Development of optimization methods to deal with current challenges in engineering design optimization

Hemant Kumar Singh

A thesis submitted in fulfilment
of the requirements for the degree of
Doctor of Philosophy

School of Engineering and Information Technology
University College
University of New South Wales
Australian Defence Force Academy

28 February 2011

Copyright Statement

Authenticity Statement

Originality Statement

I hereby declare that this submission is my own work and to the best of my knowledge it contains no material previously published or written by another person, or substantial portions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institute, except where due acknowledgment is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

Signed .....

Date .....

Abstract

In engineering design, optimization is customary, and often indispensable. Typical cases include minimization of drag for vehicles, minimization of weight for structures like buildings and bridges, maximization of power and lift for aircraft and rockets, minimization of fuel consumption for engines, etc. Therefore it comes as no surprise that development of fast and efficient optimization algorithms for engineering design is an actively pursued research area.

In recent decades, metaheuristic algorithms have proven to be efficient, robust and versatile methods for numerical optimization. However, they usually need to evaluate a large number of candidate designs to find the optimum. This becomes prohibitive for engineering optimization problems in which each design evaluation may require computationally expensive analysis and, consequently, the optimization process may take much longer time than affordable.

Considering that the number of design evaluations is a critical factor in overall optimization time, it is imperative to develop techniques to search for the optimum design using fewest evaluations possible. With this singular goal, this thesis investigates a range of domains in which existing approaches can be improved.

Engineering problems are often highly non-linear, discontinuous, and non-differentiable, which rules out (or restricts) the applicability of analytical techniques for solving them. However, they exhibit additional attributes that prove challenging even to the existing metaheuristic techniques, thus making the search difficult and, consequently, creating a necessity for carrying out large numbers of evaluations. These include: (a) *Constraints* – constraints render a fraction (possibly large fraction) of the search space infeasible, making it hard to find the optimum and at times even a feasible design; (b) *Large number of objectives* – Pareto-dominance sorting, a commonly used technique in multi-objective

optimization algorithms, is inadequate to solve problems with large numbers of objectives, a fact well reported in literature; (c) *Large number of variables* – the search space grows exponentially with the number of variables, which results in a corresponding increase in computational effort; and (d) *Multiple models* – For certain problems, there may be multiple candidate models to choose a solution from, with none of them being an obviously preferred one. In such a case, one may need to explore each one of them to find the global best.

In this thesis, studies are conducted on each of these domains individually. Shortcomings of the existing methods are analyzed, and novel techniques are developed for efficiently handling constraints, large number of objectives/variables and multiple models. For effective constraint handling, conventional evolutionary algorithm is enhanced using a novel infeasibility driven ranking technique, while conventional simulated annealing algorithm is enhanced using an approximate descent direction coupled with dominance-based acceptance criteria. For many-objective problems, improved secondary-ranking methods and a novel dimensionality reduction technique based on *Pareto corner* search are proposed. For many-variable problems, conventional cooperative coevolutionary algorithm is enhanced using a correlation based partitioning strategy which enables it to deliver competitive performances across a variety of *separable* and *non-separable* problems. Lastly, for trans-dimensional problems, a simulated annealing based algorithm which searches through *model space* and variable space simultaneously is presented.

The proposed methods are able to achieve competitive results using markedly fewer numbers of design evaluations compared to conventional optimizers, which is demonstrated through rigorous numerical experiments on benchmark problems. Finally, the proposed algorithms are applied to a number of engineering design problems in order to accentuate their functionality and viability for solving real-life problems.

Acknowledgments

This thesis would not have been possible without blessings, encouragement and support of a number of people.

Foremost, I would like to thank my supervisor, Dr Tapabrata Ray, whose passionate guidance helped me perform to the best of my abilities. His suggestions, both in academic and non-academic matters, have served as life lessons and helped me grow as a researcher and an individual. I also thank him for funding countless lunches at *Bharat*, and his family, Jayati and Pritika for their hospitality, especially during my initial days in Australia.

I would like to thank my co-supervisor Dr Warren Smith for his guidance. I also thank him and Ms Denise Russell for their meticulous proof-reading of my thesis, which resulted in a significant improvement over the initial draft.

I sincerely acknowledge the PhD scholarship from DSARC and Completion scholarship from RRTO, UNSW@ADFA to support my work.

It is not possible to describe in words the role my friends have played in the unforgettable time I have spent in UNSW@ADFA and in Australia in general. They made the difficult times worthwhile and if some of them are not mentioned here, I can only attribute it to my aging memory. Thanks to Kamal, Ovi, Mizan, Lax, Vishwas, Abhi, Ram, Mahendra, Arif, Adnan, Najia, Vinod, Rajib, Jeyakanth, Pira, Vishal, Vinh, Andrew, Dalim, Baki, Anup, Zaman, Sarah, George, Kerry, Maruth, Chandrama, Mike, Sheila, Deepak, Ashraf, Varun and Rishabh for the cherished times we spent together in the campus, soccer and cricket sessions, barbecues, trips and places around town. Thanks to Khin for her sumptuous treats and for being a person who can always be counted on. Thanks to Priyanka, Amit and Tejas for their daily doses of sarcasm, geekism and 'Americanism', respectively. Thanks to the gang of Chinese students - Xiaoshan, Fang, Zhifang, Chang, Jingfen, Xiaodan, Xiaoran, Wanrong, Fangfei, Zhipeng,

Guofeng, Yurong, Jin and Zhaosu - with whom I shared remarkable times, despite my meager and eventually unsuccessful attempts at learning Chinese. Thanks to my older friends, Vivek, Manish, Preeti, Sanjay, Ankur, Girish, Anuj, Sid, Rahul and Niraj, without whom life would be hard to imagine.

I will always hold dear the days (and nights) spent in my office. While the view from my window had a significant role to play in that, it would be a felony to undermine the contribution of the fellow inhabitant PhD students. Thanks to Sud for late night discussions about life, universe and everything (read optimization, hypersonics and cricket). Also for the late night cricket itself. Thanks to Amitay for the brainstorming sessions, technical support, and for converting me into a Linux user. Thanks to Ahmad for his office humor and unflagging companionship through tea breaks, organization of PRSF activities and 'fighting fires'. To the new students in the office - Sharif, Asaf, Khairul, Liu Min and Amit, thanks for showing me the respect which I will strive to live up to in the years to come.

Thanks to the *people upstairs* from the Research Office - Danica for her support and advice, her magnanimous laughter that permeates the corridors, and also for not shooting me on account of frequently wandering into her office unannounced; Joseph for his compassion and energy; Vera for diligently working towards our scholarship issues and organizing events for postgraduate students and staff; and Douglas for organizing various seminars and obstacle courses, apart from his help in solving crossword puzzles during lunch-time. Thanks to the people from the Student Administration Services - Mette for bringing color to the campus life; and Christa for making my student life seamless through assistance in enrollment, thesis and visa issues.

A special mention of Amitay for his cross-disciplinary presence in my PhD life - as a colleague, housemate and friend. A special thanks also to his wife Deanna, whose zest for life continues to inspire me. Thanks to Melroy and Daron for great times in Melbourne and New Zealand.

Above all, I would like to thank my family members, who have stood by me through thick and thin. To my brother and sister, I thank you for being my guiding lights for life. To my sister-in-law, thank you for your care. To my little niece, you make my life brighter. To my father and mother who have worked painstakingly towards helping me realize my dreams while caring little for their own comfort, I am forever indebted. To you I dedicate my work, my life.

List of Publications

Book Chapters

[1] H. K. Singh and T. Ray. Divide and conquer in coevolution: A difficult balancing act. In Ruhul Sarker and T. Ray, editors, *Agent Based Evolutionary Search*, Adaptation, Learning, and Optimization, pages 117–138. Springer-Verlag Berlin Heidelberg, 2010.

[2] T. Ray, H. K. Singh, A. Isaacs, and W. Smith. Infeasibility Driven Evolutionary Algorithm for constrained optimization. In Efrén Mezura-Montes, editor, *Constraint Handling in Evolutionary Optimization*, Studies in Computational Intelligence, pages 147–167. Springer-Verlag Berlin Heidelberg, 2009.

Journal Papers

[1] H. K. Singh, A. Isaacs, and T. Ray. A Pareto Corner Search Evolutionary Algorithm and dimensionality reduction in many-objective optimization problems. *IEEE Transactions on Evolutionary Computation*. Accepted.

[2] H. K. Singh and T. Ray. C-PSA: Constrained Pareto Simulated Annealing for Constrained Multi-objective Optimization. *Information Sciences*, 180(13):2499–2513, 2010.

[3] H. K. Singh, A. Isaacs, T. Ray, and W. Smith. Trans-dimensional optimization problems: Preliminary studies using a simulated annealing based algorithm. *Journal of Hybrid Computing Research*. Accepted.

[4] H. K. Singh, T. Ray, and R. Sarker. Optimum oil production using Infeasibility Driven Evolutionary Algorithm (IDEA). *Evolutionary Computation*. Under review.

[5] K. Alam, H. K. Singh, T. Ray, and S. Anavatti. An optimization framework for the design of autonomous underwater vehicles. *Ocean Engineering*. Under review.

Conference Papers

[1] H. K. Singh and T. Ray. Performance of a hybrid EA-DE-memetic algorithm on CEC 2011 real world optimization problems. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, New Orleans, USA, 2011. IEEE Press.

[2] H.K. Singh, T. Ray, and W. Smith. Performance of Infeasibility Empowered Memetic Algorithm(IEMA) on engineering design problems. In *Proceedings of 23rd Australasian Joint Conference on Artificial Intelligence (AI)*, volume 6464 of *Lecture Notes in Computer Science*, pages 425–434, Adelaide, Australia, 2010. Springer.

[3] H. K. Singh, T. Ray, and W. Smith. Surrogate Assisted Simulated Annealing (SASA) for constrained multi-objective optimization. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, pages 4202–4209, Barcelona, Spain, 2010. IEEE Press.

[4] H. K. Singh, T. Ray, and W. Smith. Performance of Infeasibility Empowered Memetic Algorithm for CEC 2010 constrained optimization problems. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, pages 3770–3777, Barcelona, Spain, 2010. IEEE Press.

[5] H. K. Singh, A. Isaacs, T. Ray, and W. Smith. An improved secondary ranking for many objective optimization problems. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO)*, pages 1837–1838, Montreal, Canada, 2009. ACM press.

