationships with the convergence rate of the algorithm. Hybridization with other popular algorithms, especially with local search algorithms such as simulated annealing, could also be potentially fruitful.

Much of the computational complexity involved in the use of population-based optimization tools is due to the fitness function evaluation, which may be either very difficult to define, or very expensive computationally. A popular solution to this challenge is to replace the expensive fitness evaluation step with an approximate model [11]. In our future work, the performance of SBB when combined with this type of solution will be studied.

A probabilistic selection of DM may further improve the search process, a suggestion for future work.

References

[1] Al-Zubaidi, S., Ghani, J., and Haron, C. (2013). Optimization of cutting conditions for end milling of ti6al4v alloy by using a gravitational search algorithm (gsa). *Meccanica*, pages 1–15.

[2] Angeline, P. (1998). Using selection to improve particle swarm optimization. In *International Conference on Evolutionary Computation*, pages 84–89.

[3] Bojowald, M. (2008). Follow the bouncing universe. *Scientific American*, 299(4):44–51.

[4] Bratton, D. and Kennedy, J. (2007). Defining a standard for particle swarm optimization. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 120–127. IEEE.

[5] Bremermann, H. (1962). Optimization through evolution and recombination. *Self-organizing systems*, pages 93–106.

[6] Castro, L. D. and Zuben, F. V. (2002). Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, 6(3):239–251.

[7] Chatterjee, A., Mahanti, G., and Pathak, N. (2010). Comparative performance of gravitational search algorithm and modified particle swarm optimization algorithm for synthesis of thinned scanned concentric ring array antenna. *Progress In Electromagnetics Research B*, 25:331–348.

[8] Chiong, R., Weise, T., and Michalewicz, Z. (2012). *Variants of evolutionary algorithms for real-world applications*. Springer.

[9] Clerc, M. and Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73.

[10] Dasgupta, D. and Michalewicz, Z. (1997). *Evolutionary algorithms in engineering applications*. Springer-Verlag, Berlin.

[11] Davarynejad, M., Ahn, C., Vrancken, J., van den Berg, J., and Coello Coello, C. (2010). Evolutionary hidden information detection by granulation-based fitness approximation. *Applied Soft Computing*, 10(3):719–729.

[12] Davarynejad, M., Forghany, Z., and van den Berg, J. (2012a). Mass-dispersed gravitational search algorithm for gene regulatory network model parameter identification. In *Simulated Evolution and Learning (SEAL'12)*, pages 62–72.

[13] Davarynejad, M., Rezaei, J., Vrancken, J., van den Berg, J., and Coello, C. C. (2011). Accelerating convergence towards the optimal pareto front. In *IEEE Congress on Evolutionary Computation (CEC'11)*, pages 2107–2114.

[14] Davarynejad, M., van den Berg, J., and Rezaei, J. (2014). Evaluating center-seeking and initialization bias: The case of particle swarm and gravitational search algorithms. *Information Sciences*, 278:802–821.

[15] Davarynejad, M., Vrancken, J., van den Berg, J., and Coello Coello, C. (2012b). A Fitness Granulation Approach for Large-Scale Structural Design Optimization. In Chiong, R., Weise, T., and Michalewicz, Z., editors, *Variants of Evolutionary Algorithms for Real-World Applications*, pages 245–280. Springer-Verlag, Berlin.

[16] Deb, K. (2012). Advances in evolutionary multi-objective optimization. In *Search Based Software Engineering*, pages 1–26. Springer.

[17] Deb, K., Anand, A., and Joshi, D. (2002). A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary computation*, 10(4):371–395.

[18] Dodelson, S. (2003). *Modern cosmology*. Academic Press.

[19] Dorigo, M. and Caro, G. D. (1999). Ant colony optimization: a new meta-heuristic. In *IEEE Congress on Evolutionary Computation (CEC'99)*, pages 1470–1477.

[20] Duman, S., Güvenç, U., and Yörükeren, N. (2010). Gravitational search algorithm for economic dispatch with valve-point effects. *International Review of Electrical Engineering (IREE)*, 5(6).

[21] Goldberg, D. (1994). Genetic and evolutionary algorithms come of age. *Communications of the ACM*, 37(3):113–119.

[22] Goldberg, D. (2002). *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers.

[23] Holland, J. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI.

[24] Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39(3):459–471.

[25] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948.

[26] Kochenberger, G. (2003). *Handbook of metaheuristics*. Springer.

[27] Koza, J. and Poli, R. (2005). Genetic programming. In Edmund, K. and Kendall, G., editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, pages 127–164.

[28] Li, C. and Zhou, J. (2011). Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm. *Energy Conversion and Management*, 52(1):374–381.

[29] Lopez-Molina, C., Bustince, H., Fernandez, J., Couto, P., and Baets, B. D. (2010). A gravitational approach to edge detection based on triangular norms. *Pattern Recognition*, 43(11):3730–3741.

[30] Mariani, V. and Coelho, L. (2011). A hybrid shuffled complex evolution approach with pattern search for unconstrained optimization. *Mathematics and Computers in Simulation*.

[31] Ono, I., Kita, H., and Kobayashi, S. (1999). A robust real-coded genetic algorithm using unimodal normal distribution crossover augmented by uniform crossover: Effects of self-adaptation of crossover probabilities. In *Genetic and Evolutionary Computation (GECCO'99)*, pages 496–503.

[32] Precup, R., David, R., Petriu, E., Preitl, S., and Paul, A. (2011). Gravitational search algorithm-based tuning of fuzzy control systems with a reduced parametric sensitivity. *Soft Computing in Industrial Applications*, pages 141–150.

[33] Precup, R., David, R., Petriu, E., Preitl, S., and Radac, M. (2013). Fuzzy logic-based adaptive gravitational search algorithm for optimal tuning of fuzzy-controlled servo systems. *Control Theory & Applications, IET*, 7(1):99–107.

[34] Rashedi, E., Nezamabadi-Pour, H., and Saryazdi, S. (2009). Gsa: a gravitational search algorithm. *Information Sciences*, 179(13):2232–2248.

[35] Rashedi, E., Nezamabadi-Pour, H., and Saryazdi, S. (2011). Filter modeling using gravitational search algorithm. *Engineering Applications of Artificial Intelligence*, 24(1):117–122.

[36] Shang, Y., , and Qiu, Y. (2006). A note on the extended rosenbrock function. *Evolutionary Computation*, 14(1):119–126.

[37] Shaw, B., Mukherjee, V., and Ghoshal, S. (2012). A novel opposition-based gravitational search algorithm for combined economic and emission dispatch problems of power systems. *International Journal of Electrical Power & Energy Systems*, 35(1):21–33.

[38] Steinhardt, P. and Turok, N. (2002). A cyclic model of the universe. *Science*, 296(5572):1436–1439.

[39] Weinberg, S. (1972). Gravitation and cosmology: principles and applications of the general theory of relativity.

[40] Yao, X., Liu, Y., and Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102.

"You cannot improve what you cannot measure."

lord kelvin

6

Evaluating Center-Seeking and Initialization Bias: The case of Particle Swarm and Gravitational Search Algorithms [1]

Abstract

Complex optimization problems that cannot be solved using exhaustive search require efficient search metaheuristics to find optimal solutions. In practice, metaheuristics suffer from various types of search bias, the understanding of which is of crucial importance, as it is directly pertinent to the problem of making the best possible selection of solvers. In this paper, two metrics are introduced: one for measuring center-seeking bias (CSB) and one for initialization region bias (IRB). The former is based on "ξ-center offset", an alternative to "center offset", which is a common but inadequate approach to analyzing the center-seeking behavior of algorithms, as will be shown. The latter is proposed on the grounds of "region

[1] This chapter is based on:

- M. Davarynejad, J. van den Berg, J. Rezaei, "Evaluating Center-Seeking and Initialization Bias: The case of Particle Swarm and Gravitational Search Algorithms", *Information Sciences*, 278:(802-821), 2014.
- Z. Forghany, M. Davarynejad, B.E. Snaar-Jagalska, "Gene Regulatory Network Model Identification Using Artificial Bee Colony and Swarm Intelligence", in 2012 Congress on Evolutionary Computation (CEC'12), Brisbane, Australia, pp. 949954, 2012 .
- M. Davarynejad, Z. Forghany, J. van den Berg, "Mass-Dispersed Gravitational Search Algorithm for Gene Regulatory Network Model Parameter Identification", in 2012 Simulated Evolution And Learning (SEAL'12), Volume 7673 of Lecture Notes in Computer Science. pp. 62-72, 2012 .

scaling". The introduced metrics are used to evaluate the bias of three algorithms while running on a test bed of optimization problems having their optimal solution at, or near, the center of the search space. The most prominent finding of this paper is considerable CSB and IRB in the gravitational search algorithm (GSA). In addition, a partial solution to the center-seeking and initialization region bias of GSA is proposed by introducing a "mass-dispersed" version of GSA, mdGSA. mdGSA promotes the global search capability of GSA. Its performance is verified using the same mathematical optimization problem, next to a gene regulatory network parameter identification problem. The results of these experiments demonstrate the capabilities of mdGSA in solving real-world optimization problems.

6.1 Introduction

Consider a *search scenario* in a finite continuous search space $E \subset X$ defined by

$$E = \bigotimes_{d=1}^{D} [\mathrm{L}_x^d, \mathrm{U}_x^d], \tag{6.1}$$

with the objective of locating $\mathbf{x}^* \in E$, where $f(\mathbf{x}^*)$ is the extremum of a function $f(\mathbf{x}) : E \to \mathbf{R}$, and where L_x^d and U_x^d are respectively the lower and upper bound of the search domain at dimension d. Optimization problems are to be found in such diverse arenas as engineering, business, medicine, etc. [5]. Here we assume that the only information available to the search for the optimal design variable is a measure to discriminate solutions, i.e., for any point $\mathbf{x} \in E$, the associated objective (fitness) value $f(\mathbf{x})$ is assumed to be the only information available to locate $\mathbf{x}^*$. Without loss of generality, a minimization problem is considered.

In contrast to exhaustive search which looks into every entry in the search space, *metaheuristics* [22] are strategies that guide the search process iteratively, in many cases by making a trade-off between exploration and exploitation. This is an important notion when it comes to allocating scarce resources to the exploration of new possibilities and the exploitation of old certainties.

The evolution of life on earth, which has been the original inspiration for many types of metaheuristics, has resulted in the family of population-based stochastic search algorithms termed "evolutionary algorithms". Common to all *population-based* metaheuristics are (*i*) a measure to discriminate solutions, and (*ii*) a set of mechanisms to modify solutions by various operators.

There are two distinct classes of nature-inspired population-based optimization algorithms that are of our interest: evolutionary algorithms (EA), and swarm intelligence (SI)-based algorithms. Some popular members of the former class are genetic algorithm (GA) [24] and differential evolution (DE) [45, 56]. Successful instances of swarm intelligence-based algorithms are particle swarm optimization (PSO) [27] and the gravitational search algorithm (GSA) [46].

Studying the properties of these algorithms, it turns out that some population-based optimization techniques suffer from a specific search bias [11, 35]: they tend to perform best when the optimum is located *at or near the center of the search space*. General purpose optimizers are those which make no assumption on the problem at stake. Consequently, if we want to compare the quality of the solutions found by a set of metaheuristics for a

series of benchmark problems with optimal solution near the center of the search space, the comparison becomes unfair.

To remedy this unfairness, the so-called *center offset* (CO) [36] approach was proposed which changes the borders of the search space in such a way that the optimal solution is no longer located in the center of the search space. Basically, the CO approach changes the search space of the original problem by reducing it on one side and expanding it at the other. When comparing a set of algorithms *qualitatively*, the comparison is valid since interference tends to be reduced when all the contenders are submitted to the same set of benchmarks, no matter if the shifting has introduced some degree of increase/decrease in the complexity of the search. Our goal, here, is to supplement the comparison by developing *quantitative* measures that can assist the observer in evaluation of the "degree" of CSB of a certain search algorithm. Quantitative measures are succinct and are the preferred disclosure form, not only for a) a comparison of the degree of CSB in a set of search algorithms, but also when the task is b) to examine if a single search algorithm has any CSB at all.

On the basis of these observations, we decided to examine generic methods for evaluating the search bias of different algorithms. In this paper, we limit ourselves to two metrics; one for measuring center-seeking bias, and one for initialization bias. These metrics are used to evaluate the behavioral bias of several algorithms related to swarm optimization and gravitational search.

The remainder of this paper is organized as follows. Section 6.2.1 elaborates on center offset and its assumptions, and presents an alternative. Section 6.2.2 presents a metric to both measure and compare the center-seeking bias of optimization algorithms. A metric to measure initialization region bias is then presented in Section 6.3. In Section 6.4.1 and 6.4.2 PSO and GSA are briefly summarized. The mass assignment in GSA is analyzed and challenged in Section 6.4.3, and an alternative is proposed. The experimental setup adopted for the evaluation and comparison, followed by the major observations derived form the experiment, are presented in Section 6.5. Section 6.6 presents discussions and provides a framework that enables a fair comparison of optimization heuristics. The last section highlights conclusions and provides suggestions for future research.

6.2 A metric for measuring center-seeking bias

6.2.1 Understanding the assumptions underlying center offset

According to the *No Free Lunch* theorem [60], all learning systems will expose equal performance over all possible cost functions. This implies that, in order to efficiently solve an optimization problem, they should be tailored to the salient problem-specific characteristics. Where there is no available information on the problem at hand, as with various real-world applications, some search biases known to us are not often of service. Such biases include center-seeking (CS) behavior and initialization region bias (IRB), the foci of this study.

When comparing nature-inspired metaheuristic algorithms, a symmetric search space can be misleading when the optimal solution is located at, or near, the center of the search space. In such a case, one must account for CS behavior in order to draw valid conclusions from an experiment [8]. One attempt to deal with CS bias is called *center offset* (CO). This is a common approach to negating the centrist bias of an optimization algorithm [3]. The

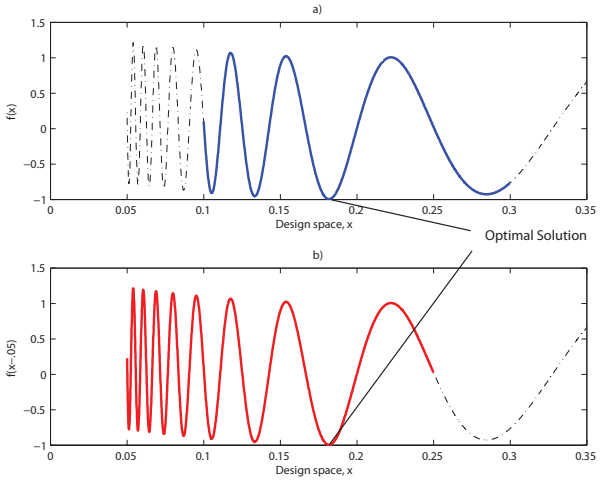

Figure 6.1: Change in complexity as a result of CO. a) The original function, b) The transformed function.

underlying assumption of CO is that the complexity of a problem does not change as a result of moving the optimal solution from the center of the search space; this is an assumption that is discussed in greater detail below.

When applying CO, the optimization problem $f(\mathbf{x})$ is changed to $f(\mathbf{x}-\mathbf{C})$ where $\mathbf{C}$ is the location of the new center. CO is equivalent to expanding the search space from one side, for each dimension d, and to shrinking it on the other side, without changing the distance $\|U_x^d - L_x^d\|$ between the lower bound L_x^d and the upper bound U_x^d. When the objective of a test is to measure the search bias of an algorithm, CO is not an adequate approach. This is because a change in the complexity of a problem is not explicitly controlled: without any additional information, the complexity of the problem might increased, decreased, or even remained the same. As a consequence, any observed difference in the performance of an algorithm cannot, to any degree of certainty, be associated with the CS bias of the algorithm; it may also have been caused by an (unknown) change in the problem complexity.

Figure 6.1 shows an example of an increase in problem complexity (due to an increase in the number of local optimal solutions) as a result of shifting the search window when the objective is to locate the minimum of the following function:

$$f(x) = 10\,(x-0.2)^2 + sin(\frac{\pi}{x}), 0.1 \leq x \leq 0.3. \tag{6.2}$$

In this case, due to an increased problem complexity, the average performance of *any* metaheuristic is expected to deteriorate whether or not the algorithm possesses CS bias. Consequently no hypothesis can be made on the CS behavior of an optimization algorithm.

Assuming we know that the problem complexity decreases, some decision making around CS behavior becomes possible. If a certain algorithm shows a better performance, one can conclude that this algorithm has no observable CS bias, since we would otherwise

have observed a deterioration in its performance, i.e., a deterioration in the best found fitness during optimization. In this study, the ξ-CO approach is introduced to remove the uncertainties on change in problem complexity.

In ξ-CO, the search space is downsized asymmetrically, as a result of which the problem complexity always decreases and the algorithm is expected to locate a near optimal solution more quickly and with greater precision if there is no center-seeking bias. This makes it possible to test the hypotheses on CS behavior of an optimization algorithm considering a benchmark problem with (*i*) a symmetric search space and (*ii*) the optimal solution near the center of the search space. Let us assume that $L_x^d = -U_x^d$ and that $U_x^d > 0$, as is the case for most of the problems studied here. In ξ-CO, the search space is downsized asymmetrically by modifying the lower bound of the search space L_x^d according to

$$L_x^d = L_x^d + \frac{\xi}{100} \times \frac{||U_x^d - L_x^d||}{2}, \tag{6.3}$$

where $\xi \in [\xi_L, \xi_U]$ is the percentage of downsizing the search space and where ξ_L and ξ_U are the predefined lower and upper bound of ξ, respectively.

Observe that CO is only worthwhile for optimization problems the support of which extends outside of the initial boundary. With the proposed ξ-CO approach this shortcoming does not arise.

6.2.2 A metric for center-seeking bias

After identifying ξ-CO as an appropriate approach for analyzing CS behavior of optimization heuristics, there is still a need to quantify the observations on the CS bias behavior, i.e., a metric is needed. By executing a series of runs when gradually increasing, with a predefined step size of ξ_s, the percentage ξ of downsizing the search space from a lower limit ξ_L to an upper limit ξ_U, one can measure the best-fitness each optimization algorithm can attain. Because randomly chosen initializations affect the outcome, experiments under equal conditions are usually repeated several times, say r_f time, yielding a data of the form $\left(\xi, f_r^{\xi}\right) \in \mathbf{R}^2$ when $r \in [1, r_f]$. Based on these observations, an estimation of best-of-run $\hat{f}^{\xi}$ as function of ξ

$$\hat{f}^{\xi} = \mathrm{CSB}_{\xi_s}^{\xi_L - \xi_U} . \xi + c_1 \tag{6.4}$$

can be calculated, where $\mathrm{CSB}_{\xi_s}^{\xi_L - \xi_U}$ is the slope of the regression line. The slope $\mathrm{CSB}_{\xi_s}^{\xi_L - \xi_U}$ has been selected as the metric to analyze CS behavior of optimization heuristics. For minimization problems, in case $\mathrm{CSB}_{\xi_s}^{\xi_L - \xi_U} \geq 0$, the best fitness found increases, implying the presence of CS bias behavior (because the quality of the solutions found does not improve when complexity is reduced). Similarly, in case $\mathrm{CSB}_{\xi_s}^{\xi_L - \xi_U} < 0$, the best fitness found decreases, implying that there is no observable CS bias behavior.

In special cases, ξ may be changed from its lower limit to its upper limit without intermediate steps. In this case the corresponding metric is referred to as $\mathrm{CSB}^{\xi_L, \xi_U}$. Comparing $\mathrm{CSB}^{\xi_L, \xi_U}$ and $\mathrm{CSB}_{\xi_s}^{\xi_L - \xi_U}$, the latter has a greater generalization ability and is the preferred metric for comparing algorithms under study.

Note, finally, that the proposed metric is not restricted to situations where the search space is downsized according to ξ-CO. The original CO approach may still be used, namely,

in cases where the problem at hand is well-known and the problem complexity due to center offset remains unchanged. Under such circumstances, the metric $\text{CSB}_{\xi_s}^{\xi_L-\xi_U}$ can be used as well. In that case, $\xi \in [\xi_L, \xi_U]$ is the percentage by which the center of the search space is offset.

6.3 A metric for initialization region bias

Most of today's real-world optimization problems are formulated in environments that undergo continual change, referred to as dynamic optimization problems [2]. When the global optimal solution changes, the population members have to move along an extended path with many local optima. To test the sensitivity of PSO on the search initialization, as well as its ability to move from the initial search space to more promising regions, Angeline [1] proposed reducing the initialization region, referred to as *Region Scaling* (RS) [36]. This initialization is adopted here as well, where the algorithm is initialized deliberately in a portion of the search space. A notable example of a class of algorithms suffering from sufficiently generating offsprings outside a given initial population, specially when the size of the population is small relative to the search space, is GA with Unimodal Normal Distribution Crossover (UNDX) [40].

Region Scaling (RS) [36] is an approach to qualitatively observe if a search algorithms has any IRB. By shrinking the initialization region (IR), an algorithm with no IRB will perform not worse than when the IR covers the entire design space. In order to quantify the IRB, in this study, the initialization space is gradually degraded, starting from the entire search space, to ζ percent of the search space. The method, hereafter referred to as ζ-RS, explicitly downsizes the initialization region to a region where the bottom left sides are all downsized to $\zeta \in [\zeta_L, \zeta_U]$ percent of the original length. A series of experiments is executed when ζ changes from ζ_L to ζ_U with a predefined step size of ζ_s. Due to the stochastic nature of most of optimization processes, each of the experiments is repeated r_f times. The observations have a form of $\left(\zeta, f_r^{\zeta}\right) \in \mathbf{R}^2$ where f_r^{ζ} is the best fitness an algorithm can find within a predefined budget on run $r \in [1, r_f]$, when the initial population is positioned randomly in a corner box of length ζ percent of the entire search space. An estimation of best-of-run $\hat{f}^{\zeta}$ as function of ζ can be calculated directly from the observed results.

$$\hat{f}^{\zeta} = \text{IRB}_{\zeta_s}^{\zeta_L-\zeta_U}.\zeta + c_2 \tag{6.5}$$

where $\text{IRB}_{\zeta_s}^{\zeta_L-\zeta_U}$ is the slope of the regression line fitted to $\left(\zeta, f_r^{\zeta}\right) \in \mathbf{R}^2$.

In special cases where ζ_s is equal to $||\zeta_U - \zeta_L||$ the metric is referred to as $\text{IRB}^{\zeta_L,\zeta_U}$. To measure the initialization region bias, the $\text{IRB}_{\zeta_s}^{\zeta_L-\zeta_U}$ has a greater generalization ability compared to $\text{IRB}^{\zeta_L,\zeta_U}$.

While search algorithms may perform better when they are initialized in the whole search space and benefit from knowing the search space, one with lower IRB is preferable over one with higher IRB.

6.4 Three population-based metaheuristics

The primary goal of this study is to asses CSB and IRB of a set of widely used and well-established metaheuristics. We do not aim at giving an exhaustive experimental comparison on a wide range of alternative search algorithms, rather we focus on a set of well benchmark instances. This section respectively formulates the particle swarm algorithm as proposed in [6] and gravitational search algorithm [46] in addition to presenting a modification of GSA.

6.4.1 A brief tour of the particle swarm optimization

Swarm intelligence, an emerging collective behavior of interacting agents with examples of ant colony [18] and bee colony [25], is a popular source of inspiration for the design of optimization algorithms. Particle swarm optimization (PSO) [27] is a successful instance of a nature-inspired algorithm for solving global optimization problems. A number of advantages have been attributed to PSO, making it a choice candidate as a benchmark algorithm. The standard PSO algorithm is suited to handle nonlinear nonconvex optimization problems with fast convergence characteristics. In this study, PSO is a reasonable choice for comparison as it does not have bias towards the center of the search space [26].

In classical PSO, every particle is a solution moving in a D-dimensional search space. A collection of particles is known as *swarm*. Each particle i has a position, $\mathbf{x}_i \in \mathbf{R}^D$, a velocity, $\mathbf{v}_i \in \mathbf{R}^D$ and the best position found so far, $\mathbf{p}_i \in \mathbf{R}^D$.

PSO uses two independent random variables, $r_1, r_2 \sim \mathcal{U}(0,1)$, scaled by constants C_1 and C_2. The constants C_1 and C_2 are known as *learning rates* and they influence the maximum step size a particle can take in a single iteration, representing the confidence of a particle on its best performance and that of the global best respectively. The movement equations of every particle $i \in 1,2,\ldots,S$, specified separately for every dimension $d \in 1,2,\ldots,D$, are given by expressions (6.6) and (6.7).

$$v_i^d = wv_i^d + C_1 r_1^d \left(p_i^d - x_i^d\right) + C_2 r_2^d \left(g_i^d - x_i^d\right), \tag{6.6}$$

$$\mathbf{x}_i = \mathbf{x}_i + \mathbf{v}_i, \tag{6.7}$$

where w is a predefined constant representing the confidence of particle on its own movements and p_i^d and g_i^d are personal best and global best positions respectively. S is the number of particles in the swarm.

To ensure convergence by avoiding explosion, Clerc et al. [6] introduces the *constriction factor* and modifies the velocity update equation as follows:

$$v_i^d = \chi \left(v_i^d + C_1 r_1^d \left(p_i^d - x_i^d\right) + C_2 r_2^d \left(g_i^d - x_i^d\right)\right), \tag{6.8}$$

where $\chi = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|}$ and $\varphi = C_1 + C_2$, $\varphi > 4$.

6.4.2 A Brief Tour of the GS Algorithm

Gravitational search algorithm (GSA) [46] is a relatively new technique that has been empirically shown to perform well on many optimization problems [4, 16, 19, 23, 31, 32, 41,

44, 47, 54]. GSA was inspired by the Newton's law of universal gravitation. In its original version, GSA scatters particles in a feasible region of the search space, where they interact with each other under Newton's gravitational force and move in the search area seeking an optimal design variable. GSA shares features with several competing schemes, for instance by sharing information between solutions. In contrast to EAs where solutions *die* at the end of each generation, in PSO and GSA, solutions survive throughout the optimization process, providing a substantial source of information for the population when searching the global optimum.

In GSA, like in many other population based optimization techniques, to guide the population in the search space E, some measure of discrimination is needed, referred here as a fitness of each candidate solution $\mathbf{x}_i$. Each candidate solution is a particle with a mass M_i. A good solution is analogous to a particle with a high mass, while a poor solution represents a particle with a low mass. A particle with a high mass resists change more than one with a low mass and tends to have higher impact on other particles, thereby sharing its features with low quality solutions. The attractive gravitational force governs the movement of the particles in the search space. The search begins by an attractive force with a strength and direction as a function of the mass of particle itself, the mass of other particles and its relative distance to the other particles. The force is applied to static particles of one under which their position in the next time step changes and they gain velocity. The quantity of the resulting force is determined by Newton's gravitational law. A solution with a higher mass exerts a stronger force compared to a smaller mass. The kinetic energy stored in particles is a form of memory, allowing them to steer their movement under the influence of their memory and external forces. The sum of the force field $\mathbf{F}_i$ and the particle's kinetic energy, induced from its velocity and mass, is the total force acting on them, which together with its current position $\mathbf{x}_i(t)$, determines the particles next position $\mathbf{x}_i(t+1)$ in the search space.

In original GSA [46], the mass of particles, considering its quality, is assigned as follows:

$$M_i = \frac{m_i}{\sum_{j=1}^{S} m_j}, i = 1, 2, \ldots, S \tag{6.9}$$

where

$$m_i = \frac{f(\mathbf{x}_i) - \max_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j)}{\min_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j) - \max_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j)}, \tag{6.10}$$

and S is the number of particles. The resulting gravitational force acting on particle i in direction d is determined using Equation (6.11).

$$F_i^d = \sum_{j\in Kbest} r_j F_{ij}^d, \tag{6.11}$$

where $Kbest$ is a set of particles with the highest mass, $r_j \sim \mathcal{U}(0,1)$ and F_{ij}^d is the gravitational force exerted by particle j on particle i. To provide a better exploration in the early iterations $|Kbest|$ is set at S in the beginning; however the exploration must be decreased gradually. Therefore choosing a decremented function for $|Kbest|$ increases the exploitation of the algorithm when the number of iterations increases.

The force exerted by particle j acting on particle i is defined as:

$$F_{ij}^{d} = G\frac{M_i \times M_j}{R_{ij}+\varepsilon}\left(x_j^d - x_i^d\right) \tag{6.12}$$

where R_{ij} is Euclidian distance between particles i and j. and G, the gravitational constant initialized at G_0 is determined using Equation (6.13) as:

$$G = G_0 e^{-\alpha \frac{t}{MaxIteration}} \tag{6.13}$$

where α is algorithmic tuning parameter and $MaxIteration$ is the maximum iteration.

The equations of motion of every particle are described using (6.14) and (6.15) as:

$$\mathbf{v}_i(t+1) = \mathbf{R} \times \mathbf{v}_i(t) + \frac{\mathbf{F}_i}{M_i}.\Delta t, \tag{6.14}$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1).\Delta t, \tag{6.15}$$

where $\Delta t = 1$, $\mathbf{R} \sim \mathcal{U}(0,1)$ is an array of size D corresponding to each element in vector $\mathbf{v}_i$.

6.4.3 mdGSA, a mass-dispersed GSA

In GSA, an increase in the number of particles changes the mass assigned to them as a result of an increase in the denominator of the Equation (6.9). This increase in the denominator smooths out the difference between the mass of particles, making them in absolute terms, more equal in exerting an attractive force and equally resistant to change in their position as a result of the applied gravitational force. The swarm can be seen as one particle with a uniform mass distribution. Under the Newtonian gravitational force, this brings the particles closer to the center of the swarm, resulting in an increase in the density of swarm. As a result, they move more quickly towards the center of the search space [11]. This may explain the center-seeking behavior of standard GSA. It is against this backdrop that a different GSA called mass-dispersed gravitational search algorithm (mdGSA) is devised and tested here.

A more intense discrimination of solutions can be achieved by using the concept introduced in the Simulated Big Bounce (SBB) algorithm [14]. SBB is a global search algorithm that is inspired by the Big Bounce theory (a cosmological oscillatory model of the Universe), that, next to exploitation, applies robust exploration in order to escape local minima. In this approach, based on their fitness, the particles are assigned a mass in the range of $[L_M, U_M]$. g, the function that maps the fitness to the mass $g : \mathbf{R} \to \mathbf{R}$, $f(\mathbf{x}_i) \mapsto g\left(f\left(\mathbf{x}_i\right)\right), \forall \mathbf{x}_i \in E$ can be any monotonically nondecreasing (and possibly time varying) function in principle with real values defined on a the set of fitness of particle $\mathbf{x}_i$ whose value is non-negative for $f(\mathbf{x}_i)$. We take g as a linear time-invariant strictly increasing function as follows [14]:

$$\begin{aligned} &M_i = g(f(\mathbf{x}_i)) = \\ &L_M + (U_M - L_M)\frac{f(\mathbf{x}_i) - \max\limits_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j)}{\min\limits_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j) - \max\limits_{j\in\{1,\ldots,S\}} f(\mathbf{x}_j)}. \end{aligned} \tag{6.16}$$

mdGSA's basic steps in pseudo-code are shown in Algorithm 6.1.

Algorithm 6.1 Pseudo code of mass-dispersed gravitational search algorithm (mdGSA)

Input: Search space E, fitness function f, S, G_0, α
1: Initialize particle's Location, $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_S)^T$
2: **while** $t < MaxIteration$ **do**
3: Fitness calculation
4: Update M_i, $\forall i = 1, \ldots, S$ ▷ According to (6.16)
5: Update G ▷ According to (6.13)
6: Update attractive force F_i^d, $\forall i = 1, \ldots, S$
7: Update $\mathbf{v}_i$, $\forall i = 1, \ldots, S$ ▷ According to (6.14)
8: Update $\mathbf{x}_i$, $\forall i = 1, \ldots, S$ ▷ According to (6.15)
9: $t++$ ▷ t is the number of iterations
10: **end while**
Output: $\mathbf{x}^*$ and $f(\mathbf{x}^*)$

6.5 Experimental results

When comparing the effectiveness of different optimization heuristics, a standard performance measure is the best fitness a certain algorithm can reach within a predefined number of function evaluations. This is based on the assumption that the dominating factor in measuring computational effort is fitness evaluation, which is usually valid for complex optimization problems [13, 15, 51]. In the experiments, this, is modeled as if the maximum computational resource budget available to carry out a task were limited, which is equivalent to a situation where the maximum time budget for which the best solution has to be delivered is limited.

Although the studied optimization algorithms can be simply extended and adapted for real-world optimization problem, such adaptation may require more elaborate mechanisms. One example of this is constraint-handling.[2] It is well-known that in real-world optimization problems there are normally constraints of different types (e.g., related to the geometry of structural elements to completion times, etc.) that must be satisfied for a solution to be acceptable. Traditionally, penalty functions have been used to handle constraints [7]. However, because of the several problems associated to penalty functions (e.g., the definition of appropriate penalty values is normally a difficult task that has a serious impact on the performance of the optimizer), some authors have proposed alternative constraint-handling approaches that require less critical parameters and perform well across a variety of problems (see for example [7, 34, 50]).