[6] H. K. Singh, A. Isaacs, T. T. Nguyen, T. Ray, and X. Yao. Performance of Infeasibility Driven Evolutionary Algorithm (IDEA) on constrained dynamic single objective optimization problems. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, pages 3127–3134, Trondheim, Norway, 2009. IEEE Press.

[7] H. K. Singh, A. Isaacs, T. Ray, and W. Smith. A study on the performance of substitute distance based approaches for evolutionary many-objective optimization. In *Proceedings of 7th International Conference on Simulated Evolution and Learning (SEAL)*, volume 5361 of *Lectures Notes in Computer Science*, pages 401–410, Melbourne, Australia, 2008. Springer.

[8] H. K. Singh, A. Isaacs, T. Ray, and W. Smith. Infeasibility Driven Evolutionary Algorithm (IDEA) for engineering design optimization. In *Proceedings of 21st Australasian Joint Conference on Artificial Intelligence (AI)*, volume 5360 of *Lecture Notes in Artificial Intelligence*, pages 104–115, Auckland, New Zealand, 2008. Springer.

[9] H. K. Singh, A. Isaacs, T. Ray, and W. Smith. A simulated annealing algorithm for single objective trans-dimensional optimization problems. In *Proceedings of 8th International Conference on Hybrid Intelligent Systems (HIS)*, pages 19–24, Barcelona, Spain, 2008. IEEE Press.

[10] H. K. Singh, A. Isaacs, T. Ray, and W. Smith. A simulated annealing algorithm for constrained multi-objective optimization problems. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, pages 1655–1662, Hong Kong, 2008. IEEE Press.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Background

Engineers endeavor to create the best possible designs. Whether to achieve goals more conveniently and safely, save precious time and resources, or break new grounds in technology, they aspire to take a leap from what barely works to what works best. This pursuit for best designs is known as design optimization.

Engineering design process invariably involves the optimization of desirable performance measures. Some typical examples include minimization of drag for vehicles, minimization of building resources needed for structures like buildings and bridges, maximization of power and lift for aircraft and rockets, minimization of fuel consumption for engines, and many more.

These performance measures, or "objectives", depend on a number of design variables; and finding the best design(s) involves finding the best combination of values of these design variables. Optimization algorithms provide a systematic way of finding these values. They may be required at different stages of a design: its conceptualization; design of individual components; and the multidisciplinary

optimization of the overall design. Therefore, to support design activity, there is a strong drive to develop faster and more effective optimization methods.

1.2 Motivation

Engineering optimization problems exhibit a number of attributes which make them difficult to solve. The objective functions involved are often highly non-linear, discontinuous and/or non-differentiable. In addition, for many cases, an explicit mathematical expression for calculating the objective may not exist (e.g. numerical simulations). This results in a severe restriction on the applicability of most *exact* (analytical) mathematical optimization techniques since they require mathematical properties that engineering problems usually do not possess. Moreover, many problems contain multiple objectives which most conventional optimization methods are not equipped to solve.

Owing to these limitations, metaheuristic methods (discussed in the next chapter) have recently gained popularity for solving difficult optimization problems. Instead of using the mathematical properties of functions (derivatives, hessians etc.), these methods search based on objective values alone. Therefore, they can be used to optimize *black box* functions (*i.e.*, those for which formulation is not explicitly available). Most of these methods are inspired from processes occurring in nature; for example, Evolutionary Algorithms (EAs) are inspired from biological evolution, swarm intelligence methods are inspired from collective behavior of systems such as ant colonies, birds in a flock, etc., whereas Simulated Annealing (SA) emulates the behavior of molten metal atoms during slow cooling. Irrespective of the metaheuristic technique used, the optimization process involves systematic evaluation of a number of candidate designs until the optimum (or near

optimum/satisfactory) design is found for the given mathematical formulation.

Numerical simulations are increasingly being used these days to evaluate designs; typical examples include finite element analysis (FEA) and computational fluid dynamics (CFD). Simulations are preferred because they can model and solve more complex systems commonly occurring in real-world problems compared to purely analytical techniques. However, this comes at the cost of computational overhead. As most of these simulations are computationally intensive, evaluation of each design takes a long time. This imposes a limit on the number of design evaluations able to be undertaken to find the optimum. Therefore, efficient optimization algorithms that can find good quality designs in as few evaluations as possible are necessary and useful. This is the motivation behind the work presented in this thesis.

1.3 Scope of Research

In this thesis, various areas which pose challenges to the existing optimization algorithms are identified. These include the presence of constraints, high numbers of objectives and/or variables, and multiple candidate models. Thereafter, new, improved strategies to deal effectively with these issues are proposed. Although these strategies are implemented either in EA or SA frameworks, most of them can be easily extended to other metaheuristic paradigms.

The aim of the work presented in this thesis is not to invent an optimization algorithm that can solve *all* problems (if that's possible), but to identify potential areas for improvement in the current metaheuristic optimization algorithms, and to demonstrate the proposed enhancements through rigorous testing on a number of mathematical benchmarks and real-world engineering design problems.

Modeling of specific problems is not included in the scope of the thesis. Efficient and effective exploration of given model(s) is the primary focus of the work. Dynamic problems, in which the objectives change with time, are also not studied in this thesis.

Although the emphasis of the work is on engineering design optimization, the methods developed are suitable as generic optimizers and can be applied to problems in other disciplines such as scheduling, finance and statistics.

1.4 Contributions of Thesis

The following contributions are made in this thesis:

1. The first contribution of this thesis is towards developing a better constraint handling method for EAs. A novel technique is proposed, in which infeasible solutions near constraint boundaries are explicitly preserved during the evolutionary search. This is in contrast to most conventional algorithms in which the preference for feasible solutions weeds out all infeasible solutions during the search. This constraint handling is incorporated in the Infeasibility Driven Evolutionary Algorithm (IDEA). The presence of marginally infeasible solutions in the population focuses the search near constraint boundaries where the optimum solutions usually lie, thus resulting in faster convergence. In addition, the marginally infeasible solutions obtained from the algorithm are also of interest to designers for *trade-off* studies. A further enhancement to IDEA (for single objective problems) is also studied by embedding local search in the algorithm.

2. The second contribution is also in the area of constraint handling, but with SA algorithm. Simulated annealing is a metaheuristic inspired from

the behavior of atoms during slow cooling of molten metals. However, as SA is a single point method, EAs (which are population based) are conventionally preferred for solving multi-objective optimization problems. Nonetheless, with certain enhancements, the ability of SA to escape local minima by accepting *uphill* moves can be advantageously used to solve difficult multi-objective constraint optimization problems. To this effect, a Constrained Pareto Simulated Annealing (C-PSA) is developed and studied in this thesis. Further enhancement by using surrogate modeling within the algorithm is also investigated.

3. The third contribution is in the area of optimization problems with large numbers of objectives (many-objective optimization). Two relevant directions are pursued in the context:

   (a) *Secondary ranking*: As reported in a number of studies in the literature, Pareto-dominance is an inadequate strategy for dealing with problems with high numbers of objectives (typically more than three). For handling many-objective optimization problems, two new secondary ranking methods are proposed.

   (b) *Dimensionality reduction*: For a number of many-objective problems, it is possible to find a reduced set of objectives in lieu of the original set in order to solve the problem. Traditional dimensionality reduction methods predict the dimensionality by analyzing a representative set of solutions obtained by running a conventional Multi-objective Evolutionary Algorithm (MOEA) for large numbers of generations; which is time consuming. The approach proposed in this thesis searches for a few key solutions instead of the whole Pareto front, and thus requires

far fewer evaluations than those used in the earlier studies. These key solutions are then analyzed to estimate the true dimensionality of the problem.

4. The fourth contribution is in the area of problems with large numbers of design variables. Conventional EAs exhibit slow convergence for problems with large numbers of variables, as the search space grows exponentially with the number of variables. To deal with such problems, improvements are proposed for existing Cooperative Coevolutionary Algorithms (CCEAs). CCEAs divide the variables in to a number of partitions, and the variables in each partition are evolved separately using a *subpopulation*. However, as current CCEAs lack appropriate partitioning strategies, their performance deteriorates for *non-separable* problems (problems with strong variables interactions). In this thesis, a new partitioning strategy is proposed, which partitions the variables based on correlations among them and enables CCEAs to solve a broad class of problems efficiently.

5. In addition, the thesis also explores the scarcely studied area of *trans-dimensional* optimization (TDO). Traditionally, most studies in design optimization have operated on a fixed model, in which the number of decision variables is known. However, sometimes the optimization problem at hand may have several possible candidate models. In such a case, one way to find the global optimum solution is to optimize the problem exhaustively using each model and then choose the best found solution; which is inefficient. In this thesis, a SA based algorithm is proposed for TDO problems, which searches through both, the "model space" and the variable space, thereby identifying the most appropriate model and the corresponding optimum

variable values simultaneously. Some preliminary studies are presented to exemplify the potential of this approach.

6. Finally, a number of numerical experiments are conducted on several benchmark test problems and engineering applications using the above mentioned algorithms. Comparison with previously reported studies are included in order to highlight the improvements achieved in optimization algorithms through the work presented in this thesis.

1.5 Organization of Thesis

Following this introduction, this thesis is divided into seven further chapters. While Chapter 2 lays the groundwork for the research, Chapters 3–6 present the principal technical contributions and the numerical experiments conducted in detail. Chapter 7 focuses on application of proposed algorithm to real-life engineering problems. Chapter 8 provides a summary and a few future directions of the presented work. Since the thesis explores diverse disciplines within optimization, the relevant literature is included in each chapter instead of as one large unit. Individual contents of the chapters are outlined as follows.

1. In Chapter 2, a brief introduction to optimization is given. A broad classification of various contemporary stochastic techniques used for optimization is discussed. Metaheuristic techniques, which form the skeleton of the optimization methods developed in this thesis, are particularly discussed in detail. Thereafter, various areas in which this thesis seeks to make improvements in existing approaches are highlighted. These include constraint handling, large-scale problems (many-objective and many-variable),

and the trans-dimensional problems. The detailed contributions in each of these ares are presented individually in further chapters.