In the experiments described in this section, the common parameters used in each algorithm, such as population size and total number of fitness evaluation, where chosen to be the same. Unless indicated otherwise, the population size is set at 50 and the maximum number of fitness evaluation, $MaxIteration$, is set at 100,000. For Gene regulatory network (GRN) model identification problem (Section 6.5.2), the maximum number of fitness evaluation is set at 200,000.

[2] Although constraint-handling techniques are very important in real-world optimization problems, their discussion is beyond the scope of this article, due to space limitations. Interested readers are referred to other references for more information on this topic (see for example [34, 49]).

PSO settings

The PSO parameters across the experiments have been $\varphi = 4.1$, $\varphi_1 = \varphi_2$ and $\chi = 0.729$, which is equivalent to setting $C_1 = C_2 = 1.496$ and $w = 0.729$ [6].

GSA settings

The GSA parameters are as follows: G_0 is set at 100, α is set at 20, *Kbest* is set at number of particles, S, and is linearly decreased to 1 in the final iteration, ($MaxIteration$) [46].

mdGSA settings

The common setting are GSA settings. The upper and lower bound of mass are set at 1 and 0.01, respectively.

Because the optimization techniques under study are stochastic in nature, for a given function of a given dimension 30 independent runs where executed for all ξ and ζ values. Throughout the experiments discussed in this study, the population size and maximum fitness evaluation remain fixed, although it is well known that these control-parameters affect the performance of the algorithms. The reason not to change the parameters was primarily the motivation of our study in exposing the center-seeking behavior and IRB of GSA, rather than emphasizing its performance under different control-parameter settings. The second reason relates to the assumption that end-users do not know much about the algorithmic parameters for their optimization problem.

6.5.1 Experiment 1: Standard optimization problems

From the test beds studied in [46, 61], those with varying dimensions are used in this study to capture the CS behavior of GSA [46], in addition to those studied in [33]. The test beds along with their characteristics are listed in Table 6.12[3].

Because the primary objective of this study is to specify the center-seeking behavior of GSA, the Schwefel function is excluded, since its optimal solution is not close to the center of the search space. In Table 6.12, D is the dimension of function. The optimal solution $f(\mathbf{x}^*)$ for all the adopted test functions is located at $[0, \ldots, 0]$, with the exception of Dixon-Price, that has its optimal solution located at $2^{-\frac{2^d-2}{2^d}}$ for $d = 1, 2, \ldots, D$ as well as Levy and Rosenbrock with optimal solution at $[1, \ldots, 1]$. Of the 14 adopted test studies, half are unimodal, while the others are multimodal. The set contains five separable and nine non-separable functions. A separable function can be decomposed into D one-dimensional functions.

The performance of the algorithms is evaluated from both *accuracy* and *robustness* perspectives. Accuracy is the degree of precision of an optimization algorithm in locating an optimal solution. An algorithm with a higher accuracy tends to come closer to the optimal solution. Accuracy is studied in two different settings for optimization problems,

[3]Note that, in [46], the Rosenbrock function (also known as Banana problem) is treated as a unimodal test function when D is set at 30, while it is indeed multimodal when the problem dimensions is more than three [17, 53].

ξ-Accuracy and ζ-Accuracy. ξ-Accuracy refers to the performance of the optimization algorithms (OAs) under study when the center of the search space changes, while ζ-Accuracy refers to the performance of OAs when the initialization region changes. Robustness is defined here as the degree of bias of an optimization heuristics on the center of the search space or the initialization region. A robust optimization algorithm has no CS nor IR bias.

In our experiments, the metrics are measures in a log-linear scale, because the best-of-run found by each algorithm, in many cases, changes several orders of magnitude as a result of ξ-CO and/or ζ-RS.

ξ-CO test results

Figures 6.2 to 6.3 are the test results of the ξ-CO on a selection of the studied standard benchmark problems when D is set at 50 and 100, respectively. The x-axis is ξ and the y-axis is the performance of each OA, averaged over $r_f = 30$ independent runs. Throughout this study ξ_L and ξ_U are set at 5 and 45 respectively and the step size ξ_s is set at 5. These choices for ξ_L and ξ_U are based on the assumption that an optimal solution of a real-world problem is usually neither at the center of the search space, nor at the boundaries, suggesting $\xi = 0$ and $\xi = 50$ are not interesting cases to study.

Figures 6.2 to 6.3 show that, as a result of downsizing the search space, the performance of the PSO algorithm is nearly a horizontal line in most of the experiments. The performance of the GSA deteriorates quickly by moving the optimal solution from the center of the search space. mdGSA falls somewhere in between.

Tables 6.1 and 6.3 summarize the $\mathrm{CSB}_{\xi_s}^{\xi_L-\xi_U}$, when D is set at 50 and 100, respectively. In each table, asterisk symbols are used to denote no statistically significant association between the observed change in estimation of the best-of-run as a result of change in ξ using F-statistics. Statistical testing is performed to determine whether or not $\mathrm{CSB}_{\xi_s}^{\xi_L-\xi_U}$ measures are zero, in addition to testing if $\mathrm{CSB}_{\xi_s}^{\xi_L-\xi_U}(\mathrm{mdGSA})$ is statistically smaller than that of $\mathrm{CSB}_{\xi_s}^{\xi_L-\xi_U}(\mathrm{GSA})$.

In the case of Step function, F_{14}, the optimal solution, 0, is attainable under a relatively wide range of design values. This, in log-linear scale, leaves us with no way to fit a line to the observed performance. Hence, the Step function is excluded from Tables 6.1 and 6.3. As a replacement for it, the convergence curse of some selected ξ values are visualized in Figure 6.4.

For each of the nine ξ values on each of the 14 problems, the optimization methods are statistically compared using pairwise contrast. The number of times an OA has a statistically significant superiority (SSS) compared to other optimization algorithms on a total of $9 * 14$ problems is shown in Tables 6.2 and 6.4. We also report the number of times an optimization approach achieves the best result, best mean and best median when it is statistically superior to others. As an example, the number of times an algorithm performs best is the number of times a) it is statistically superior to others and b) it has the best fitness over 30 runs compared to other competing algorithms. In addition, the number of times the worst result is achieved is reported.

Results on 50D problems First, we evaluate the robustness of each algorithm when the dimension of the optimization problems is set at 50. For each optimization algorithm,

CSB_5^{5-45} are reported in Table 6.1. The slope of fitted line describing center-seeking bias of PSO (Mdn = -0.3019) was not significantly different from zero (Wilcoxon signed-rank, W=33, p=0.2071) while for GSA (Mdn = 5.8795, Wilcoxon signed-rank, W=1, p=2.44E-4) and mdGSA (Mdn = 0.9621, Wilcoxon signed-rank, W=8, p=0.0030) the fitted line had a slope significantly different from zero. Interestingly, the observed $CSB_5^{5-45}(\text{GSA})$ was significantly higher than $CSB_5^{5-45}(\text{mdGSA})$ (Wilcoxon rank sum test, W=224, p = 0.069).

Out of total of 9*14 experiments each repeated 30 times, when D = 50, mdGSA and GSA are competing closely when looking at the number of times they were statistically superior to the others (Table 6.2). As a result, a statistical test of significance was performed on median of fitness they both can achieve under different settings of studied optimization problems. For that, logarithmic transformation of the median of fitness values was performed in the first place (because the Wilcoxon test assumes that the distribution of the data, although not normal, is symmetric). Wilcoxon paired sample test (W=3399, p=0.8865) confirms that there is no significant difference between the performance of GSA and mdGSA. Note that, due to logarithmic transformation of the medians, the Step function is excluded from the test, which means that the test is performed on 13 test problems, each with 9 different ξ values.

So, while mdGSA and GSA come in joint first place, PSO, with only two cases of SSS, cames second (Table 6.2). While PSO is the most robust of the algorithms under study, it did not perform better than the others in terms of its ξ-accuracy.

The picture changes when looking at other measures, for instance the worst solutions over the 30 runs. While GSA and mdGSA come close in terms of their statistical superiority, mdGSA shows the worst fitness in only 13 cases, compared to 47 cases for GSA, which suggests that GSA is more susceptible to trapping around a local optimum and missing the global optimum.

Table 6.1: CSB_5^{5-45} of the studied algorithms when D is set at 50.

	Simulation results		
	$CSB_5^{5-45}(\text{GSA})$	$CSB_5^{5-45}(\text{mdGSA})$	$CSB_5^{5-45}(\text{PSO})$
Ackley	9.219	1.25	-0.49*
Dixon	1.217	0.01371	0.4631
Griewank	5.879	2.195	0.8024*
Levy	29	12.42	4.599
Penalty1	31.3	5.48	-1.596*
Penalty2	49.75	0.856*	-2.843
Quartic	-0.05537*	0.01948*	-1.119
Rastrigin	2.02	1.503	0.2341
Rosenbrock	0.3789	0.9621	-0.04733*
Schwefel222	4.5	1.074	-0.3019
Schwefel12	11.01	-0.5769	0.1159*
Schwefel121	11.79	-0.02214*	-0.3442
Sphere	4.285	0.02609*	-0.6474

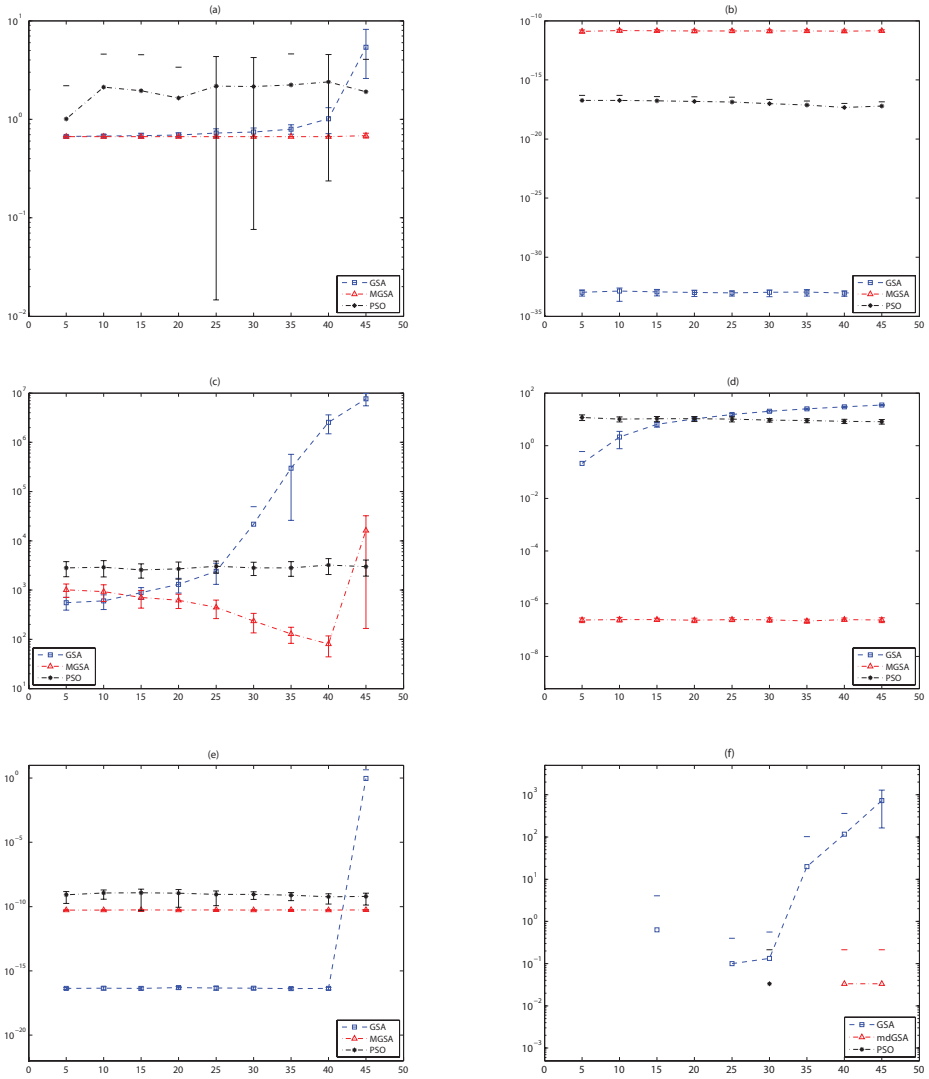

Figure 6.2: Test results of the ξ-CO for GSA (blue), mdGSA (red) and PSO (black) when $D = 50$. a) Dixon-Price, b) Quartic, c) Schwefel P1.2, d) Schwefel P2.21, e) Sphere, f) Step.

Table 6.2: Comparison results of the three studied algorithms (GSA, mdGSA and PSO) when ξ changes from 5 to 45 and when D is set at 50.

	Simulation results		
	GSA	mdGSA	PSO
SSS [a]	57	51	2
Best [b]	56	33	1
Worst [c]	48	12	63
Best mean [d]	43	49	2
Best median [e]	57	51	2

[a] # of times the fitness values are statistically superior.
[b] # of times the best fitness value is obtained.
[c] # of times the worst fitness value is obtained.
[d] # of times the best mean of fitness values is obtained.
[e] # of times the best median of fitness values is obtained.

Results on 100D problems Table 6.3 summarizes the results of the studied algorithms when D is set at 100 considering CSB_5^{5-45} metric. In this case again the null hypothesis of equality of median of $CSB_5^{5-45}(\text{PSO})$ to zero is accepted (Mdn = -0.1096, Wilcoxon signed-rank, W=32, p=0.1878) while for both GSA (Mdn = 12.3409, Wilcoxon signed-rank, W=0, p=1.22E-4) and mdGSA (Mdn = 3.4746, Wilcoxon signed-rank, W=0, p=1.22E-4) this hypothesis is rejected. $CSB_5^{5-45}(\text{mdGSA})$ was statistically lower than $CSB_5^{5-45}(\text{GSA})$ (Wilcoxon rank sum test, W=215, p = 0.0227), suggesting that mdGSA indeed dilutes the strong center-seeking bias of GSA. This can also be confirmed by looking at the median of the two optimization heuristics.

Note that, as a result of an increase in the dimension of the search space, GSA loses its ξ-accuracy in favor of mdGSA (Table 6.4). GSA, out of a total of 9*14 experiments, is statistically superior to the others in 20 cases, mdGSA in 67 cases and PSO in only 11 cases. Wilcoxon paired-sample test (W=1341, p=9.4594E-9) also confirms the superiority of mdGSA when looking at the logarithmic transformation of the median of their performance on the set of optimization problems.

While GSA shows the worst results in 71 cases, and PSO in 55 cases, this was never the case for mdGSA, which, again suggests that GSA, when compared to mdGSA, is more susceptible to trapping around a local optimum.

ξ-CO test results on Step function The results of the Step function are not presented in Tables 6.1 and 6.3. Instead, for $\xi = \{5, 25, 45\}$, Figure 6.4 compares the performance of the algorithms under study. In this figure, GSA_5 for instance means the performance of GSA when ξ is set at 5. The results presented are averaged over 30 independent runs to eliminate the random effect of arbitrary initialization of initial population.