2. In Chapter 3, two new algorithms for dealing with constrained optimization problems are proposed. One of them, IDEA, is an improved EA that uses infeasible solutions effectively in order to converge faster to the Pareto front. The other algorithm, C-PSA, extends the conventional SA to deal with difficult constrained multi-objective problems.

3. In Chapter 4, two approaches for handling optimization problems with large numbers of objectives are proposed: the first is a modification in the secondary ranking procedure to improve the convergence while also ensuring a good diversity among solutions; and second is a novel method for identifying the true dimensionality of the Pareto front using *Pareto corner search*.

4. In Chapter 5, a Cooperative Coevolutionary Algorithm with adaptive variable partitioning (CCEA-AVP) for dealing effectively with optimization problems containing large number of decision variables is proposed.

5. In Chapter 6, a Simulated Annealing based Algorithm for Trans-dimensional Optimization (SA-TDO) is proposed. In the context of the presented work, a trans-dimensional optimization problem refers to one in which the objective function(s) can be evaluated using more than one model with possibly a different number of design variables involved in each.

6. In Chapter 7, further enhancements to the IDEA and C-PSA algorithms are explored, which include the embedding of local search and surrogate modeling in these algorithms. Thereafter, results for a number of engineer-

ing examples are presented using various algorithms proposed in the thesis. The engineering design examples include both single- and multi-objective constrained problems.

7. In Chapter 8, a summary of the findings of the work is presented. In addition, future issues and directions which could be pursued with the aim of making the algorithms more efficient for handling various types of optimization problems, are identified.

Chapter 2

Optimization and Metaheuristics

Abstract

In this chapter, an introduction to optimization is presented. A broad classification of optimization algorithms developed to date is discussed and evaluated. Subsequently, shortcomings of the existing optimization methods, especially in the context of engineering design problems, are identified. These shortcomings have motivated the research work conducted in this thesis.

2.1 Overview

In simple terms, optimization means the minimization or maximization of a given quantity. This "quantity" to be optimized is referred to as the objective. The objective may be calculated using an explicit mathematical function, a computer simulation, a statistical estimate/metric or even a performance measure based on a human decision making process.

This simple sounding process of optimization has immense manifestations in

the real world. Many things we see around us are either optimized with respect to certain performance measure(s) or are in the process of optimization. Nature, in its own way, tries to optimize the chances of survival of various living forms by evolution through natural selection. The development of wings for flying, the ability of some animals to camouflage to escape predators, the development of opposable thumbs in humans, etc., are some biological examples of how living forms are "optimized", albeit slowly but steadily, to suit their existence on earth. In addition, physical processes such as crystallization; nuclear processes such as carbon decay and radioactivity; and chemical processes such as combustion, favor the formation of the most stable states (that have minimum energy) of isotopes, elements or compounds.

In the context of engineering design, optimization is often essential, primarily for the following reasons.

1. There is a limit on the available human resources, time, and physical resources such as space, minerals, oil, etc. Therefore, an ideal design must perform a given task at minimum possible expense of resources. In reality, a competitive design will do it better than available alternatives.

2. Humans have a perpetual drive to break new frontiers in technology and come up with new, previously inconceivable, designs. Although, in many cases, the "absolute best" is either ambiguous or hard to conceive, any sustainable improvement in existing technology can be considered to be a step towards it.

For example, if a bridge or tunnel is to be constructed to meet certain transportation requirements, it can be constructed using excessive quantities of construction material (safe but over-designed), but considering the cost and

limited availability of the resources, it will not be a good or viable design. Therefore, the design must be optimized in order to give a required standard of safety using as little resources as possible. Similarly, for an efficient industry to produce goods in large quantities, the manufacturing process has to be optimized using concepts such as the assembly line. Sustainable transport vehicles, such as cars, ships and aircraft have to be optimized for safety, power and fuel efficiency by using better engines and body profiles.

Given the applicability of optimization at the root of most design processes, there is a natural inclination among researchers to develop efficient methods of optimization, as even small improvements in optimization algorithms may amount to huge collective benefits in the real world. As a simple example, consider a new optimized body design for automobiles which reduces drag forces so that fuel efficiency increases by merely 0.1 %. Currently, there are approximately 1 billion automobiles in the world, predominantly cars running on gasoline. An average car consumes about 1000 liters of gasoline per year. Saving 0.1% of that, or 1 liter per car, translates to a collective savings of about 1 billion liters per year, which is an substantial amount.

In this thesis, an effort is made to explore the possible challenges facing the existing optimization methods. Subsequently, novel methods for dealing with these challenges effectively and countering the shortcomings of existing optimization algorithms are proposed.

In the following sections, the mathematical definition of an optimization problem is given and various optimization techniques proposed in literature are presented. Thereafter, the challenges faced by the existing techniques are discussed in order to build a foundation for the work that is presented in the following chapters.

2.2 Optimization Problem

A generic optimization problem can be posed as shown in Equation 2.1

$$\begin{aligned} \text{Minimize} \quad & f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}) \\ \text{Subject to} \quad & g_i(\mathbf{x}) \geq 0, \quad i = 1, \ldots, m \\ & h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, p \end{aligned} \tag{2.1}$$

where $x_1, \ldots, x_n$ (which form the decision vector $\mathbf{x}$) are the design variables that can be integer or real, continuous or discrete. If only one objective is to be minimized ($k = 1$), the problem is called a single-objective problem, whereas if more than one objective is to be minimized, then the problem is termed a multi-objective problem. The design variables are usually bounded by lower and upper bounds to form the design space $\mathcal{S} \subset \mathbf{R}^n$. The objective functions $f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})$ are simultaneously minimized while satisfying the inequality constraints $g_i(\mathbf{x}) \geq 0$ and equality constraints $h_j(\mathbf{x}) = 0$. In practice, the equality constraints are converted to inequality constraints using a small tolerance as $|h_j(\mathbf{x})| \leq \epsilon$. A maximization problem can be easily converted into a minimization problem by multiplying the objectives by -1. For optimization problems with constraints, the design space is divided into feasible region $\mathcal{F} \subset \mathcal{S}$ (in which all constraints are satisfied) and infeasible region (in which one or more constraints are violated).

Since maximization and minimization problems can be interchanged by merely altering the sign of the objective functions (duality principle[5]), for the sake of uniformity, (and without loss of generality), all optimization problems henceforth considered are studied as *minimization* problems, unless stated otherwise.

2.2.1 Single-objective problem

If only one objective is to be optimized, the problem is termed a single-objective optimization problem. The aim of optimization for such a problem is to find all $\mathbf{x} \subset \mathcal{S}$ such that $f(\mathbf{x})$ assumes the minimum value f^*. The solution $\mathbf{x}$ to such a problem can be either a unique solution, or there may be multiple values of $\mathbf{x}$ for which the objective value is f^*.

2.2.2 Multi-objective problem

If multiple objectives have to be optimized simultaneously, the problem is termed a multi-objective optimization problem. If the objectives are not in conflict with each other (*i.e.*, if optimizing one objective also optimizes the remaining objective(s)), the problem can be solved as a single objective problem by considering any one objective. However, if the objectives are in conflict, the optimum values of the individual objectives do not occur at the same solution(s). For such a problem, the optimum does not comprise a single solution but instead a set of solutions, which corresponds to the *best* trade-off front that can be achieved for the given objectives. To explicate this, it is essential to introduce the concept of *dominance* (also referred to as *Pareto-dominance*).

A solution $\mathbf{x_1}$ is said to *dominate* a solution $\mathbf{x_2}$ if

1. $f(\mathbf{x_1})$ is not worse than $f(\mathbf{x_2})$ for any objective, *i.e.,*

$$f_i(\mathbf{x_1}) \leq f_i(\mathbf{x_2}), \quad \text{for} \quad i = 1, 2, \ldots k;$$

and

2. $f(\mathbf{x_1})$ is better than $f(\mathbf{x_2})$ for at least one objective, *i.e.,*

$$f_i(\mathbf{x_1}) < f_(\mathbf{x_2}) \quad \text{for at least one } i \in \{1, 2, \ldots k\}$$

If $\mathbf{x_1}$ does not dominate $\mathbf{x_2}$, and vice versa, $\mathbf{x_1}$ and $\mathbf{x_2}$ are said to be *non-dominated* with respect to each other. By extension, a set of solutions in which none of the solutions dominate any other solution(s), is known as a *non-dominated set*. In multi-objective optimization, two non-dominated solutions are considered equally good in terms of convergence. In Figure 2.1, four solutions, A, B, C and D are shown in the objective space. Among these solutions, A dominates C because it is better in both objectives, *i.e.* $f_{1,A} < f_{1,C}$ and $f_{2,A} < f_{2,C}$. On the other hand, A and B are non-dominated with respect to each other, since A is better in f_2, where as B is better in f_1, *i.e.* $f_{1,A} > f_{1,B}$ and $f_{2,A} < f_{2,B}$. For the same reason, pairs B and C, A and D and C and D are non-dominated with respect to each other.

Among these four solutions, A, B and D form a non-dominated set since they are not dominated by any solution.

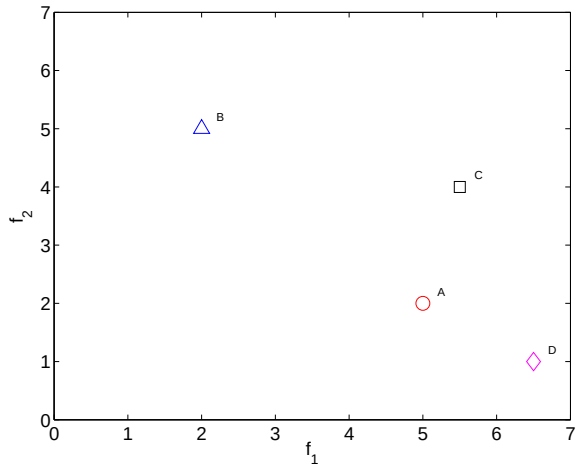


Figure 2.1: Dominance relationships for multi-objective optimization (A dominates C; A, B and D form a non-dominated set)

The aim of multi-objective optimization is to capture a non-dominated set consisting of solutions that are not dominated by any other solution in the search space. This non-dominated set is the optimum solution, known as Pareto optimal front (POF), or Pareto front. It is then up to the end user to choose design(s) from this set based on preference for different objectives.