When $D = 50$ (Figure 6.4.a) as a result of moving the center of the search space, the performance of both PSO and mdGSA does not change very much. For GSA, there is a clear deterioration in performance when the center of the search space is moved. When $\xi = 5$ (which basically means that the optimal solution is near to the center of the search space), GSA locates the optimal solution very quickly and defeats its contenders, while it

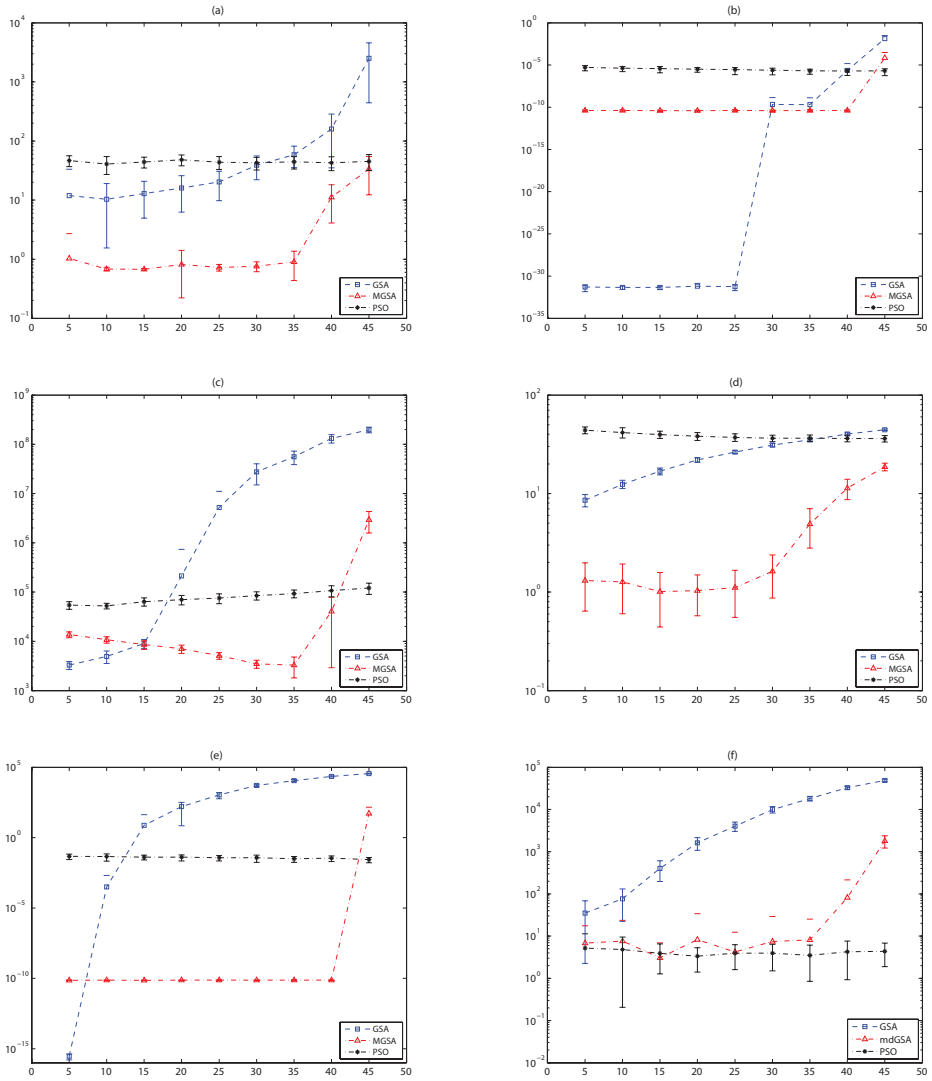

Figure 6.3: Test results of the ξ-CO for GSA (blue), mdGSA (red) and PSO (black) when $D = 100$. a) Dixon-Price, b) Quartic, c) Schwefel P1.2, d) Schwefel P2.21, e) Sphere, f) Step.

Table 6.3: CSB_5^{5-45} of the studied algorithms when D is set at 100.

	Simulation results		
	CSB_5^{5-45}(GSA)	CSB_5^{5-45}(mdGSA)	CSB_5^{5-45}(PSO)
Ackley	26.59	15.35	-0.1096*
Dixon	5.409	2.789	-0.02233*
Griewank	4.043	2.863*	-0.6713
Levy	10.98	7.655	-0.3052
Penalty1	7.644	5.933	-0.1683
Penalty2	12.43	5.624	-0.5562
Quartic	44.37	5.285	-1.158
Rastrigin	1.943	1.627	0.2927
Rosenbrock	12.34	2.723	0.06779*
Schwefel222	17.93	12.84	-0.1035*
Schwefel12	14.21	3.023	0.897
Schwefel121	1.737	3.475	2.457
Sphere	54.84	2.904	-0.5122

Table 6.4: Comparison results of the three studied algorithms (GSA, mdGSA and PSO) when ξ changes from 5 to 45 and when D is set at 100.

	Simulation results		
	GSA	mdGSA	PSO
SSS [a]	20	67	11
Best [b]	19	58	5
Worst [c]	71	0	55
Best mean [d]	14	63	11
Best median [e]	20	67	11

[a] # of times the fitness values are statistically superior.
[b] # of times the best fitness value is obtained.
[c] # of times the worst fitness value is obtained.
[d] # of times the best mean of fitness values is obtained.
[e] # of times the best median of fitness values is obtained.

fails to locate the optimal solution when it is removed from the center of the search space ($\xi = \{25, 45\}$).

With regard to the case where the dimension of the search space is set at 100, the results of PSO basically remain the same (Figure 6.4.b). While, in the beginning of the search, for $\xi = 5$, GSA has the greatest reduction of fitness among its competitors, it has the highest performance deterioration when the center of the search space is moved, confirming its strong center-seeking bias. mdGSA, although defeating GSA under equal settings in all three cases, shows a degrading performance when ξ is set at 45. These observations are

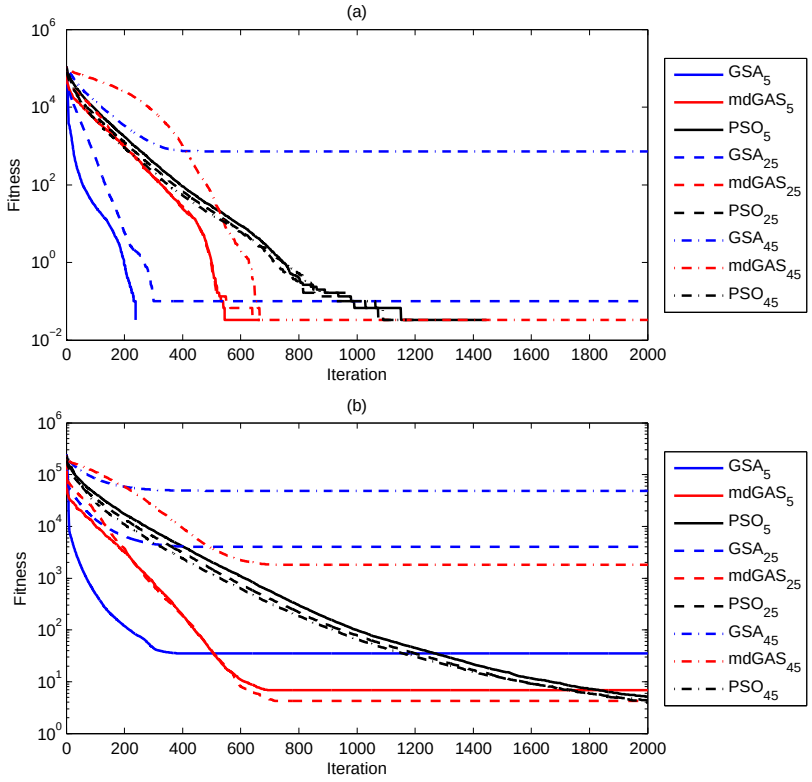

Figure 6.4: Performance comparison on Step function for $\xi = \{5, 25, 45\}$ *when a)* $D = 50$*, b)* $D = 100$*.*

consistent with our previous findings.

ζ-RS test results

Figures 6.5 to 6.6 contain the test results of the ζ-RS on the test problems when D is 50 and 100, respectively. The x-axis is the percentage of shrinking the initialization region and the y-axis is the performance averaged over 30 independent runs. An algorithm with no IRB has the opportunity to explore areas outside the initialization region.

Throughout this study, ζ_s is set at 10 when $\zeta_L = 0$ and $\zeta_U = 90$. A line best fitted to 10*30 observations has a slope of $\text{IRB}^{\frac{\zeta_L - \zeta_U}{\zeta_s}}$.

Tables 6.5 and 6.7 present an estimation of the degree of change in the quality of best-of-run of each optimization heuristic as a result of shrinking the initialization region, when the dimension of problems are set at 50 and 100, respectively. Here, again, in each table asterisk symbols are used to denote no statistically significant association between the observed change in the estimation of the best-of-run as a result of change in ζ using F-statistics.

For the same reason as ξ-CO, the Step function is excluded from the Tables 6.5 and 6.7. Instead, for some selected ζ values, its convergence curse is visualized in Figure 6.7.

To compare ζ-accuracy of the optimization heuristics under study, we look at the number

Table 6.5: IRB_{10}^{0-90}, Initialization region bias when ζ changes from 0 to 90 and when D is set at 50.

	Simulation results		
	IRB_{10}^{0-90}(GSA)	IRB_{10}^{0-90}(mdGSA)	IRB_{10}^{0-90}(PSO)
Ackley	1.162	-0.005047*	3.556
Dixon	0.001543*	-8.645e-015*	0.1817
Griewank	3.462	-0.48*	0.1506*
Levy	1.255*	-0.5644*	3.842
Penalty1	13.02	0.6169*	0.2494*
Penalty2	22.8	-0.6841*	0.5677*
Quartic	-0.02836*	-0.01201*	0.8033
Rastrigin	-0.000824*	-0.03876	0.1463
Rosenbrock	0.7701	0.01091*	0.2249
Schwefel222	-0.007248*	-0.002421*	0.4908
Schwefel12	6.375	0.01072*	0.8677
Schwefel121	5.495	-0.04105	0.3853
Sphere	30.28	0.003999*	0.2297

of times one is statistically superior to the others and the number of times one has the worst fitness (Tables 6.6 and 6.8). For each optimization heuristic, the number of test problems equals 10*14 (10 ζ values for each of the 14 problems). We also report the number of times an optimization method has the best performance, best mean and best median when it is statistically superior. The number of times each has the worst fitness is reported as well.

Results on 50D problems In 10 of the 13 optimization problems, the performance of the PSO algorithm degrades as a result of shrinking the initialization space. However compared to GSA, its IRB_{10}^{0-90} is small. The performance of the GSA on eight optimization problem degrades under ζ-RS test. The mdGSA is more robust to the initialization region and is nearly a straight line in 11 of the 13 experiments. The slope is significantly different than zero in only two cases. In both of these cases, the slope is negative, meaning that the performance of the algorithm increases as a result of shrinking the initialization region. A possible explanation for this will be suggested later.

For GSA (Wilcoxon signed-rank, W=227.5, p=0.0012) and PSO (Wilcoxon signed-rank, W=240.5, p=1.60E-4), the IRB_{10}^{0-90} values are significantly different than zero, while in the case of mdGSA (Wilcoxon signed-rank, W=162.5, p=0.1655), the IRB_{10}^{0-90}'s does not differ significantly from zero, which suggests that both GSA and PSO are not robust with regard to the initialization region.

In the combination of 14 test problems and 10 initializations regions, GSA and mdGSA are competing closely when looking at the number of times each of them is significantly superior to the others (in 75 and 51 cases, respectively). So, similar to ξ-CO, a statistical test of significance is performed on the median of fitness each of them can obtain on different settings of studied optimization problems when a logarithmic transformation of the median

of fitness values is performed. Wilcoxon signed-rank test[4] results (W=3981, p=0.5205) confirm that there is no significant difference between the performance of GSA and mdGSA. So, mdGSA and GSA come in joint first place, while PSO comes in second place.

In 69 cases, PSO has the worst fitness over 30 runs, while GSA, with 58 cases, comes in second place and mdGSA, with 11 cases, shows the best performance. This suggests that mdGSA is less susceptible to the attraction of local optima. Table 6.6 summarizes the results.

Table 6.6: Comparison results of the three studied algorithms (GSA, mdGSA and PSO) when ζ changes from 0 to 90 and when D is set at 50.

	Simulation results		
	GSA	mdGSA	PSO
SSS [a]	75	51	0
Best [b]	74	43	0
Worst [c]	58	11	69
Best mean [d]	56	50	0
Best median [e]	75	51	0

[a] # of times the fitness values are statistically superior.
[b] # of times the best fitness value is obtained.
[c] # of times the worst fitness value is obtained.
[d] # of times the best mean of fitness values is obtained.
[e] # of times the best median of fitness values is obtained.

Results on 100D problems In Table 6.7, the slope of the line fitted to observed best-of-run fitness values for each benchmark problem is presented when D is set at 100. $\text{IRB}_{10}^{0-90}(\text{PSO})$ has significantly positive associations with ζ in all cases. GSA has a significantly positive IRB_{10}^{0-90} in 11 of the total of 13 cases, while $\text{IRB}_{10}^{0-90}(\text{mdGSA})$ has no significant positive associations with ζ. This suggests that GSA (Wilcoxon signed-rank, W=247, p=9.97E-5) and PSO (Wilcoxon signed-rank, W=260, p=4.15E-6) both have significant initialization region bias when D is set at 100, while mdGSA (Wilcoxon signed-rank, W=156, p=.29) has IRB values that are significantly close to zero.

As was the case when the dimension of the test problems is set at 50, mdGSA again has some negative IRB values. An intuitive explanation for this observation goes as follows. Due to high mass discrimination of mdGSA, when the initialization region is small compared to the entire design space, the particle with the highest fitness that is equivalent to highest mass is better able to direct the entire swarm towards the optimal solution. As a result the algorithm performs slightly better compared to when the initialization is the entire search space. It is important to point out that the improvement is not visible in most of the cases from Figures 6.5 and 6.6 and that this improvements, in all 13 cases, has no significant correlation with ζ, confirmed by analysis of covariance (F-test). When compared to GSA, mdGSA has significantly less IRB (Wilcoxon rank sum test, W=246, p = 1.65E-4). So, in

[4]Note that, again, the Step function is excluded from the test due to logarithmic transformation of the medians, which means that the test is performed on 13 test problems, each with 10 different ζ values.

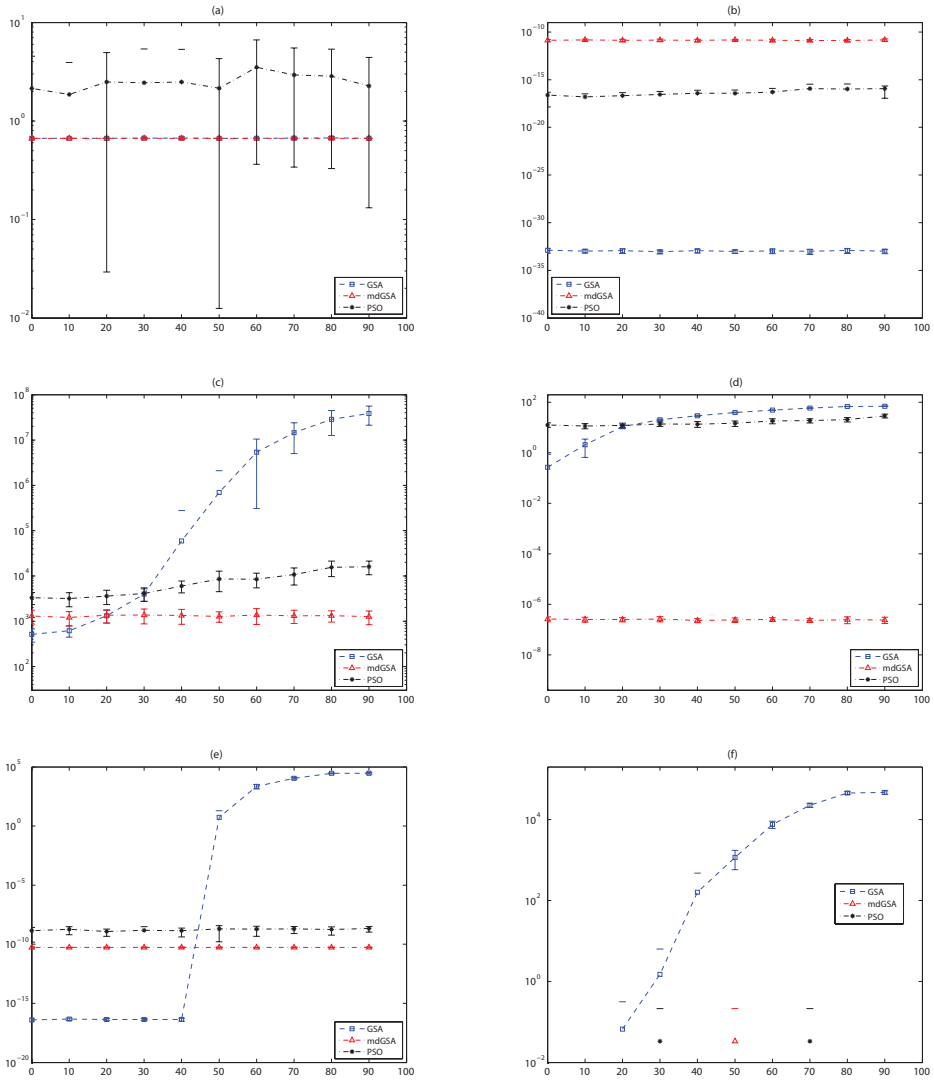

Figure 6.5: Test results of the ζ-RS for GSA (blue), mdGSA (red) and PSO (black) when D = 50. a) Dixon-Price, b) Quartic, c) Schwefel P1.2, d) Schwefel P2.21, e) Sphere, f) Step.

terms of robustness in change in IR, mdGSA defeats its competitors.

Summarized in Table 6.8, mdGSA presents higher ζ-accuracy compared to its two competitors. With 87 cases of significant superiority out of total of 14*10 cases, mdGSA ranked in the first place, followed by GSA, with 26 cases, and PSO, with zero cases. Wilcoxon paired-sample test (W=1735, p=4.5869E-9) also confirms the superiority of mdGSA when looking at the logarithmic transformation of the median of their performance.