Definition 1. *A Pareto front is a non-dominated set* $\mathcal{F}$ *, such that there does not exist a solution* $\mathbf{x} \notin \mathcal{F}$ *in the feasible search space* $\mathcal{S}$ *that dominates a solution* $\mathbf{x}^* \in \mathcal{F}$*.*

2.3 Optimization and Metaheuristics

As optimization represents an expansive area of research with numerous applications, a number of approaches to optimization have been developed in the literature. There is no unanimously accepted "best" approach, and perhaps never will be, as different approaches are tailored to perform the best in their respective

domain(s). Therefore, before applying optimization to any (class of) problem, it is important to choose the approach wisely. At the same time, optimization techniques that are widely applicable are also well sought for reducing the variability of the final results due to the choice of algorithms, eventually making the choice less critical in the overall design process.

A broad classification of single objective optimization problems and methods in the literature is shown in Figure 2.2, which is adapted from the classification that appears in Collette and Siarry [1]. At the top level, optimization problems can be of three fundamental types: combinatorial (*discrete*), continuous, or mixed. These properties refer to the nature of the design variables. In combinatorial problems, the design variables can take only certain discrete values (e.g. integers), whereas in continuous problems, these variables can take any real value withing certain bounds. This distinction is important because for combinatorial problems, *all* possible solutions can be enumerated, but for continuous problems this is not possible. In a *mixed* problem, some variables are continuous while others are discrete.

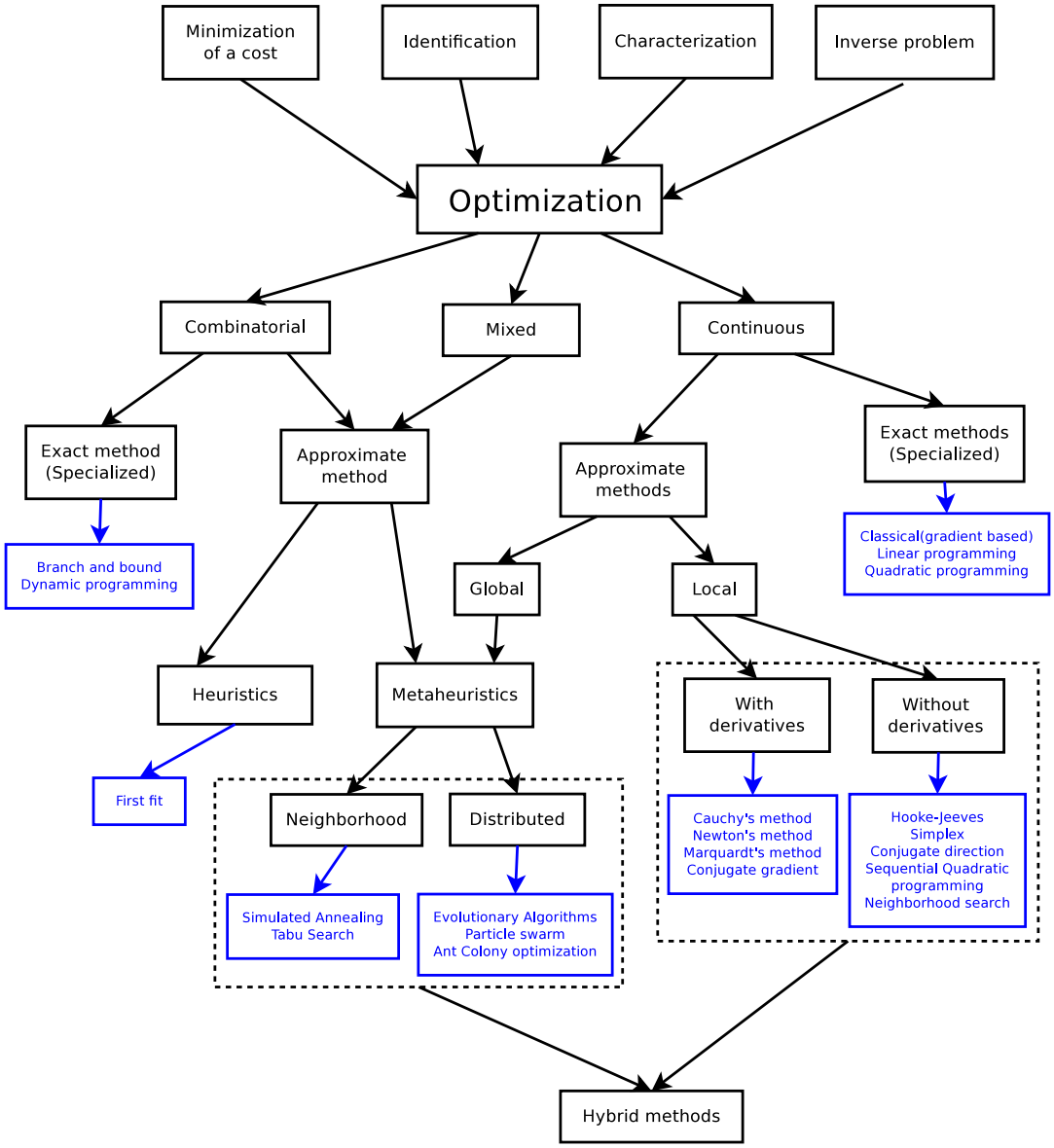


Figure 2.2: Classification of optimization algorithms (figure adapted from [1])

For solving optimization problems, there are two possible kinds of methods: *exact* and *approximate*. Exact methods are those which guarantee to achieve the precise optimum solution to the mathematical model posed. On the other hand, approximate methods seek the best possible solution near the optimum solution by searching the variable space through *intelligent guesses* and continually im-

proving the solution in the process. Although the approximate methods do not guarantee to find the optimum solution(s), they are useful for many reasons as detailed in the following discussions.

Since all possible solutions can be enumerated for combinatorial problems, a crude (but exact) technique is to enumerate all possible solutions, and then choose the one corresponding to the minimum value of the objective. However, the computational complexity of this procedure grows exponentially with the number of variables, and is not practical for most applications. Other exact techniques (such as branch and bound or dynamic programming) behave in similar way for difficult problems (more specifically, *NP-hard problems*[1]), and therefore tend to be as computationally expensive as complete enumeration.

On the other hand, complete enumeration is not possible for continuous problems. Finding exact solution for such problems requires analytically solving the equations, which is possible only for certain types of functions, e.g. linear or quadratic. In addition, classical methods often use calculations of the first- and/or second-order derivatives, which require the functions to be continuous and smooth. Unfortunately, most problems encountered in engineering design do not possess such properties. In fact, for many problems, an explicit mathematical expression for calculating the objective function(s) may not even exist (e.g., the objective value may be a result of a CFD/FEA analysis on a design with given input variables). For this reasons, most exact techniques can not be applied to engineering design problems.

Due to the limitations of exact optimization methods, a number of approximate methods have been developed. While the usage of the term *approximate* might suggest that these methods are inferior, in practice they are more versatile and advantageous for handling complex problems. Approximate methods

predominantly use objective values rather than derivatives to guide the search, although some of them also use numerically calculated gradients. For combinatorial problems, approximate methods can be further classified into *heuristic* and *metaheuristic* methods. Heuristic methods were defined by Reeves [6] in the context of discrete optimization as follows.

Definition 2. *A heuristic is a technique which seeks good (i.e., near optimal) solutions at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution is.*

Heuristics are usually designed to solve a specific problem. More generic methods, that still follow the above definition, and work for continuous, discrete and mixed problems alike, are termed *metaheuristics*.

For continuous problems, approximate methods may be global or local in nature. Local methods search for the optimum starting from a given solution, and usually converge to the nearest local optimum solution. Therefore, for multimodal problems (*i.e.*, those with multiple local/global minima), the solution achieved using local search depends on the starting solution. On the other hand, global methods search for the global optimum solution. Metaheuristics are global search methods which operate using one of the following approaches. First is the *neighborhood metaheuristics*, where a single solution is evolved by searching its neighborhood until no further improvements are possible in the objective value(s). Second is the *distributed metaheuristics*, where a *population* of solutions are evolved simultaneously. Information exchange amongst the individuals in the population helps in effective explorations of the search space in order to find the global optimum.

Research on metaheuristic techniques has proliferated in recent years, the reasons for which can be inferred from the classification given in Figure 2.2, as detailed below.

1. Their biggest advantage is their versatility, as they can deal with combinatorial, continuous, and mixed problems with no or minimal customization being required.

2. They do not require the objective functions to be continuous, smooth or to possess any specific properties such as linearity. As most real-life problems lack such properties, metaheuristic methods are suitable for dealing with them. Also, since they require only objective values to guide the search, even *black box* functions (*i.e.*, in which relationship between the objective function and the design variables is not explicitly known) can be optimized.

3. They seek global optimum solution, without relying significantly on the quality of the starting solution. Additionally, the search need not even start from a feasible solution, as a number of constraint handling techniques can be integrated into the algorithm so that it can find feasible solutions during the search.

4. They can be combined with local search methods to form *hybrid* search algorithms. Hybrid algorithms combine the advantages of effective global search with rapid local search to get good quality solutions faster.

5. Distributed (population based) metaheuristic algorithms can provide the Pareto front for multi-objective problems in a single run, whereas most of the single-point methods can provide only one Pareto optimal solution in one run.

6. Population based algorithms can be implemented on parallel processors, which can significantly reduce the overall time taken for optimization.

7. Metaheuristics are often inspired from physical / biological processes occurring in nature. For example, evolutionary algorithm (EA) tries to simulate the process of evolution of biological species in nature through natural selection. Simulated annealing (SA) emulates the behavior of atoms during slow cooling of metals to form crystals.

   The consequences of these processes in nature (*e.g.*, well adapted species, minimum energy crystals etc.) provide a proof in principle of how the metaheuristics can be used to improve designs (objectives). Therefore, though conclusive proof of optimality can not be guaranteed, these methods can be applied with confidence to a number of real-life optimization problems.

Due to the advantages stated above, metaheuristic methods are used as a tool for solving optimization problems in many studies, including this thesis. However, in spite of their advantages, metaheuristic methods currently cannot be used for a number of potential applications for the following reason.