Table 6.7: IRB_{10}^{0-90}, Initialization region bias when ζ changes from 0 to 90 and when D is set at 100.

	Simulation results		
	IRB_{10}^{0-90}(GSA)	IRB_{10}^{0-90}(mdGSA)	IRB_{10}^{0-90}(PSO)
Ackley	11.29	0.05864*	1.153
Dixon	0.286	-0.02044*	0.1956
Griewank	2.623	0.4001*	0.2914
Levy	2.865	-0.05098*	0.9592
Penalty1	10.06	0.03984*	0.3084
Penalty2	10.05	0.007134*	0.1713
Quartic	-0.04523*	-0.002844*	0.9314
Rastrigin	0.00851*	-0.02011*	0.2116
Rosenbrock	8.438	-0.0005933*	0.4377
Schwefel222	2.635	-0.2029*	0.6911
Schwefel12	7.004	-0.01445*	1.124
Schwefel121	1.116	-0.09892*	0.3697
Sphere	21.57	0.004679*	0.2886

GSA has the worst fitness in 75 cases, and PSO in 65 cases. mdGSA has the smallest number of worst fitness compared to that of GSA and PSO, confirming its superiority over its competitors in terms of ζ-accuracy. This again suggests that mdGSA is less susceptible to premature convergence to local optimum when compared to PSO and GSA.

Table 6.8: Comparison results of the three studied algorithms (GSA, mdGSA and PSO) when ζ changes from 0 to 90 and when D is set at 100.

	Simulation results		
	GSA	mdGSA	PSO
SSS [a]	26	87	0
Best [b]	26	73	0
Worst [c]	75	0	65
Best mean [d]	24	85	0
Best median [e]	26	87	0

[a] # of times the fitness values are statistically superior.
[b] # of times the best fitness value is obtained.
[c] # of times the worst fitness value is obtained.
[d] # of times the best mean of fitness values is obtained.
[e] # of times the best median of fitness values is obtained.

ζ-RS test results on Step function Again for the Step function, the results are not presented in Tables 6.5 and 6.7. So the performance of the studied algorithms when $\zeta =$

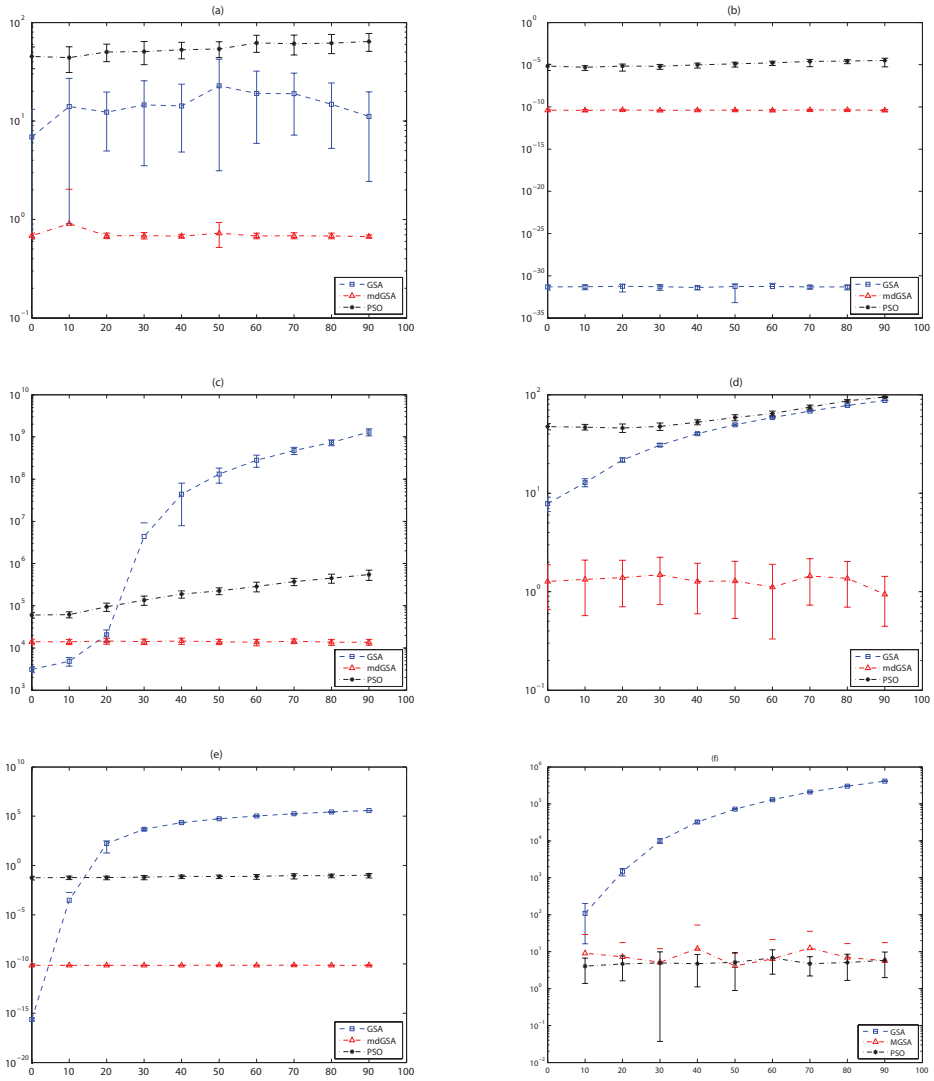

Figure 6.6: Test results of the ζ-RS for GSA (blue), mdGSA (red) and PSO (black) when $D = 100$. a) Dixon-Price, b) Quartic, c) Schwefel P1.2, d) Schwefel P2.21, e) Sphere, f) Step.

$\{10, 50, 90\}$ are presented in Figure 6.7.

When $D = 50$ (Figure 6.7.a) GSA_{10} has a sharp fitness decrease. The performance degrades significantly as a result of shrinking the initialization region (GSA_{50} and GSA_{90}). For mdGSA, the performance slightly changes as a result of change in IR, but the pattern was not clear. The performance of PSO was basically the same under different IRs.

GSA_{10}, when the dimension is set at 100 (Figure 6.7.b), has a sharp fitness change

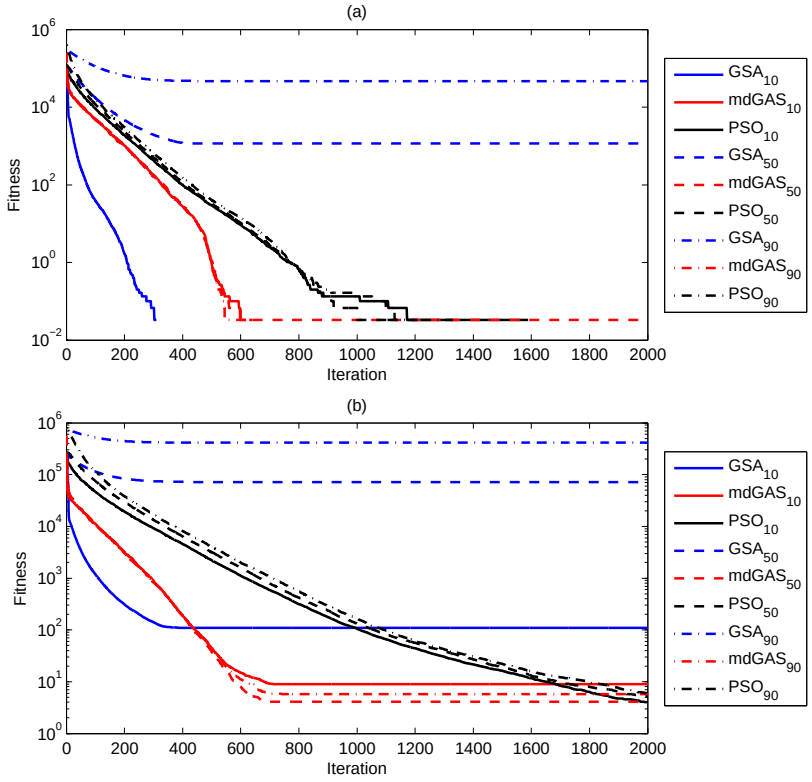

Figure 6.7: Performance comparison on Step function for $\zeta = \{10, 50, 100\}$ *when a)* $D = 50$*, b)* $D = 100$*.*

and becomes almost stagnant after approximately 300 iterations on average. As a result of shrinking the search space, there is a clear pattern in deterioration of the performance GSA. PSO exhibits similar patterns as expected, shrinking the initialization region does not appear to affect its performance. For mdGSA, on the Step function and when the dimension of the problem is set at 100, moving the optimum away from the center of the search space improves the performance slightly. These observations are compatible with our former findings.

6.5.2 Experiment 2: Gene regulatory network model identification

Gene regulatory network (GRN) model identification can be a good real-world application to test the center-seeking behavior and convergence speed of the optimization algorithms, since the optimal solution is, naturally, not at the center of the search space. Moreover the problem is highly nonlinear and complex [20, 28, 55]. A short introduction to GRN is provided below.

Gene regulatory Network

The activation and inhibition of genes are governed by factors within a cellular environment and outside of the cell. This level of activation and inhibition of genes is integrated by gene regulatory networks (GRNs), forming an organizational level in the cell with complex dynamics [9].

GRNs in a cell are complex dynamic network of interactions between the products of genes (mRNAs) and the proteins they produce, some of which in return act as regulators of the expression of other genes (or even their own gene) in the production of mRNA. While low cost methods to monitor gene expression through microarrays exist, we still know little about the complex interactions of these cellular components. Mathematical modeling of GRNs is becoming popular in the post-genome era [29, 30]. It provides a powerful tool, not only for a better understanding of such complex systems, but also for developing new hypotheses on underlying mechanisms of gene regulation. The availability of high-throughput technologies provides time course expression data, and a GRN model built by reverse engineering, may explain the data [38]. Since many diseases are the result of polygenic and pleiotropic effects controlled by multiple genes, genome-wide interaction analysis is preferable to single-locus studies. Readers looking for more information on GRN might refer to Schlitt [52].

S-system gene network model

Usually, sets of ordinary differential equations (ODEs) are used as mathematical models for these systems [58]. S-system approaches, on the other hand, use time-independent variables to model these processes. Assuming the concentration of N proteins, mRNAs, or small molecules at time t is given by $y_1^t, y_2^t, \ldots, y_i^t, \ldots, y_N^t$, S-systems model the temporal evolution of the ith component at time t by power-law functions of the form (6.17).

$$\frac{dy_i^t}{dt} = \alpha_i \left(\prod_{j=1}^{N} (y_j^t)^{g_{ij}} \right) - \beta_i \left(\prod_{j=1}^{N} (y_j^t)^{h_{ij}} \right). \tag{6.17}$$

The first term represents all factors that promote the expression of component i at time t, y_i^t, whereas the second term represents all factors that inhibit its expression. In a biochemical engineering context, the non-negative parameters α_i , β_i are called rate constants, and real-valued exponents g_{ij} (G matrix, $[G]$) and h_{ij} (H matrix, $[H]$) are referred to as kinetic order for synthesis and kinetic order for degradation, respectively.

$\{\alpha, \beta, [G], [H]\}$ are the parameters that define the S-system. The total number of parameters in the S-system is $2N(N+1)$. The parameter estimation is used to determine model parameters so that the dynamic profiles fit the observation.

Population based S-system model parameter identification

S-system based GRN inference was formulated by Tominaga et al. [57] as an optimization problem to minimize the difference between the model and the system. To guide the population in the search space, some measure of discrimination is needed. The most commonly used quality assessment criterion is the relative mean quadratic discrepancy between the observed expression pattern y_i^t and the model output $\hat{y}_i^t$ [39].

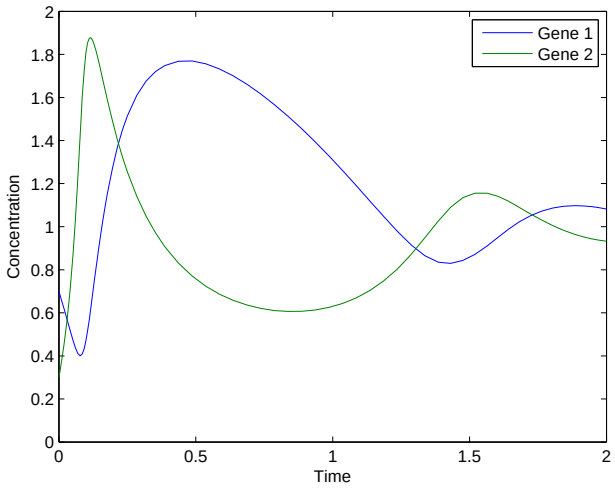

Figure 6.8: Target time dynamics of adopted gene network.

$$f = \sum_{i=1}^{N} \sum_{t=1}^{T} \left(\frac{\hat{y}_i^t - y_i^t}{y_i^t} \right)^2, \tag{6.18}$$

where T represents the number of time points.

To assess the performance of the methodologies studied here, a gene regulatory network consist of two genes generated by the parameters provided in Table 6.9 is adopted [57]. In the original implementation, the search space for α_i and β_i is limited to $[0.0,\ 20]$ and for g_{ij} and h_{ij} to $[-4.0,\ 4.0]$ and y_1^0 and y_2^0 are set at 0.7 and 0.3, respectively. The gene expression levels are plotted in Figure 6.8 and each consist of 50 time course of expression level per gene. To study the effect of initialization region on the converge of the optimization algorithms the initialization set to cover part of the search space. In this study, α_i and β_i is initialized in $[10,\ 20]$ and both g_{ij} and h_{ij} to $[2.0,\ 4.0]$.

Table 6.9: S-System Parameters adopted for model validation [57].

	Network parameters					
i	α_i	β_i	g_{i1}	g_{i2}	h_{i1}	g_{i2}
1	3	3	0	2.5	-1	0
2	3	3	-2.5	0	0	2

Results

The fitness transitions of studied methodologies are plotted in Figure 6.9. All algorithms discussed here start with a randomly generated population of solutions, which means they all start with close fitness values. The Figures are averaged over 30 independent runs.

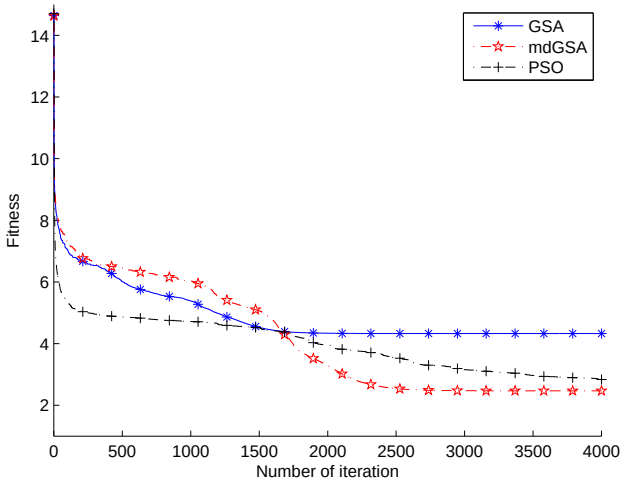

Figure 6.9: Performance comparison of the GSA, mdGSA and PSO on GRN parameter identification.

All three OAs start with a sharp fitness decrease in the beginning. GSA almost stagnates after approximately 2,000 iterations. mdGSA shows a much better progression compared to PSO and GSA.

As shown in Table 6.10, the results of the proposed mdGSA are better than those of GSA when the standard cut-off for considering a p-value for a statistically significant difference is set at $p < 0.05$. While mdGSA is not significantly superior to PSO, it shows a better performance, with a smaller standard deviation.

Table 6.10: A Wilcoxon Rank Sum test of the fitness of last generation for GRN parameter identification (30 runs).

	Simulation results		
NET1	Mean	std.	p-Value
PSO	2.8420	2.1192	0.3052
GSA	4.3252	1.6087	3.32E-6
mdGSA	2.4323	0.8699	-

6.6 Discussions

The heuristics we studied are compared on the basis of their robustness and accuracy (performance). We divided the evaluation of robustness into the following two assessments:

- When studying center-seeking bias, PSO is found to be the most appropriate opti-

mization algorithm (OA). It shows no observable CS bias. mdGSA comes second, followed by GSA. Statistical comparisons confirm that mdGSA holds less bias towards the center of the search space when compared to GSA.

- When studying initialization region bias, the performance of the PSO deteriorates (statistically) when the IR is tightened. This is consistent with existing literature [42] and confirms that an efficient swarm initialization improves performance. GSA also showed significant deterioration in its performance. mdGSA showed no statistical change in its performance as a result of shrinking the IR.

From the change in performance that comes from a change in the center of the search space, as well as a change in IR, we can conclude that mdGSA has less CS bias and less IRB compared to GSA. It is thus more robust.