Metaheuristic methods, being stochastic in nature, explore a number of solutions during a search. Not all of these solutions lead to a better objective value and such solutions are discarded. For example, in an EA, a new population (of solutions) is created every generation and only those solutions which are better than the existing ones are passed on to the next generation. Often, it may take the algorithm a large number of generations to converge to the optimum. This results in a correspondingly large number of function evaluations, which may be prohibitive. For many engineering design problems in particular, each function evaluation may be a result of a computationally intensive simulation such as

CFD, FEA etc. Therefore, a very limited number of function evaluations are affordable in order to come up with the best possible solution.

This defines the direction of the research carried out in this thesis. The basic question this study tries to answer is: *how can the existing algorithms be enhanced so that they are able to find near optimum solutions in as few function evaluations as possible ?* To this effect, firstly, the major roadblocks for the existing algorithms are identified, and thereafter, mechanisms for countering them are proposed and documented in detail.

Before delving into the major challenges facing existing optimization algorithms, the two most popular metaheuristic techniques are briefly described. One of them, EA, is a distributed approach, where as the second, SA, is a neighborhood approach. Both these methods are explored in this thesis, and the proposed improvements to them can be implemented in other algorithms of similar nature.

2.3.1 Evolutionary Algorithms

Evolutionary Algorithms (EAs) are inspired by the evolution of biological species in nature through Charles Darwin's theory of *natural selection*[7] or *survival of the fittest*. A detailed account on biologically inspired computing can be found in [8]. In nature, the process of evolution operates by recombining genes from the parents to produce offspring. Since all the individuals have to compete for limited resources (food, water, sunlight etc.), only the fittest members survive. This process repeats for generations, making these species better suited for their survival. This evolutionary mechanism is simulated in EAs to create *fitter* solutions (solutions with superior objective values). EAs evolve a *population* of solutions iteratively. New solutions are generated by applying

mathematical operators on the existing population (parents). These solutions are then evaluated for fitness. Out of the parents and offspring solution, those with good objective values are passed on to the next generation while the rest are rejected. This process continues until all solutions are converged near the optimum. A typical framework of an EA is shown in Algorithm 2.1.

Algorithm 2.1 Evolutionary Algorithm framework

Require: $N_G > 1$ {Number of Generations}
1: Initialize(P_1)
2: Evaluate(P_1)
3: Rank(P_1)
4: **for** $i = 2$ to N_G **do**
5: C_i = Evolve(P_{i-1})
6: Evaluate(C_i)
7: Rank($P_{i-1} + C_i$)
8: P_i = Reduce($P_{i-1} + C_i$)
9: **end for**

A typical EA comprises following key steps.

Initialization

The search starts with initialization of the population by sampling solutions from the design space. Random sampling is commonly adopted, in which each design variable is sampled with uniform probability between its lower and upper bounds.

Evaluation

For each solution in the population, the objective and constraint functions are evaluated using appropriate simulation, analysis or mathematical formulation. These objective and constraint values are used during ranking to assign fitness to each solution.

Evolution

In an EA, an offspring population is evolved from the current population using crossover and mutation operations. In crossover, two new solutions are created from two parent solutions using a crossover operator. In mutation, one or more variables of a solution are perturbed using a mutation operator. To select a parent, two solutions are picked randomly from the current population and the solution with better fitness is considered for creating offspring. This comparison between two solutions is referred to as binary tournament.

Ranking

Individual solutions in a population are ranked based on their fitness value. In most algorithms, feasible solutions are ranked higher than infeasible solutions.

For single-objective optimization, feasible solutions are sorted based on the objective value as the fitness. For multi-objective optimization, solutions are compared using Pareto-dominance relationships.

For infeasible solutions the fitness corresponds to the maximum or the sum of constraint violations. Infeasible solutions are sorted in the increasing order of constraint violation (maximum or sum of violations).

Reduction

The reduction process is used to retain N best solutions (where N is the population size) from a set of $2N$ solutions (parent and offspring populations) for the next generation. It uses the fitness values or ranks obtained in the ranking procedure.

Currently, Non-dominated Sorting Algorithm II (NSGA-II)[9] is arguably the best performing and most widely used generic EA for optimization. Therefore,

NSGA-II is used as a benchmark algorithm in this thesis. The comparisons of the newly proposed algorithms are done with NSGA-II in order to gauge their efficacy. As the name suggests, NSGA-II uses non-dominated sorting as the primary ranking method. For single-objective problems, non-dominated sorting is equivalent to sorting the solutions based on the objective value itself, in which case the algorithm reduces to a basic Real-coded Genetic Algorithm (RGA). To evolve the solutions, NSGA-II uses Simulated Binary Crossover (SBX) [10] and polynomial mutation [11]. For maintaining diversity, it uses crowding distance sorting. The infeasible solutions are sorted based on the sum of constraint violations, and ranked lower than the feasible solutions.

2.3.2 Simulated Annealing

Simulated Annealing (SA) [12] is an optimization algorithm which emulates the behavior of hot metal atoms subjected to slow cooling. Initially, when the atoms are at a high temperature (high energy state), they are relatively free to move in the molten metal. As the temperature is lowered, the atoms try to arrange themselves so as to attain the lowest possible energy state. During this process, the atoms also occasionally make transitions to higher energy states, the probability of which reduces as the temperature decreases. Freedom of movement of the atoms is gradually restricted as the temperature is lowered, until finally the atoms rearrange themselves to form solid crystals having minimum energy state (provided the annealing is slow enough).

This concept is used for numerical optimization by incorporating similar kind of properties into the search. To minimize a given objective (energy), the search is started with a random solution. A trial solution is then generated in the neighborhood of the current solution. Whether the search moves to the proposed

trial solution depends on the *acceptance ratio*, which in turn is affected by a parameter T which is analogous to the temperature for the physical annealing process. If the current solution is $\mathbf{x_1}$ with a function value of f_1, and the trial solution $\mathbf{x_2}$ has a function value of f_2, the trial solution is always accepted if $f_2 <= f_1$ (thereby improving the function value, assuming a *minimization* problem). However, if $f_2 > f_1$, the search moves to the new solution with an acceptance ratio (or acceptance probability) AR which is calculated as

$$AR = e^{-(f_2 - f_1)/T},$$

where T is the current temperature. Note that this conventional formulation is for single-objective unconstrained problems only. As is clear from the above equation, for high values of T, AR will be high, thereby allowing the search to accept even worse solutions, and consequently performing a *random walk* through the search space. However, at low temperatures, this transition is very unlikely. The search is started with a high value of T which is lowered gradually over iterations to a small value. Hence, emphasis in the early stages of the search is on global search, whereas the later stages focus on local search to fine-tune the variables near the optimum. Usually, at a given temperature T, $M(> 1)$ trials are conducted, where M is called the *epoch length*. The temperature is usually reduced exponentially using a cooling ratio α. However, other annealing schemes such as linear decrement also appear in the literature. SA is outlined in Algorithm 2.2. The built-in finite probability to accept worse or "uphill" solutions during the search allows SA to escape local minima, thereby making it a robust optimization algorithm. Under certain conditions, a proof of convergence to the global optimum for SA has been shown by Geman and Geman [13].

Algorithm 2.2 Simulated Annealing algorithm

Require: T_{max}, T_{min}, $M > 1$, $\alpha < 1$

Calculate Number of Iterations N
Set $T = T_{max}$, $\mathbf{x_{old}}$ = random solution
for $i = 1$ to N **do**
  for $j = 1$ to M **do**
    $\mathbf{x_{new}} = \text{perturb}(\mathbf{x_{old}})$
    if $f_{\text{new}} \leq f_{\text{old}}$ **then**
      Set $\mathbf{x}_{new} = \mathbf{x}_{old}$
    else
      $AR = exp(-(f_{\text{new}} - f_{\text{old}})/T)$
      Set $\mathbf{x_{new}} = \mathbf{x_{old}}$ with probability AR
    end if
  end for
  $T = T \times \alpha$
end for

2.4 Performance Measurement

2.4.1 Single objective optimization

For a single objective optimization problem, performance comparison is relatively straightforward as the objective value alone provides an unambiguous measure of the quality of the solution. In addition, since stochastic algorithms (such as those used in this study) can converge to a different solution every time they are run, their reliability in performance has to be evaluated by conducting multiple runs. In this thesis, multiple independent runs are carried out and the statistics of the objective values for these runs are used to compare the performance of various algorithms.

2.4.2 Multi-objective optimization

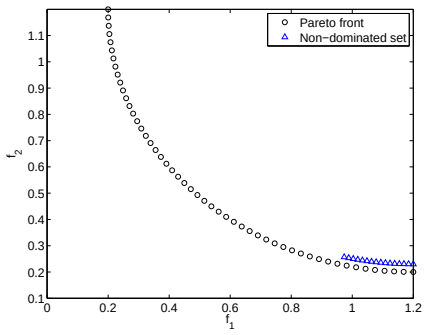
For a multi-objective optimization problem, the solution does not comprise a unique minimum objective value, but instead a set of trade-off solutions. There-

fore, the objective values alone cannot be used for direct comparison of the performances of two or more algorithms.

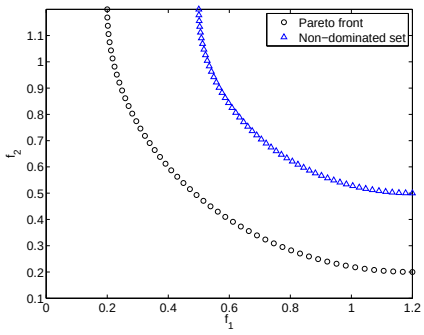
Multi-objective optimization algorithms aim to achieve a non-dominated set with the following two characteristics:

- **Convergence**: the solutions in the non-dominated set should be close to the solutions on the Pareto optimal front.
- **Diversity**: the solutions should be uniformly spread to span the Pareto front.

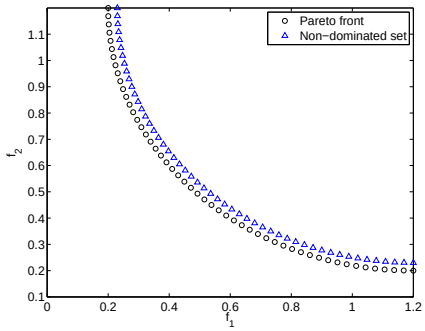
Illustrated in Figure 2.3 is a solution with good convergence but poor diversity, a second with poor convergence but good diversity, and a third with both good convergence and good diversity.



(a) Good convergence, poor diversity



(b) Poor convergence, good diversity



(c) Good convergence, good diversity

Figure 2.3: Non-dominated solutions obtained from multi-objective optimization

A number of performance metrics have been formulated in the literature to evaluate the quality of non-dominated solutions obtained by an algorithm. Of these, the two most common performance metrics are used in this study, namely *displacement* and *hypervolume*.

Displacement

The displacement metric [14, 15, 16] evaluates the quality of the solution set by measuring the distance of the set from the Pareto front in objective space. Mathematically, the displacement metric is calculated as shown in Equation 2.2.

$$\text{Displacement} = \frac{1}{|P|} \times \sum_{i=1}^{|P|} \left(min_{j=1}^{|Q|} d(i,j) \right), \tag{2.2}$$

where P is the known Pareto front, Q is the non-dominated solution set and $d(i,j)$ is the Euclidean distance between the i^{th} solution of set P and the j^{th} solution of set Q (see figure 2.4).

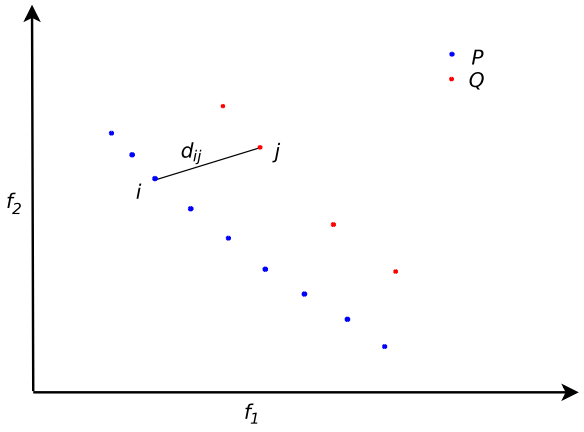


Figure 2.4: Calculation of displacement ($P \equiv$ Pareto front, $Q \equiv$ non-dominated set)

A low displacement value indicates that the solutions in Q are close to those in P, implying a good convergence. At the same time, diversity is also implicitly captured in the metric. This is because if the solutions in the set Q lie in a localized region of the front P (e.g. Figure 2.3(a)), then distance of the remote regions of the Pareto front from the solutions in set Q will be high, thereby resulting in an inferior value of the displacement metric.

Hypervolume

The hypervolume metric estimates the quality of a solution set Q by measuring the *dominated volume*. Hypervolume for a set Q is defined as the volume occupied by the set in objective space with respect to a reference point R. Mathematically,

it is calculated as shown in Equation 2.3.

$$\text{Hypervolume} = \left\{ \bigcup_i a_i | v_i \epsilon Q \right\}, \tag{2.3}$$

where a_i is the volume (area in the case of two objectives) occupied by the solution v_i (Figure 2.5) with respect to a reference point R.

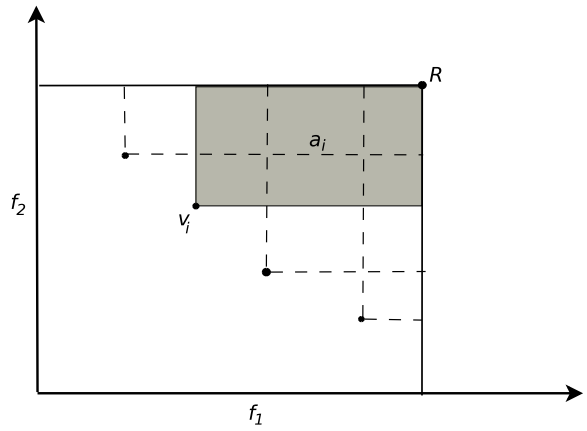


Figure 2.5: Calculation of hypervolume (both objectives are being minimized)

Like displacement, hypervolume also captures both the convergence and diversity of a set. The more converged the solutions are to the optimum, and the larger their diversity is, the greater is the area dominated by them. For example, the total area dominated by the non-dominated sets shown in Figures 2.3(a) and (b) will be much less than that by the non-dominated set shown in Figure 2.3(c). A higher value of hypervolume indicates a better quality of the non-dominated set under consideration. Further discussion on the hypervolume metric can be found in [1, 17, 18].

Some additional metrics are also used occasionally in the thesis for comparison of non-dominated sets for certain problems. The definitions of these metrics are provided where necessary.

2.5 Current Challenges in Optimization

As stated earlier, the aim of the work reported in this thesis is to reduce the computational effort (more specifically, the number of evaluations) required to obtain near-optimum solutions. Evidently, the first step is to identify the factors responsible for the high computational effort for engineering design optimization. Through an extensive literature review, three major challenging factors are identified and studied in this thesis. A brief overview of them is given in the following subsections, while their detailed description and contributions to counter them are discussed in succeeding chapters.

2.5.1 Constraint handling

Engineering design optimization problems often involve a number of constraints, which may result from factors such as practicality, safety and functionality of the design and/or limit on time and resources. Mathematically, constraints have the effect of rendering some (or a large) fraction of the search area infeasible. Consequently, an optimization algorithm often has difficulty searching for the optimum through the restricted feasible space, and often difficulty in even finding a feasible solution. Therefore, an efficient constraint handling mechanism is imperative to enable an algorithm to search effectively through the variable space.

2.5.2 Large scale optimization

The large scale of a problem can adversely affect the performance of most algorithms. Large scale optimization problems can be of two kinds:

- *Many-objective problems*: most of the multi-objective optimization algorithms predominantly use non-dominated sorting (or equivalent) in order

to drive the population towards the Pareto front. However, as will be detailed later, non-dominated sorting is an inadequate strategy for creating a selection pressure towards the optimum if the problem has more than three objectives. Problems with four or more objectives are termed many-objective optimization problems.

- *Many-variable problems*: some problems encountered in engineering can have a large number of variables (sometimes even hundreds or thousands). The search space grows exponentially with the number of variables and so does the corresponding search effort. This presents a major challenge for optimization algorithms, especially the conventional metaheuristic algorithms which do not utilize any mathematical relationships between various variables during the search.

2.5.3 Trans-dimensional optimization

Most of the existing algorithms assume the number of design variables to be fixed, thereby operating on a fixed model. However, for some design problems, there may be a number of possible models to choose from, before solving for optimum values of variables for that model. Such problems are referred to as trans-dimensional optimization problems. Exhaustively exploring each model one by one using conventional optimization methods may incur a huge computational cost. Hence, techniques are needed which can explore the model space as well as the variable space simultaneously. Very few studies have been undertaken and reported in the literature to address such problems. Further, the available methods for trans-dimensional optimization have very limited applicability for engineering design problems due to the absence of mechanisms to handle con-

straints and multiple objectives.

2.6 Summary

In this chapter, an introduction to optimization and a broad classification of optimization algorithms developed to date are given. Subsequently, the advantages of metaheuristic algorithms which make them ideal candidates for solving engineering problems are highlighted. Thereafter, certain disadvantages of these methods are described, which prevents them from being applied to a number of potential areas presently. Finally, the factors that cause these shortcomings, which build the basis for the work presented in this thesis, are discussed.

Chapter 3

Constraint Handling in Optimization

Abstract

Real-life optimization problems often involve a number of constraints, and the performance of an optimization algorithm depends largely on how effectively it can handle them. In this chapter, two new algorithms to deal with constrained optimization problems are proposed. One of them is implemented in an Evolutionary Algorithm (EA) paradigm, the other in a Simulated Annealing (SA) paradigm. However, the proposed ideas can also be extended to other similar frameworks, such as Differential Evolution (DE), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), etc. Comparison with the published state-of-the-art algorithms on various benchmark problems are included in order to highlight the contributions of the work.

3.1 Introduction

In real life, one often encounters problems in which one or more objectives have to be optimized simultaneously, subject to a set of constraints. In recent years, population-based metaheuristic optimization algorithms have been largely preferred for solving optimization problems, particularly MO problems, for the reasons detailed in previous chapter. The performance of these methods for constrained optimization problems is known to be largely dependent on the mechanism used for handling constraints. A detailed review of various constraint handling techniques used with EAs is presented in [19, 20, 21]. Some of the most commonly used constraint-handling techniques are listed below.

- **Penalty function-based methods:** Penalty function methods are one of the most commonly adopted forms of constraint handling. In this approach, the fitness of infeasible solutions is degraded using a weighted sum of constraint violations. Variants of the penalty function based approach include static penalty [22, 23], dynamic penalty [24], annealing penalty [25, 26], adaptive penalty [27, 28] and death penalty [29]. Implementations of most of these schemes often require additional parameters. The choice of these parameters is often not trivial, and the result of the optimization process is known to be highly sensitive to these parameters.

- **Dominance-based approaches:** "A dominance based" constraint handling technique implies that while performing a Pareto-dominance ranking of solutions, the constraints (or a quantity calculated based on them) are also considered as objectives. Ray *et al.* [30] developed an EA based on the non-dominance of solutions in both the objective and the constraint space. Ho and Shimizu [31] converted the objective function value and the

constraint violation into numerical values with the same order of magnitude. Concepts of dominance have also been used in a recent simulated annealing based optimization algorithm developed by Hedar and Fukushima [2]. Often, a single-objective constraint problem is solved by converting it into a multi-objective problem by adding constraints as objectives. A comparison of performance of various multi-objective evolutionary algorithms (MOEAs) on constrained optimization (single-objective non-linear problems) using concepts of Pareto-dominance can be found in [32]. For multi-objective problems, Vieira *et al.* [33, 34] used constraints as additional objectives. In the above approaches, a significant amount of time may be spent on non-dominated sorting if there are a large number of constraints. There is also a risk of generating solutions with excellent objective function values but poor constraint satisfaction.