From an exploration-exploitation perspective, ζ-RS provides us with better understanding of the behavior of the optimization algorithms. Algorithms with high IRB have limited abilities to explore promising regions outside of the IR. This is associated with the algorithm's weak exploration. Looking at the CS bias metric, GSA holds a strong search bias towards the center of the search space. It even does not have sufficient exploration to search beyond the initialization region, which lack is confirmed by the metric proposed to measure IRB. This puts into question the robustness of GSA. mdGSA, on the other hand, while it enjoys less center-seeking bias, has enough exploration which is confirmed by its statistically zero IRB. This is considered to be caused by the dispersed mass assignment procedure. Consequently, mdGSA has a high level of robustness.

As had been shown, mdGSA has a number of negative IRBs. This observation is in line with the statement made in [43]. There we see that the initial population is beneficial when it guides the population towards the global optimum, and that, whenever possible, the alleviation of the negative effects of this bias should be sought. Among the optimization techniques which we studied, mdGSA is the only one that takes advantage of the initialization region to guide the population.

The evaluation of accuracy is divided into the following two assessments:

- In low-dimensional optimization problems, both GSA and mdGSA outperform PSO. There is no significant difference between GSA and mdGSA when counting the number of times one is statistically superior to the others. This is confirmed by a statistical test. However, when we collect total number of worst solutions, mdGSA performs better than GSA.

- In high-dimensional optimization problems, mdGSA performs better than both PSO and GSA when we consider the number of times one is statistically superior to the others. The same results is achieved when total of worsts solutions are looked at.

Table 6.11 presents a summary of the comparison of the optimization heuristics examined in this study. It does so in terms of both their robustness and this accuracy. Robustness is compared by looking at the metrics presented to measure CSB and IRB. Accuracy is compared by looking at the quality of solutions found for benchmark optimization problems under two different settings, ξ-accuracy and ζ-accuracy. To summarize, in terms of robustness when the center of the search space is changed, PSO is the best of those we studied. In terms of robustness when changes are made in the initialization region, mdGSA

Table 6.11: Best of the studied optimization heuristics when looking at their Robustness and Accuracy.

	Robustness		ξ-Accuracy		ζ-Accuracy	
	CSB	IRB	S-test	Worst	S-test	Worst
Low D	PSO	mdGSA	GSA / mdGSA	mdGSA	GSA / mdGSA	mdGSA
High D	PSO	mdGSA	mdGSA	mdGSA	mdGSA	mdGSA

places first. When looking at ξ-accuracy and ζ-accuracy, GSA and mdGSA come joint first for low-dimensional problems, while mdGSA places first for high-dimensional problems. mdGSA comes in first place when looking at the number of times it has the worst fitness compared to the other contenders. High numbers of resulting in worst fitness suggests the susceptibility of PSO and GSA to becoming trapped in local optima.

Note that it is not in our interest to suggest not to use GSA because of its strong search bias. User should be aware, rather, of the way in which it might affect their needs. It is also notable that, in this work, the setting are those recommenced in original work. It must be noted, however, that changing the algorithmic parameter settings and stopping criteria, the benchmark functions, and even the grading criteria may change the results and conclusions. In spite of these caveats, we believe these preliminary results are a promising indication of the success of the proposed mdGSA on a wide range of optimization problems.

6.7 Conclusions and Future Work

Metaheuristics are a family of approximate methods used to find good solutions to computationally difficult optimization problems. While some optimization heuristics suffer from various types of search bias, a review of the literature reveals a lack of an appropriate quantification metric. The major contribution of this study is the development of metrics that measure the center-seeking and initialization region bias of optimization heuristics. We also propose an alternative for *center offset*, as we identified its assumption does not always hold.

Using the proposed metrics, the center-seeking (CS) bias and initialization region bias (IRB) of GSA are exposed. Our interest in this study was not to improve the performance of GSA, which can be archived by the integration of useful heuristics. Rather, it was about presenting a solution to dilute its CS behavior and its IRB. Inspired by our recently introduced global optimization process, we established a "mass-dispersed" version of GSA called mdGSA. PSO served as our benchmark because it shows no bias towards the center of the search space [26].

To further substantiate the limitations and capabilities of GSA and mdGSA in dealing with real-world optimization problems, we want to apply them to a wider range of problems, such as structural design optimization [15], the detection of hidden information in a spread-spectrum watermarked signal [10], and problems of traffic control [12].

Several optimization heuristics have evolved in the last decade to facilitate solving op-

timization problems (see for example [21, 37, 48]), some of which suffer from different types of search bias [59]. The framework presented in this study appears to be a viable approach when it comes to comparing different optimization heuristics. As part of our future work, we are interested in using the framework proposed here to contrast different optimization heuristics suitable to handling high-dimensional and complex real-world optimization problems.

References

[1] Angeline, P. (1998). Using selection to improve particle swarm optimization. In *International Conference on Evolutionary Computation*, pages 84–89.

[2] Branke, J. (2001). Evolutionary approaches to dynamic optimization problems-updated survey. In *GECCO Workshop on evolutionary algorithms for dynamic optimization problems*, pages 27–30.

[3] Bratton, D. and Kennedy, J. (2007). Defining a standard for particle swarm optimization. In *Swarm Intelligence Symposium*, pages 120–127.

[4] Chatterjee, A., Mahanti, G., and Pathak, N. (2010). Comparative performance of gravitational search algorithm and modified particle swarm optimization algorithm for synthesis of thinned scanned concentric ring array antenna. *Progress In Electromagnetics Research B*, 25:331–348.

[5] Chiong, R., Weise, T., and Michalewicz, Z. (2011). *Variants of evolutionary algorithms for real-world applications*. Springer-Verlag New York Inc.

[6] Clerc, M. and Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73.

[7] Coello, C. A. C. (2002). Theoretical and numerical constraint handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287.

[8] Cohen, J., Cohen, P., West, S., and Aiken, L. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences*. Lawrence Erlbaum Associates.

[9] Crombach, A. and Hogeweg, P. (2008). Evolution of evolvability in gene regulatory networks. *PLoS computational biology*, 4(7):e1000112.

[10] Davarynejad, M., Ahn, C., Vrancken, J., van den Berg, J., and Coello Coello, C. (2010). Evolutionary hidden information detection by granulation-based fitness approximation. *Applied Soft Computing*, 10(3):719–729.

[11] Davarynejad, M., Forghany, Z., and van den Berg, J. (2012a). Mass-dispersed gravitational search algorithm for gene regulatory network model parameter identification. In *Simulated Evolution and Learning (SEAL'12)*, pages 62–72.

[12] Davarynejad, M., Hegyi, A., Vrancken, J., and van den Berg, J. (2011a). Motorway ramp-metering control with queuing consideration using *Q*-learning. In *14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1652–1658.

[13] Davarynejad, M., Rezaei, J., Vrancken, J., van den Berg, J., and Coello, C. C. (2011b). Accelerating convergence towards the optimal pareto front. In *IEEE Congress on Evolutionary Computation (CEC'11)*, pages 2107–2114.

[14] Davarynejad, M. and van den Berg, J. (2012). Simulated big bounce: a continuous space global optimizer. Technical report, Faculty of technology policy and management, Delft University of Technology, The Netherlands.

[15] Davarynejad, M., Vrancken, J., van den Berg, J., and Coello Coello, C. (2012b). A Fitness Granulation Approach for Large-Scale Structural Design Optimization. In Chiong, R., Weise, T., and Michalewicz, Z., editors, *Variants of Evolutionary Algorithms for Real-World Applications*, pages 245–280. Springer-Verlag, Berlin.

[16] David, R., Precup, R., Petriu, E., Rădac, M., and Preitl, S. (2013). Gravitational search algorithm-based design of fuzzy control systems with a reduced parametric sensitivity. *Information Sciences*, 247(20):154–173.

[17] Deb, K., Anand, A., and Joshi, D. (2002). A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary computation*, 10(4):371–395.

[18] Dorigo, M. and Caro, G. D. (1999). Ant colony optimization: a new meta-heuristic. In *IEEE Congress on Evolutionary Computation (CEC'99)*, pages 1470–1477.

[19] Duman, S., Güvenç, U., and Yörükeren, N. (2010). Gravitational search algorithm for economic dispatch with valve-point effects. *International Review of Electrical Engineering (IREE)*, 5(6):2890–2895.

[20] Forghany, Z., Davarynejad, M., and Snaar-Jagalska, B. (2012). Gene regulatory network model identification using artificial bee colony and swarm intelligence. In *IEEE Conference on Evolutionary Computation (CEC'12)*, pages 949–954.

[21] Ghosh, S., Das, S., Kundu, D., Suresh, K., and Abraham, A. (2012). Inter-particle communication and search-dynamics of *lbest* particle swarm optimizers: An analysis. *Information Sciences*, 182(1):156–168.

[22] Glover, F. and Kochenberger, G. (2003). *Handbook of metaheuristics*. Springer.

[23] Güvenç, U., Sönmez, Y., Duman, S., and Yörükeren, N. (2012). Combined economic and emission dispatch solution using gravitational search algorithm. *Scientia Iranica*, 19(6):1754–1762.

[24] Holland, J. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI.

[25] Karaboga, D. and Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization*, 39(3):459–471.

[26] Kennedy, J. (2007). Some issues and practices for particle swarms. In *IEEE Swarm Intelligence Symposium (SIS'07)*, pages 162–169. IEEE.

[27] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948.

[28] Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K., and Tomita, M. (2003). Dynamic modeling of genetic networks using genetic algorithm and s-system. *Bioinformatics*, 19(5):643–650.

[29] Kimura, S., Ide, K., Kashihara, A., Kano, M., Hatakeyama, M., Masui, R., Nakagawa, N., Yokoyama, S., Kuramitsu, S., and Konagaya, A. (2005). Inference of s-system models of genetic networks using a cooperative coevolutionary algorithm. *Bioinformatics*, 21(7):1154.

[30] Lee, W. and Tzou, W. (2009). Computational methods for discovering gene networks from expression data. *Briefings in bioinformatics*, 10(4):408–423.

[31] Li, C. and Zhou, J. (2011). Parameters identification of hydraulic turbine governing system using improved gravitational search algorithm. *Energy Conversion and Management*, 52(1):374–381.

[32] Lopez-Molina, C., Bustince, H., Fernandez, J., Couto, P., and Baets, B. D. (2010). A gravitational approach to edge detection based on triangular norms. *Pattern Recognition*, 43(11):3730–3741.

[33] Mariani, V. and Coelho, L. (2011). A hybrid shuffled complex evolution approach with pattern search for unconstrained optimization. *Mathematics and Computers in Simulation*, 81(9):1901–1909.

[34] Mezura-Montes, E., editor (2009). *Constraint-Handling in evolutionary optimization*. Springer, Berlin, Germany.

[35] Mitchell, T. (1997). *Machine learning*. McGraw Hill.

[36] Monson, C. and Seppi, K. (2005). Exposing origin-seeking bias in pso. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 241–248.

[37] Nasir, M., Das, S., Maity, D., Sengupta, S., Halder, U., and Suganthan, P. (2012). A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization. *Information Sciences*, 209:16–36.

[38] Navlakha, S. and Bar-Joseph, Z. (2011). Algorithms in nature: the convergence of systems biology and computational thinking. *Molecular Systems Biology*, 7(1).

[39] Noman, N. and Iba, H. (2005). Inference of gene regulatory networks using s-system and differential evolution. In *Genetic and Evolutionary Computation Conference, Washington, DC*, pages 439–446.

[40] Ono, I., Kita, H., and Kobayashi, S. (1999). A robust real-coded genetic algorithm using unimodal normal distribution crossover augmented by uniform crossover: Effects of self-adaptation of crossover probabilities. In *Genetic and Evolutionary Computation (GECCO'99)*, pages 496–503.

[41] Pal, K., Saha, C., Das, S., and Coello, C. C. (2013). Dynamic constrained optimization with offspring repair based gravitational search algorithm. In *2013 IEEE Congress on Evolutionary Computation (CEC'13)*, pages 2414–2421.

[42] Pant, M., Thangaraj, R., and Abraham, A. (2009). Particle swarm optimization: Performance tuning and empirical analysis. In Abraham, A., Hassanien, A., Siarry, P., and Engelbrecht, A., editors, *Foundations of Computational Intelligence Volume 3*, volume 203 of *Studies in Computational Intelligence*, pages 101–128. Springer Berlin / Heidelberg.

[43] Pelikan, M. and Sastry, K. (2009). Initial-population bias in the univariate estimation of distribution algorithm. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 429–436.

[44] Precup, R., David, R., Petriu, E., Preitl, S., and Paul, A. (2011). Gravitational search algorithm-based tuning of fuzzy control systems with a reduced parametric sensitivity. *Soft Computing in Industrial Applications*, pages 141–150.

[45] Price, K., Storn, R., and Lampinen, J. (2005). *Differential evolution: a practical approach to global optimization*. Springer Natural Computing Series.

[46] Rashedi, E., Nezamabadi-Pour, H., and Saryazdi, S. (2009). Gsa: a gravitational search algorithm. *Information Sciences*, 179(13):2232–2248.

[47] Rashedi, E., Nezamabadi-Pour, H., and Saryazdi, S. (2011). Filter modeling using gravitational search algorithm. *Engineering Applications of Artificial Intelligence*, 24(1):117–122.

[48] Rowhanimanesh, A. and Akbarzadeh-T, M. (2011). Perception-based heuristic granular search: Exploiting uncertainty for analysis of certain functions. *Scientia Iranica*, 18(3):617–626.

[49] Runarsson, T. P. (2004). Constrained evolutionary optimization by approximate ranking and surrogate models. In *Proceedings of 8th Parallel Problem Solving From Nature (PPSN VIII)*, pages 401–410.

[50] Runarsson, T. P. and Yao, X. (2000). Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294.

[51] Sastry, K., Goldberg, D., and Pelikan, M. (2001). Dont evaluate, inherit. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 551–558.

[52] Schlitt, T. (2013). Approaches to modeling gene regulatory networks: A gentle introduction. In *In Silico Systems Biology*, pages 13–35.

[53] Shang, Y., , and Qiu, Y. (2006). A note on the extended rosenbrock function. *Evolutionary Computation*, 14(1):119–126.

[54] Shaw, B., Mukherjee, V., and Ghoshal, S. (2012). A novel opposition-based gravitational search algorithm for combined economic and emission dispatch problems of power systems. *International Journal of Electrical Power & Energy Systems*, 35(1):21–33.

[55] Sîrbu, A., Ruskin, H., and Crane, M. (2010). Comparison of evolutionary algorithms in gene regulatory network model inference. *BMC Bioinformatics*, 11(59).

[56] Storn, R. and Price, K. (1995). Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute, Berkley.

[57] Tominaga, D., Okamoto, M., Maki, Y., Watanabe, S., and Eguchi, Y. (1999). Nonlinear numerical optimization technique based on a genetic algorithm for inverse problems: Towards the inference of genetic networks. In *German Conference on Bioinformatics Computer Science and Biology*, pages 127–140.

[58] Tsai, K. and Wang, F. (2005). Evolutionary optimization with data collocation for reverse engineering of biological networks. *Bioinformatics*, 21(7):1180.

[59] Whitacre, J. (2011). Recent trends indicate rapid growth of nature-inspired optimization in academia and industry. *Computing*, 93(2):121–133.

[60] Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.

[61] Yao, X., Liu, Y., and Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3(2):82–102.