- **Maintaining infeasible solutions:** A few researchers have proposed maintaining a proportion of infeasible solutions in the population during the evolution. For single-objective optimization, Coello Coello [35] proposed splitting the population into various sub-populations, each of which use either the objective or one of the constraints as the fitness function. Hamida and Schoenauer [36] developed an Adaptive Segregational Algorithm (ASCHEA) in which the proportion of feasible solutions in the population is controlled using an adaptive penalty. This approach used a single penalty coefficient for all constraints and was later extended [37] to incorporate a separate penalty coefficient for each constraint. Hinterding and Michalewicz [38] proposed another approach (CONGA) for constraint handling using effective parent matching in which mating is done between two infeasible solutions satisfying different constraints in

order to create children which will satisfy all constraints. Mezura-Montes and Coello Coello [39] suggested a Simple Multimembered Evolutionary Strategy (SMES) in which the "best" infeasible solution determined by its objective function value is allowed to be copied into the next generation. Although these algorithms (ASCHEA, CONGA, and SMES) effectively illustrate the benefits of preserving infeasible solutions in the population, their scope was demonstrated on single-objective optimization problems only and their extension to multi-objective domain has not been discussed. In an attempt to simultaneously generate solutions to unconstrained and constrained optimization formulations of a multi-objective problem, Isaacs, Ray and Smith [40] introduced a Constraint Handling Evolutionary Algorithm (CHEA). In CHEA, some of the infeasible solutions are preserved during the search. The infeasible solutions in the population are ranked using the original objectives along with an additional objective, the number of constraint violations. The incorporation of search through the infeasible space improves the efficiency of the algorithm. However, CHEA does not have any provisions for quantifying the amount of constraint violation and the infeasible solutions obtained are not suitable for *trade-off* studies. Trade-off studies imply searching for a possibility of deriving benefits in the objective value(s) by marginal compromise on the constraints.

- **Other constraint-handling methods:** These include special representation schemes for maintaining feasibility [41, 42], repair algorithms [43, 44, 45, 46], handling constraints and objectives separately [47], and incorporation of heuristic rules such as linear ranking [48] and binary tournament [49] to compare individuals in the population. The main drawbacks of these approaches include the need to develop problem-specific repair mechanisms,

the unavailability of a feasible starting point, and early loss of diversity.

- **Simulated annealing (SA):** While many of the above mentioned constraint handling techniques have been used with EAs extensively, not many efforts have been made to solve difficult constrained multi-objective problems using SA. This may be attributed to the single-point nature of its search and also to the fact that, for multi-objective problems, defining the "energy function" to be minimized becomes a problem in itself. However, some recent works[50, 51, 2, 14, 52, 53, 54] have shown that with certain enhancements, SA can be used to solve multi-objective or constrained problems. The robustness of SA as an optimizer stems from its unique ability to accept worse or *uphill* solutions during the search in contrast to most other single-point methods.

Based on the work done on constraint handling in the literature, two key ideas for dealing with constrained problems effectively are explored in this chapter. Both employ non-greedy search strategies, which are more suited for solving difficult constrained problems (with irregular features such as discontinuous search space, narrow feasible regions etc.) than greedy strategies as they are less prone to getting stuck in a local optimum.

1. **Infeasibility Driven Evolutionary Algorithm (IDEA):** Most existing EAs consider a feasible solution to be better than an infeasible solution during the ranking process. While this preference is justified as the final aim is to get a feasible set of Pareto-optimal solutions, it may adversely affect the convergence rate of the algorithm. This is because for constrained optimization problems, the optimum solution(s) are very likely to lie on the constraint boundary; in which case an infeasible solution near the constraint

boundary may be more beneficial to guide the search than a feasible solution far away from it.

2. **Constrained Pareto Simulated Annealing (C-PSA):** Owing to the convenient applicability of MOEAs to constrained multi-objective problems, single-point search methods such as SA have rarely been employed for such problems. The key characteristic that differentiates SA from other single-point methods is that it allows transitions to *worse* or *uphill* solutions probabilistically during a search. This makes it less prone to getting stuck in a local optimum and, consequently, an attractive choice to solve optimization problems. In the present work, conventional SA is enhanced to deal with multi-objective constrained optimization problems.

The above two concepts are detailed in the following sections. Numerical experiments on benchmark problems are included, which highlight the performance and the benefits of the proposed approaches.

3.2 Infeasibility Driven Evolutionary Algorithm

As evident from a number of previous studies [35, 33, 34, 36, 37, 39, 38], information from infeasible solutions can be utilized to improve convergence. In addition, a designer is often interested in exploring solutions that might be marginally infeasible. This is because some of the constraints might have been imposed artificially in order to set up the problem but, in reality, can be compromised marginally if it results in a significant gain in the objective values. A classic case is a budget constraint modeled so that the cost is less than q. However, with negotiation, a cost of $q+\delta q$ may be reasonably acceptable. Of course, for example, some constraints that may perhaps deal with physics of a problem cannot be

violated. The proposed algorithm, IDEA, is aimed at delivering: (a) the set of optimal solutions (best objective values for single-objective and Pareto fronts for multi-objective problems); (b) a few *marginally* infeasible solutions for trade-off studies; and (c) an improvement in the rate of convergence by effectively utilizing the infeasible solutions during the search. Usually, as there are no defining limits on the absolute values of constraint violations, the term *marginal* is used to denote the solutions that have relatively small constraint violations among the members of the population, and are likely to lie close to the constraint boundary. The performance of IDEA is compared with NSGA-II on a number of SO problems (g-series problems [55, 56]) and MO problems (CTP series problems [57]). IDEA is outlined in Algorithm 3.1. The key steps involved in the algorithm are discussed in the following subsections.

Algorithm 3.1 Infeasibility Driven Evolutionary Algorithm (IDEA)

Require: N {Population Size}
Require: $N_G > 1$ {Number of Generations}
Require: $0 < \alpha < 1$ {Proportion of infeasible solutions}
1: $N_{inf} = \alpha * N$
2: $N_f = N - N_{inf}$
3: $pop_1 =$ Initialize()
4: Evaluate(pop_1)
5: **for** $i = 2$ to N_G **do**
6: $childpop_{i-1} =$ Evolve(pop_{i-1})
7: Evaluate($childpop_{i-1}$)
8: $(S_f, S_{inf}) =$ Split($pop_{i-1} + childpop_{i-1}$)
9: Rank(S_f)
10: Rank(S_{inf})
11: $pop_i = S_{inf}(1, N_{inf}) + S_f(1, N_f)$
12: **end for**

3.2.1 Problem reformulation

A generic optimization problem can be posed as shown in Equation 2.1 in Chapter 2. Optimal solutions to constrained optimization problems often lie along the constraint boundary. Therefore, in the proposed approach, to effectively search along the constraint boundary, the original k objective constrained optimization problem is reformulated as a $k+1$ objective unconstrained optimization problem, as given in Equation 3.1. The first k objectives are the same as those in the original constrained problem. The additional objective represents the amount of the constraint violation, which is referred to as the *Constraint Violation Measure (CVM)* in the present study. The details of the calculation of CVM are discussed later in Section 3.2.4.

$$\begin{aligned} \text{Minimize} \quad & f'_1(\mathbf{x}) = f_1(\mathbf{x}), \ldots, f'_k(\mathbf{x}) = f_k(\mathbf{x}) \\ & f'_{k+1}(\mathbf{x}) = \text{CVM} \end{aligned} \tag{3.1}$$

3.2.2 Evolution

The generation of offspring in IDEA is done the same way as in NSGA-II. The parents are selected through binary tournament. For crossover, Simulated Binary Crossover (SBX) [10] operator is applied (variable by variable), given in Equation 3.2.

$$\begin{aligned} y_i^1 &= 0.5\left[(1+\beta_{q_i})\, x_i^1 + (1-\beta_{q_i})\, x_i^2\right] \\ y_i^2 &= 0.5\left[(1-\beta_{q_i})\, x_i^1 + (1+\beta_{q_i})\, x_i^2\right] \end{aligned} \tag{3.2}$$

where β_{q_i} is calculated as,

$$\beta_{q_i} = \begin{cases} (2u_i)^{1/\eta_c+1}, & \text{if } u_i \leq 0.5, \\ \left(\frac{1}{2(1-u_i)}\right)^{1/\eta_c+1} & \text{if } u_i > 0.5. \end{cases} \tag{3.3}$$

and where u_i is a uniform random number in the range $[0, 1)$ and η_c is the user defined parameter *Distribution Index for Crossover*. The subscript i in the parent(x) and child(y) solutions refer to the i^{th} decision variable. For mutation, a polynomial mutation [11] operator, as defined in Equation 3.4, is used.

$$y_i = x_i + (\overline{x_i} - \underline{x_i})\,\bar{\delta}_i \tag{3.4}$$

where $\bar{\delta}_i$ is calculated as,

$$\bar{\delta}_i = \begin{cases} (2r_i)^{1/(\eta_m+1)} - 1, & \text{if } r_i < 0.5, \\ 1 - [2(1-r_i)]^{1/(\eta_m+1)}, & \text{if } r_i \geq 0.5. \end{cases} \tag{3.5}$$

where r_i is the uniform random number in the range $[0, 1)$ and η_m is the user defined parameter *Distribution Index for Mutation*.

3.2.3 Ranking and reduction

The main difference between NSGA-II and IDEA is the mechanism for elite preservation. In IDEA, a few infeasible solutions are retained in the population at every generation. Individual solutions in the population are evaluated as per the original problem definition (Equation 2.1) and marked infeasible if any of the constraints are violated. The solutions in the parent and the offspring population are divided into a feasible set (S_f) and an infeasible set (S_{inf}). The solutions in

the feasible and the infeasible sets are both ranked using non-dominated sorting and crowding distance sorting (Algorithm 3.2) of $k+1$ objectives. On the other hand, NSGA-II uses non-dominated sorting and crowding distance for ranking feasible solutions and ranks the infeasible solutions according to the increasing value of maximum constraint violation. For the feasible solutions, non-dominated sorting using $k+1$ objectives is equivalent to non-dominated sorting using the original k objectives, as the additional objective value (which is based on the constraint violations) for feasible solutions is always 0.