Table 6.12: Test problems used in the experiments. U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable

Function name	Mathematical Representation	Characteristic	Search space
Ackley	$f_1(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	MN	$[-30,30]^D$
Dixon-Price	$f_2(x) = (x_1-1)^2 + \sum_{i=2}^{D} i\left(2x_i^2 - x_{i-1}\right)^2$	UN	$[-10,10]^D$
Griewank	$f_3(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	MN	$[-600,600]^D$
Levy	$f_4(x) = \sin^2(\pi y_1) + \sum_{i=1}^{D}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right] + (y_D-1)^2$, $y_i = 1 + \frac{x_i-1}{4}$	MN	$[-10,10]^D$
Penalized1	$f_5(x) = \frac{\pi}{D}\{10\sin(\pi y_1) + \sum_{i=1}^{D}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})] + (y_D-1)^2\} + \sum_{i=1}^{D} u(x_i,10,100,4)$, $y_i = 1+\frac{x_i+1}{4}$, $u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m; & x_i > 0 \\ 0; & -a < x_i < a \\ k(-x_i-a)^m; & x_i < 0 \end{cases}$	MN	$[-50,50]^D$
Penalized2	$f_6(x) = .1\{\sin^2(3\pi x_1) + \sum_{i=1}^{D}(x_i-1)^2[1+10\sin^2(3\pi x_i+1)] + (x_D-1)^2[1+sin^2(2\pi x_n)]\} + \sum_{i=1}^{D} u(x_i,5,100,4)$	MN	$[-50,50]^D$
Quartic	$f_7(x) = \sum_{i=1}^{D} i x_i^4$	US	$[-1.28,1.28]^D$
Rastrigin	$f_8(x) = \sum_{i=1}^{D}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	MS	$[-5.12,5.12]^D$
Rosenbrock	$f_9(x) = \sum_{i=1}^{D-1}[100\left(x_i^2 - x_{i+1}\right)^2 + (x_i-1)^2]$	MN	$[-50,50]^D$
Schwefel P2.22	$f_{10}(x) = \sum_{i=1}^{D}\lvert x_i\rvert + \prod_{i=1}^{D}\lvert x_i\rvert$	UN	$[-10,10]^D$
Schwefel P1.2	$f_{11}(x) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$	UN	$[-100,100]^D$
Schwefel P2.21	$f_{12}(x) = \max\{\lvert x_i\rvert, 1 \le i \le n\}$	US	$[-100,100]^D$
Sphere	$f_{13}(x) = \sum_{i=1}^{D} x_i^2$	US	$[-100,100]^D$
Step	$f_{14}(x) = \sum_{i=1}^{D}(\lfloor x_i + .5\rfloor)^2$	US	$[-100,100]^D$

"Not every end is the goal. The end of a melody is not its goal, and yet if a melody has not reached its end, it has not reached its goal. A parable."

Friedrich Nietzsche

7

Conclusions and future research

This chapter provides a brief summary of the findings and contributions of this thesis, followed by a discussion of directions for future research.

Metaheuristics are amongst a set of well-known and widely used techniques for real-world optimization problems. Depending on the factors responsible for the increased complexity of the problems we are dealing with, there are at least two solutions to improve metaheuristics: a) Reduction of computational complexity related to high computation costs of fitness evaluations and b) More effective search strategies by improved balancing of exploration and exploitation. In this thesis we have taken up this challenge not only by reducing computational cost using techniques such as *fitness approximation*, but also by developing a *new metaheuristic* as well as proposing *metrics* for measuring certain search biases that directly pertinent to the problem of making the best possible selection of solvers.

Fitness Approximation

Population-based metaheuristics are ruled by the competitive nature of the "survival of the fittest", a process that is robust against uncertainties in fitness value, as long as a proper ranking of candidate solutions is preserved. This robustness implies that a proper approximation of the fitness of candidate solutions may replace the exact fitness calculation. When exact fitness evaluation is computationally expensive, fitness approximation is a natural approach, reducing, as it does, the computational cost.

Granulation of information is a satisfactory way of handling *information* by abstraction at a level of coarseness suited to providing appropriate and sufficient input for problem solving. Graduated granulation of information was proposed by Zadeh in 1979, and is a technique by which a class of objects are partitioned into granules. This is a process whereby human reasoning can be formed, organized and manipulated so as to handle complexity and imprecision. In order to exploit evolutionary algorithms' natural tolerance for imprecision in fitness values, in **Chapter 2** this concept of fuzzy granulation is adopted to reduce the

overall computational cost. It does so by computing exact fitness selectively and only in the event that it has deficient similitude to a pool of granules whose true fitness has been registered. The proposed method is adaptive in the sense that the feedback from the current population is used to determine the granules *radius of influence*. Moreover, the minimum similarity a candidate solution needs to share with the granule pool changes in a way designed to encourage fewer exact fitness evaluations in initial stages, and more exact fitness evaluations in later stages of evolution. In these later stages, competition is fierce amongst similar and converging solutions, meaning that the level of approximation changes over time. Moreover, the mechanism that is embedded to control the size of the granule pool, replaces specific granules with new ones adaptively. That is why the proposed approach is referred to as *adaptive fuzzy fitness granulation* (AFFG). Statistical analysis reveals that the proposed method significantly decreases the number of fitness function evaluations while finding equally acceptable, or even better, solutions when applied to a set of benchmark problems and hardware design problems.

Most metaheuristics include some tuning parameters that influence the optimization algorithm. The selection of these algorithm parameters is, to a large extent, empirical. AFFG is not an exception. When using fitness approximation, it is crucial to have an accurate estimation of the fitness function of the individuals in the finishing generations. In the case of AFFG, this can be accomplished by controlling the radius of influence of the granules. This radius is the width of the membership function (WMFs). During the early steps of evolution, and by choosing relatively large WMFs, the algorithm accepts individuals with a lower degree of similarity. Fitness is therefore more often estimated. As the individuals mature and reach higher fitness values, the width decreases, implying that, in order for the fitness of an individual to be approximated, its similarity to the granule pool should increase. This prompts a higher selectivity and a higher threshold for fitness estimation. In later generations, the degree of similarity between two individuals must be larger than that in the early generations, in order for them to be accepted as similar individuals. This procedure ensures a fast convergence rate, due to rapid computation in the early phase and accurate fitness estimation in the later stage.

To achieve these desiderata without having to tune the parameters manually, a fuzzy supervisor with three inputs is employed in **Chapter 3**. During the AFFG search, the fuzzy logic controller observes the Number of Design Variables (NDV), the Maximum Range of Design Variables (MRDV) and the percentage of completed trials. It also specifies the WMFs. The combined effect of granule enlargement/shrinkage is to realize both rapid computation and accurate fitness estimation. Instead of one controller, and in order to reduce the large number of rules and the extraction of rules, it is separated in two controllers. This diminishes the complexity of the system and so reduces the number of rules. The effectiveness of the proposed controller is investigated with a number of optimization benchmarks: four different choices are given for the dimensionality of the search space. The effect of the number of granules on the rate of convergence is also studied. The proposed method is then applied to the hidden information detection problem to recover a pseudo noise (PN) sequence with a chip period equal to 63, 127 and 255 bits. In comparison with the standard application of EAs, experimental analysis confirms that the proposed approach has the ability to considerably reduce the computational complexity of the detection problem, and to do so without compromising performance.

Real-world problems frequently have two, or more, (possibly conflicting) objectives that

we aim to optimize at the same time. Such problems are called multiobjective problems and have been studied intensively using metaheuristics (particularly, evolutionary algorithms) over the last few years. In multiobjective optimization problems there is normally no single solution that is best for all the criteria. Rather, there exists a set of solutions for which no objective can be improved without worsening another. This is known as a Pareto optimal set. When plotted in an objective function space, these solutions are collectively known as the Pareto front.

Multiobjective evolutionary algorithms (MOEAs) are known to be computationally expensive, since they normally require a high number of objective function evaluations in order to produce a reasonably good approximation of the Pareto front of the problem being solved. Nevertheless, relatively little research has been reported so far on the development of techniques that reduce the computational cost of MOEAs. **Chapter 4** of this thesis contributes to this area by adapting the proposed AFFG for reducing the number of objective function evaluations required by a MOEA. The proposed approach is compared with respect to the standard NSGA-II, using the *Set Coverage*, *Hypervolume* and *Generational Distance* performance measures. The results indicate that the proposed approach is a very promising alternative for dealing with multiobjective optimization problems that involve expensive fitness function evaluations.

Looking for efficient algorithms

Genetic algorithms, with their principles rooted in natural selection, are among the first optimization algorithms inspired by nature that deviate widely from the working principles of classical optimization algorithms with superior characteristics. According to the "No Free Lunch" theorem, any elevated performance of an optimization algorithm over one class of problems is offset by the performance over another class. This suggests that we look for an optimization algorithm that surpasses the performance of others on a specific class of problems. And indeed, this is the reasoning that led to the development of a vast number of optimization algorithms, e.g. particle swarm optimization. By deploying this idea, **Chapter 5** endeavors to develop an optimization algorithm inspired by the *Big Bounce* theorem, and that is competitive over a class of high dimensional optimization problems.

Metrics

In practice, metaheuristics suffer from different types of search bias, the understanding of which is of crucial importance when it comes to making the best possible choice for a given problem. **Chapter 6** of this thesis introduced two metrics, one for measuring center-seeking bias (CSB) and one for initialization region bias (IRB). The proposed metric to measure CSB is based on the well-known *center offset*, and the metric to measure IRB is proposed on the grounds of *region scaling*. A framework is presented in this study that considers both *accuracy* and *robustness* when different optimization heuristics are compared. The proposed framework is used to evaluate the bias of three metaheuristics.

The first metaheuristic studied is particle swarm optimization (PSO), and is chosen as a benchmark because it shows no bias towards the center of the search space. The second metaheuristic studied is the gravitation search algorithm (GSA), mainly because of its unexpectedly poor results on the benchmark problems studied in Chapter 5. The most prominent finding is the considerable CSB and IRB of the gravitational search algorithm (GSA). Inspired by the mass assignment procedure of SBB presented in Chapter 5, a partial solution

to the center-seeking and initialization region biases is proposed in Chapter 6. This modified GSA is referred to as *mass-dispersed* GSA (mdGSA). The performance of mdGSA, which promotes the global search capability of GSA, is verified using the same mathematical optimization problems as in Chapter 5.

The evaluation of robustness is divided into the following two assessments:

- When studying center-seeking bias, PSO is found to be the most appropriate optimization algorithm with no observable CS bias, while mdGSA comes second, followed by GSA. Statistical comparisons confirm that mdGSA holds less bias towards the center of the search space compared to GSA.
- When studying initialization region bias, the performance of PSO statistically deteriorates when the initialization region is tightened. This is consistent with existing literature and confirms that an efficient swarm initialization improves performance. GSA also showed significant deterioration in its performance. mdGSA showed no statistical change in its performance as a result of shrinking the initialization region.

The evaluation of accuracy is divided into the following two assessments:

- In low-dimensional optimization problems, both GSA and mdGSA outperform PSO. There is no significant difference between GSA and mdGSA when counting the number of times one is statistically superior to the others. This is confirmed by statistical testing. However, when looking at the number of worst solutions, mdGSA performs better than GSA.
- In high-dimensional optimization problems, mdGSA performs better than either PSO or GSA when counting the number of times one is statistically superior to the others, as well as when looking at the number of worst solutions.

The results of the gene regulatory network system parameter identification demonstrates the capabilities of the mdGSA in solving real-world optimization problems.

7.1 Directions for Future Research

This thesis presents a number of solutions in order to reduce the computational cost of complex optimization problems, either through fitness approximation or through the development of a metaheuristic suitable to a class of problems at hand. Moreover, when studying the properties of metaheuristics, it turned out that some of them suffer from a notable, specific search bias. To remedy this unfairness, two generic metrics have been developed to evaluate the search biases of different algorithms. While the contributions presented in this study are exciting, what is perhaps even more exciting is the fact that this study has generated more ideas than could conceivably be handled, even given ample time and worlds. Some of which follows.

Fitness Approximation

- AFFG is deterministic, meaning that when it is plugged into a deterministic search algorithm, the combined algorithm will remain deterministic. It is not clear if the

ideal approach for controlling the size of granules is the removal of the granule with the lowest life index, or the (as yet to be developed) stochastic selection of a granule to be removed from the granule pool. Another deterministic aspect of AFFG is the granules' radius of influence. The effect of introducing stochasticity to the granules' radius of influence, and its effect on the overall performance of optimization search algorithms, is an open question.

- Approximate models are, in general, able to preserve the history of optimization. AFFG preserves the history of a search through the pool of granules, and uses this history to associate the fitness of granules to a candidate solution. A mean of the fitness value of granules is another way to assign a fitness to a candidate solution, when weighted by its degree of similarity to the granules. This may improve the performance of the search, and deserves to be studied in future work.

Looking for efficient algorithms

- Given the success demonstrated by SBB on high-dimensional optimization problems, there is a need for it to be evaluated properly on real-world problems.

- When the objective function is noisy (i.e. each solution has different objective values over time, e.g. in uncertain environments), for many metaheuristics optimization tends to be difficult. The greedy selection mechanism in some optimization algorithms is, in part, responsible for attracting the population to unproductive locations of the search space. While many of the existing metaheuristics suffer from the same problem in various forms, it is of interest to see how the performance of SBB changes under such conditions.

- In multiobjective optimization problems, there is normally no single solution that is best for all the criteria. Rather, there exists a set of solutions for which no objective can be improved without worsening another, known as the Pareto optimal set. When plotted in objective function space, these solutions are known collectively as the Pareto front. Multiobjective optimization has been studied intensively using metaheuristics (particularly, evolutionary algorithms) over the last few years. Given that many real-word problems are multiobjective, something that deserves special attention is the studying of the performance of SBB on real-world problems where two or more objectives have to be optimized at the same time.

Metrics

- Several optimization heuristics have evolved in the last decade to facilitate solving optimization problems, some of which suffer from different types of search bias. The framework presented in Chapter 6 of this study appears to be a viable approach to the comparison of different optimization heuristics. It is of interest, using the framework proposed here, to contrast different optimization heuristics suitable to the handling of both high-dimensional and complex real-world optimization problems.

TRAIL Thesis Series

The following list contains the most recent dissertations in the TRAIL Thesis Series. For a complete overview of more than 100 titles see the TRAIL website: www.rsTRAIL.nl.

The TRAIL Thesis Series is a series of the Netherlands TRAIL Research School on transport, infrastructure and logistics.

Davarynejad, M., *Deploying Metaheuristics for Global Optimization*, T2014/4, June 2014, TRAIL Thesis Series, The Netherlands

Li, J., *Characteristics of Chinese Driver Behavior*, T2014/3, June 2014, TRAIL Thesis Series, the Netherlands

Mouter, N., *Cost-Benefit Analysis in Practice: A study of the way Cost-Benefit Analysis is perceived by key actors in the Dutch appraisal practice for spatial-infrastructure projects*, T2014/2, June 2014, TRAIL Thesis Series, the Netherlands

Ohazulike, A., *Road Pricing mechanism: A game theoretic and multi-level approach*, T2014/1, January 2014, TRAIL Thesis Series, the Netherlands

Cranenburgh, S. van, *Vacation Travel Behaviour in a Very Different Future*, T2013/12, November 2013, TRAIL Thesis Series, the Netherlands

Samsura, D.A.A., *Games and the City: Applying game-theoretical approaches to land and property development analysis*, T2013/11, November 2013, TRAIL Thesis Series, the Netherlands

Huijts, N., *Sustainable Energy Technology Acceptance: A psychological perspective*, T2013/10, September 2013, TRAIL Thesis Series, the Netherlands

Zhang, Mo, *A Freight Transport Model for Integrated Network, Service, and Policy Design*, T2013/9, August 2013, TRAIL Thesis Series, the Netherlands

Wijnen, R., *Decision Support for Collaborative Airport Planning*, T2013/8, April 2013, TRAIL Thesis Series, the Netherlands

Wageningen-Kessels, F.L.M. van, *Multi-Class Continuum Traffic Flow Models: Analysis and simulation methods*, T2013/7, March 2013, TRAIL Thesis Series, the Netherlands

Taneja, P., *The Flexible Port*, T2013/6, March 2013, TRAIL Thesis Series, the Netherlands

Yuan, Y., *Lagrangian Multi-Class Traffic State Estimation*, T2013/5, March 2013, TRAIL

Thesis Series, the Netherlands

Schreiter, Th., *Vehicle-Class Specific Control of Freeway Traffic*, T2013/4, March 2013, TRAIL Thesis Series, the Netherlands

Zaerpour, N., *Efficient Management of Compact Storage Systems*, T2013/3, February 2013, TRAIL Thesis Series, the Netherlands

Huibregtse, O.L., *Robust Model-Based Optimization of Evacuation Guidance*, T2013/2, February 2013, TRAIL Thesis Series, the Netherlands

Fortuijn, L.G.H., *Turborotonde en turboplein: ontwerp, capaciteit en veiligheid*, T2013/1, January 2013, TRAIL Thesis Series, the Netherlands

Gharehgozli, A.H., *Developing New Methods for Efficient Container Stacking Operations*, T2012/7, November 2012, TRAIL Thesis Series, the Netherlands

Duin, R. van, *Logistics Concept Development in Multi-Actor Environments: Aligning stakeholders for successful development of public/private logistics systems by increased awareness of multi-actor objectives and perceptions*, T2012/6, October 2012, TRAIL Thesis Series, the Netherlands

Dicke-Ogenia, M., *Psychological Aspects of Travel Information Presentation: A psychological and ergonomic view on travellers' response to travel information*, T2012/5, October 2012, TRAIL Thesis Series, the Netherlands

Wismans, L.J.J., *Towards Sustainable Dynamic Traffic Management*, T2012/4, September 2012, TRAIL Thesis Series, the Netherlands

Hoogendoorn, R.G., *Swiftly before the World Collapses: Empirics and Modeling of Longitudinal Driving Behavior under Adverse Conditions*, T2012/3, July 2012, TRAIL Thesis Series, the Netherlands

Carmona Benitez, R., *The Design of a Large Scale Airline Network*, T2012/2, June 2012, TRAIL Thesis Series, the Netherlands

Schaap, T.W., *Driving Behaviour in Unexpected Situations: A study into the effects of drivers' compensation behaviour to safety-critical situations and the effects of mental workload, event urgency and task prioritization*, T2012/1, February 2012, TRAIL Thesis Series, the Netherlands

Muizelaar, T.J., *Non-recurrent Traffic Situations and Traffic Information: Determining preferences and effects on route choice*, T2011/16, December 2011, TRAIL Thesis Series, the Netherlands

Cantarelli, C.C., *Cost Overruns in Large-Scale Transportation Infrastructure Projects: A theoretical and empirical exploration for the Netherlands and Worldwide*, T2011/15, November 2011, TRAIL Thesis Series, the Netherlands

Vlies, A.V. van der, *Rail Transport Risks and Urban Planning: Solving deadlock situations between urban planning and rail transport of hazardous materials in the Netherlands*, T2011/14, October 2011, TRAIL Thesis Series, the Netherlands

Summary

Deploying Metaheuristics for Global Optimization

Global optimization is an active research topic in many areas including engineering, business, social sciences and mathematics. With the advent of new optimization algorithms, the set of solvable optimization problems grows steadily and the area of applications widens. Optimization problems encountered in practice appear in various types and with various mathematical properties. According to the No-Free-Lunch (NFL) theorem it is impossible to design a general-purpose universal optimization strategy. This implies that for optimization algorithms to solve a problem efficiently, they have to be tailored to the problem-specific characteristics. However since there are too many factors to be considered, it is often hard to accomplish this task by an analytical method. Therefore, in practice a trial-and-error method is used instead. In this thesis we have taken up this challenge by trying out a number of metaheuristics.

Evolutionary algorithms (EAs) have been very popular optimization methods for a wide variety of applications. However, in spite of their advantages, their computational cost is still a prohibitive factor in certain real-world applications involving computationally expensive fitness function evaluations. In Chapter 2, we adopt the observation that nature's survival of the fittest is not about exact measures of fitness; rather it is about correct ranking of competing peers. Thus, by exploiting this natural tolerance for imprecision, we propose a fuzzy granules-based approach for reducing the number of necessary function calls. The approach is based on adaptive fuzzy fitness granulation having as its main aim to strike a balance between the accuracy and the utility of the computations. The adaptation algorithm adjusts the radii of influence of granules according to the perceived performance and level of convergence attained. Experimental results show that the proposed approach accelerates the convergence towards optimal solutions, when compared to the performance of other more popular approaches. This suggests its applicability to other complex real-world problems. The proposed solution does not have the drawbacks of existing solutions for fitness approximations, such as time consuming online training.

The solution proposed in Chapter 2 has a number of tuning parameters that are problem dependent. In practice, a number of trials is needed to adjust these parameters. In Chapter 3 we propose a fuzzy supervisor as an auto-tuning strategy, in order to avoid the tuning of parameters. Its effectiveness is investigated with three traditional optimization benchmarks of four different choices for the dimensionality of the search space. The effect of the number of granules on the rate of convergence is also studied. The proposed method is then applied to the hidden information detection problem to recover a pseudo noise sequence with a chip

period equal to 63, 127 and 255 bits. In comparison with the standard application of EA, experimental analysis confirms that the proposed approach has an ability to considerably reduce the computational complexity of the detection problem without compromising performance. Furthermore, the auto-tuning of the fuzzy supervisor removes the need for exact parameter determination.

In Chapter 2 we have introduced a solution to accelerate the convergence towards the optimal solution in a single objective function setting. In real-world problems, however, the number of objectives are often two or more. In Chapter 4 we extend the solution presented in Chapter 2 for the case where the problem at hand is multi-objective. Our proposed approach is compared to the standard NSGA-II, using the *Set Coverage*, *Hypervolume* and *Generational Distance* performance measures. Our results indicate that our proposed approach is a promising alternative for dealing with multi-objective optimization problems involving expensive fitness function evaluations.

Some evolutionary computing techniques have advantages over others in terms of ease of implementation, preservation of diversity of the population, efficiency, etc. For advancement of their performance they may be simplified, hybridized etc. There has also been a steady increase in the number of global optimization algorithms, each characterized by its unique population dynamics. Different population dynamics characterizes the way two conflicting goals are balanced, exploration (diversification) and exploitation (intensification). These algorithms were constructed to address the need for faster optimization algorithms. Although existing metaheuristics are suitable for complex optimization problems, their convergence deteriorates when the complexity increases. Metaheuristics apply a search strategy that balances exploration and exploitation in an algorithm-specific way. It is observed that metaheuristic algorithms in practice often find local minima, sometimes of low quality, meaning that the chosen balance is inadequate to the problem at stake. For example, due to an algorithm's search bias, too great an emphasis may be placed on the exploitation of solutions found, while little attention is paid to the further exploration of the search space as a whole. Based on these observations, and inspired by the *Big Bounce* theory (a cosmological oscillatory model of the Universe), in Chapter 5 we developed a Simulated Big Bounce (SBB) algorithm that, next to exploitation, applies robust exploration in order to escape from local minima. This paper presents the design of this new algorithm and shows the results of a series of comparative experiments in which the performance of SBB on a set of high-dimensional mathematical benchmarks is compared to that of five other popular metaheuristics. The results obtained indicate that the proposed algorithm (*i*) is competitive with (and in most cases surpasses) other population-based optimization algorithms, and (*ii*) substantially decreases the number of fitness function evaluations needed to find equally good solutions. Although SBB has features in common with existing optimization methods, such as particle swarm optimization (PSO), it possesses additional unique features. These owe to the diverse kinetic energy of particles, and enable the algorithm to escape from local minima. Furthermore, the experimental outcomes provide evidence that the characteristic of robust exploration, which marks SBB, underlies the superior performance observed.

When comparing various optimization strategies, we have observed that in practice, metaheuristics suffer from various types of search bias, the understanding of which is directly pertinent to the problem of making the best possible selection of solvers. In Chapter 6, two metrics are introduced: one for measuring center-seeking bias (CSB) and one for initialization region bias (IRB). The former is based on ξ-*center offset*, an alternative to *center*

offset, which is a common but inadequate approach to analyzing the center-seeking behavior of algorithms, as is shown. IRB is proposed on the grounds of *region scaling*. The introduced metrics are used to evaluate the bias of three algorithms while running on a test bed of optimization problems having their optimal solution at, or near, the center of the search space. The most prominent finding of this paper is considerable CSB and IRB in the gravitational search algorithm (GSA). In addition, a partial solution to the center-seeking and initialization region bias of GSA is proposed by introducing a *mass-dispersed* version of GSA: mdGSA. This promotes the global search capability of GSA. Its performance is verified using the same test bed, next to a gene regulatory network parameter identification problem. The results of these experiments demonstrate the capabilities of mdGSA in solving real-world optimization problems.

We finally present the main finding of this thesis in Chapter 7. We close by suggestions for future research.

Samenvatting

Deploying Metaheuristics for Global Optimization

Globale optimalisatie is een actief onderzoeksgebied in vele disciplines, waaronder technische, bedrijfskundige en sociale wetenschappen en wiskunde. Met de komst van nieuwe optimalisatie-algoritmen groeit de verzameling van oplosbare optimalisatieproblemen gestaag en wordt het toepassingsgebied breder. Optimalisatieproblemen uit de praktijk komen voor in diverse typen en met diverse wiskundige eigenschappen. Volgens de Geen-Gratis-Lunch stelling is het onmogelijk om een algemeen bruikbare, universele optimalisatiestrategie te ontwerpen. Dit impliceert dat een optimalisatie-algoritme een probleem alleen efficint kan oplossen als het afgestemd wordt op de specifieke eigenschappen van dat probleem. Door het grote aantal factoren waarmee rekening gehouden moet worden, is het vaak moeilijk om dit afstemmen met een analytische methode uit te voeren. Daarom wordt in de praktijk een trial-and-error methode gebruikt. In dit proefschrift hebben we deze uitdaging opgepakt door een aantal metaheuristieken te onderzoeken.

Hoofdstuk 2: Evolutionaire Algoritmen (EAn) waren een heel populaire optimalisatiemethode voor een grote verscheidenheid aan toepassingen. Maar, ondanks de vele voordelen, is hun rekenintensiteit nog steeds een belemmering voor bepaalde praktische toepassingen waarbij sprake is van een rekenintensieve geschiktheidsfunctie. In dit hoofdstuk nemen we als uitgangspunt de constatering dat survival-of-the-fittest in de natuur niet gaat over de precieze mate van geschiktheid; meer van belang is een correcte volgorde in geschiktheid van de concurrerende kandidaten. Door aldus deze natuurlijke tolerantie voor onnauwkeurigheid te benutten, stellen we een vage korrel-gebaseerde aanpak voor om het aantal noodzakelijke functie-aanroepen te reduceren (een korrel is een groep van vergelijkbare exemplaren uit de populatie). Deze aanpak is gebaseerd op een adaptieve, vage geschiktheidsbenadering, die als belangrijkste doel heeft om een balans te vinden tussen nauwkeurigheid en nuttigheid van het rekenwerk. Het adaptieve algoritme past de grootte van de invloedssfeer van korrels aan, naargelang de rekenintensiteit en het bereikte niveau van convergentie. Experimentele resultaten laten zien dat de voorgestelde aanpak de convergentie naar optimale oplossingen versnelt, in vergelijking met andere, bekendere methoden. Dit wijst op toepasbaarheid op andere complexe praktijkproblemen. De voorgestelde oplossing heeft niet de nadelen van bestaande oplossingen voor geschikheidsbenadering, zoals de noodzaak van rekenintensief, on-line trainen.

Hoofdstuk 3: De oplossing , voorgesteld in hoofdstuk 2, heeft een aantal afstemmingsparameters die probleem-specifiek zijn. In de praktijk zijn een aantal rondes nodig om deze parameters te bepalen. Dit hoofdstuk stelt een vage supervisor voor als automatische af-

stemmingsstrategie, teneinde het expliciete instellen van deze parameters te vermijden. De effectiviteit ervan is onderzocht met drie traditionele optimalisatie-benchmarks en 4 verschillende keuzen voor de dimensie van de zoekruimte. Ook is het effect van het aantal korrels op het convergentietempo onderzocht. De voorgestelde methode is vervolgens toegepast op het probleem van de detectie van verborgen informatie in de vorm van een pseudoruisreeks met chipperioden van 63, 127 en 255 bits. In vergelijking met de standaardtoepassing van EA laat de experimentele analyse zien dat de voorgestelde aanpak de mogelijkheid biedt om de rekencomplexiteit van het detectieprobleem aanzienlijk te verminderen, zonder het prestatieniveau aan te tasten. Het automatisch afstemmen door de vage supervisor heft de noodzaak op van het bepalen van exacte parameterwaarden.

Hoofdstuk 4: In hoofdstuk 2 hebbeen we een oplossing gentroduceerd om de convergentie te versnellen naar de optimale oplossing voor de situatie met n doelfunctie. In praktijkproblemen is echter vaak sprake van 2 of meer doelfuncties. In dit hoofdstuk breiden we de oplossing uit hoofdstuk 2 uit naar het geval met meer dan n doelfunctie. De voorgestelde aanpak wordt vergeleken met de standaard NSGA-II, met gebruik van *Set Coverage*, *Hypervolume* en *Generational Distance* om prestaties te meten. Onze resultaten laten zien dat onze aanpak een veelbelovend alternatief is voor het omgaan met optimalisatie voor meerdere doelen wanneer het berekenen van de geschiktheidsfunctie rekenintensief is.

Hoofdstuk 5: Sommige evolutionaire rekentechnieken hebben voordelen boven andere voor aspecten zoals het gemak van implementeren, het behouden van diversiteit van de populatie, efficintie, etc. Om efficintie te bevorderen, kunnen ze vereenvoudigd en/of gekruisd worden. Er is ook een gestage toename in het aantal globale optimalisatie-algoritmen, waarbij elke algoritme zijn eigen unieke populatiedynamiek heeft. Verschillen in de populatiedynamiek karakteriseren de wijze waarop verkenning (= diversificeren van de populatie) en benutting (= benutten wat al gevonden is) uitgevoerd worden en hoe een balans wordt gevonden tussen conflicterende doelen. Deze algoritmen zijn ontwikkeld om tegemoet te komen aan de behoefte aan snellere optimalisatie. Alhoewel bestaande metaheuristieken geschikt zijn voor complexe optimalisatieproblemen, verslechterd hun convergentiegedrag bij toenemende complexiteit. Metaheuristieken passen een zoekstrategie toe die een balans zoekt tussen verkennen en benutten in een algoritme-specifieke manier. Men kan vaak constateren dat metaheuristieke algoritmen lokale minima vinden, soms zelfs van lage kwaliteit, wat betekent dat de gevonden balans niet geschikt is voor het op te lossen probleem. Bijvoorbeeld kan een algoritme, door zijn zoek-afwijking, te veel nadruk leggen op het benutten van gevonden oplossingen, waarbij weinig aandacht overblijft voor het verder exploreren van de zoekruimte als geheel. Wegens deze observatie, en genspireerd door de theorie van de Grote Stuiterpartij (”Big Bounce Theory”: een oscillatorisch model voor het heelal), hebben we een Gesimuleerd Stuiteren (Simulated Big Bounce: SBB) algoritme ontwikkeld dat, naast benutting ook robuuste verkenning toepast, teneinde uit lokale minima te kunnen ontsnappen. Dit hoofdstuk beschrijft het ontwerp van dit algoritme en toont de resultaten van een serie vergelijkende experimenten waarin de prestaties van SBB op een verzameling wiskundige benchmarks met hoge dimensie worden vergeleken met die van vijf andere populaire heuristieken. De verkregen resultaten duiden erop dat het voorgestelde algoritme (*i*) vergelijkbaar is met (en in sommige gevallen beter is dan) andere populatie-gebaseerde optimalisatie-algoritmen, en (*ii*) het aantal berekeningen van de geschiktheidsfunctie dat nodig is om even goede oplossingen te vinden, aanzienlijk vermindert . Alhoewel SBB een aantal kenmerken gemeenschappelijk heeft met bestaande optimalisatiemethoden, zo-

als deeltjeszwermoptimalisatie (Particle Swarm Optimization: PSO), bezit het daarnaast ook unieke kenmerken. Deze komen voort uit diversiteit in kinetische energie van de deeltjes, en maken het mogelijk dat het algoritme kan ontsnappen uit lokale minima. Bovendien bieden de experimentele resultaten aanwijzingen dat het kenmerk van robuuste verkenning, wat SBB karakteriseert, de grond vormt voor de gemeten superieure prestaties.

Hoofdstuk 6: Bij het vergelijken van verschillende optimalisatiestrategien, hebben we geconstateerd dat in de praktijk metaheuristieken te lijden hebben onder diverse vormen van zoek-afwijkingen. Een goed begrip hiervan is direct relevant voor het selecteren van de meest geschikte strategie. In dit hoofdstuk worden twee metrieken gentroduceerd: n voor de middelpunt-zoekende afwijking (Center Seeking Bias: CSB) en n voor de zoekafwijking door het gekozen initialisatiegebied (Initialization Region Bias: IRB). De eerste is gebaseerd op ξ-*center offset* (verschuiving van het centrum), een alternatief voor *center offset*, want deze laatste is niet geschikt voor de analyse van de middelpunt-zoekende afwijking, zoals wordt aangetoond. De metriek voor IRB wordt voorgesteld op grond van *regio-schaling (region scaling)*. De gentroduceerde metrieken worden gebruikt voor het evalueren van de de afwijkingen van drie algoritmen die losgelaten worden op een testverzameling van optimalisatieproblemen die hun optimale oplossing hebben in, of dicht bij, het centrum van de zoekruimte. Het meest opvallende resultaat in dit hoofdstuk is een aanzienlijke CSB en IRB in het zwaartekrachts-zoekalgoritme (GSA: Gravitational Search Algorithm). Dit hoofdstuk doet bovendien een voorstel voor de gedeeltelijke oplossing van CSB en IRB in GSA door de introductie van een *gespreide massa* -versie van GSA: mdGSA (md staat voor: mass dispersed). Dit bevordert de globale zoekmogelijkheden van GSA. De prestaties hiervan zijn geverifieerd met dezelfde testverzameling en met een parameter-identificatieprobleem voor een genen-regulerend netwerk. De resultaten van deze experimenten tonen de mogelijkheden van mdGSA voor het het oplossen van realistische optimalisatieproblemen uit de praktijk.