Algorithm 3.2 Crowding distance sorting mechanism

Require: F {Non-dominated set}
1: $N_s = |F|$ {Number of solutions in the non-dominated set}
2: $M =$ Number of objectives
3: $F(i).dist = 0 \quad \forall \quad i = 1, 2, \ldots N_s$ {Initialize distance}
4: **for** $m = 1$ to M **do**
5: $F = sort(F, m)$ {Sort based on objective value}
6: $F(1).dist = F(N_s).dist = \infty$ {Assign infinity to the corner points}
7: **for** $i = 2$ to $(N_s - 1)$ **do**
8: $F(i).dist = F(i).dist + (F(i+1, m) - F(i-1, m))/(f_m^{max} - f_m^{min})$ {calculate *F(i).dist* based on neighboring points}
9: **end for**
10: **end for**
11: Higher $dist \Rightarrow$ Higher rank

The next step is to choose the solutions that form the population for the next generation. In IDEA, a user-defined parameter α is used to identify the proportion of the infeasible solutions to be retained in the population. The numbers N_f $(= (1 - \alpha) \times N)$ and N_{inf} $(= \alpha \times N)$ denote the number of feasible and infeasible solutions in the population respectively, where N is the population size. If the infeasible set S_{inf} has more than N_{inf} solutions, the first N_{inf} solutions are selected based on their ranking; otherwise all the solutions from S_{inf} are selected. The rest of the solutions are selected from the feasible set

S_f, provided there are at least N_f feasible solutions. If S_f has fewer solutions, all the feasible solutions are selected and the rest are filled with infeasible solutions from S_{inf}. The solutions are ranked from 1 to N in the order they are selected. Hence, the infeasible solutions selected first (at most N_{inf}) receive a higher rank than the feasible solutions.

As an example, assuming a population size of 100, during any given generation 100 child solutions will be created. In the pool of 200 (parent + child) solutions, if there are 40 infeasible and 160 feasible solutions, then NSGA-II will select the best 100 feasible solutions for the next generation, hence preferring *all* feasible solutions over *all* infeasible solutions. On the other hand, assuming $\alpha = 0.2$, IDEA would select the 20 best infeasible solutions (based on non-dominated + crowding distance sorting of $k + 1$ objectives) and the 80 best feasible solutions to form the new population. Hence, *good* infeasible solutions are preferred over feasible solutions during the course of evolution.

In NSGA-II, the elite preservation mechanism weeds out the infeasible solutions from the population. To retain the infeasible solutions in the population, an alternate mechanism is required. In IDEA, the "good" infeasible solutions are ranked higher than the feasible solutions, thus adding selection pressure to generate *better* infeasible solutions. The presence of infeasible solutions with higher ranks than the feasible solutions translates into an active search through the infeasible space, in addition to the search through the feasible region. This feature of IDEA accelerates the movement of solutions towards the constraint boundary. With the modified problem definition and by ranking the infeasible solutions higher than the feasible solutions, IDEA can find the solutions to the original problem more efficiently and effectively.

3.2.4 Constraint Violation Measure (CVM)

The additional objective, CVM, in the modified problem formulation is based on the amount of relative constraint violations among the population members. Consider one of the constraints (g_i). All solutions in the population are sorted in ascending order based on the value of the constraint violation for g_i. The solutions that do not violate the constraint g_i are assigned a constraint violation value of 0 (and g_i does not contribute to the CVM of those solutions). The rest of the solutions are assigned constraint violation for the constraint g_i based on the sorted list, starting with rank 1 for the solution with least constraint violation. Solutions with the same value of constraint violation get the same rank. This ranking procedure is repeated for all constraints. The CVM for each solution is then calculated as the sum of the ranks (based on their constraint violations) obtained for all the constraints.

The process of determining CVM is illustrated using the following example. Consider an optimization problem with three constraints (C_1, C_2 and C_3). A sample population of 10 individuals is shown in Table 3.1. For each individual, the first three columns list the value of the constraint violation. The constraint violation values are sorted for each constraint and each individual is assigned relative ranks for each constraint, which are shown in next three columns. Individuals 3, 7 and 9 do not violate constraint C_1 and get a relative rank of 0. Individual 4 with the least constraint violation value (1.25) for C_1 gets rank 1 and individual 6 with the highest constraint violation value (100.70) is assigned rank 7. The last column shows the CVM which is the sum of the ranks with respect to each constraint.

It can be seen that the CVM favors individuals with good ranks for most (or all) of the constraints. As a result, an individual with large violation in only one

Table 3.1: Calculation of Constraint Violation Measure (CVM)

	Violations			Relative ranks			
Individual	C_1	C_2	C_3	C_1	C_2	C_3	CVM
1	3.50	90.60	8.09	3	8	7	18
2	5.76	7.80	6.70	4	6	5	15
3	0.00	3.40	7.10	0	4	6	10
4	1.25	0.00	0.69	1	0	1	2
5	13.75	90.10	5.87	6	7	4	17
6	100.70	2.34	3.20	7	3	2	12
7	0.00	5.09	4.76	0	5	3	8
8	1.90	0.00	0.00	2	0	0	2
9	0.00	0.56	0.00	0	1	0	1
10	8.90	2.30	9.80	5	2	8	15

of the constraints will have roughly the same preference as an individual with marginal violations of multiple constraints. The scheme tries to incorporate the amount of constraint violation and not just the number of violated constraints as in CHEA [40]. As a consequence of using CVM (as an additional objective) to rank infeasible solutions, the final population contains some solutions with marginal constraint violations.

3.2.5 Numerical experiments

The performance of IDEA is studied and compared against NSGA-II. Two sets of benchmark problems are chosen for the experiments. These problems and the corresponding numerical experiments are as follows:

1. **Single-objective optimization:** The g-series problems g01, g02, g04, g06, g07, g08, g09, g10 and g12 (g-series problems without equality constraints) are used as single objective optimization test problems. The mathematical formulations of the g-series test problems can be found in Appendix A. Thirty independent runs of IDEA and NSGA-II are performed. The parameters used are listed in Table 3.2.

Table 3.2: Parameters used for IDEA and NSGA-II for studies on g-series and CTP test functions

Parameter	Value
Crossover probability	0.9
Mutation probability	0.1
Crossover distribution index	15
Mutation distribution index	20
Infeasibility ratio α (for IDEA)	0.2

A population of size 200 is used and both algorithms are run for 1750 generations for all test runs, resulting in 350,000 function evaluations.

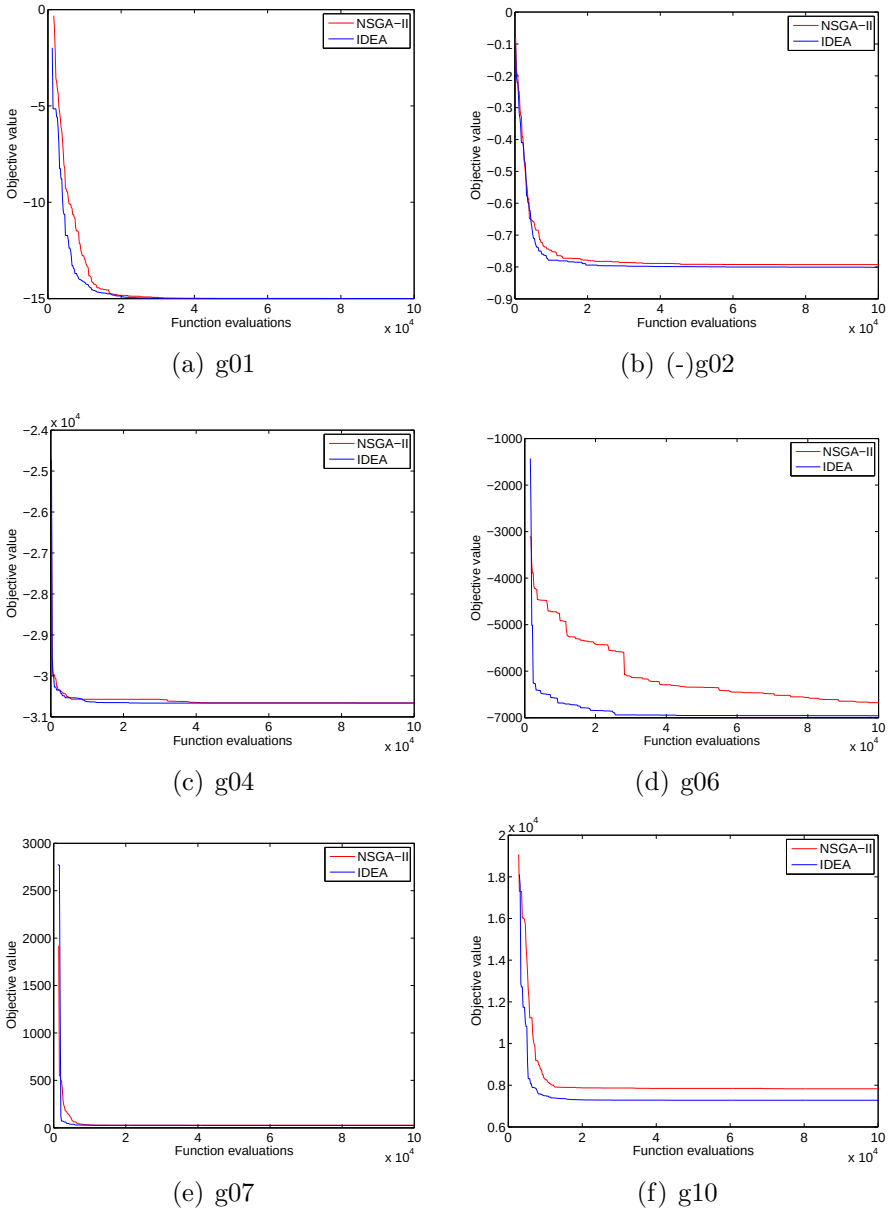
2. **Multi-objective optimization:** For multi-objective experiments, problems from the CTP series are chosen. CTP series is a set of seven constrained bi-objective optimization problems. CTP problems pose various challenges to optimization algorithms – constricted feasible spaces near the constraint boundaries, discontinuous Pareto fronts, discontinuity in the variable and function space, etc. A detailed discussion on CTP test functions can be found in [57]. The problem formulations are given in Appendix B.

   For CTP problems, thirty independent runs are performed using IDEA and NSGA-II by varying the random seed. The crossover and mutation parameters are the same as shown in Table 3.2. A population size of 200 is evolved over 200 generations.

Results: g-series test problems

The results of the runs for the g-series test problems are compared using two attributes – convergence rate and best solution obtained. Furthermore, trade-off solutions obtained for one of the problems (g06) are also analyzed.

1. **Convergence rate**: The median runs of both algorithms for the g-series test problems are shown in Figure 3.1. The plots are shown only for the first 100,000 evaluations in order to aid visualization.



(a) g01 (b) (-)g02

(c) g04 (d) g06

(e) g07 (f) g10

Figure 3.1: Median runs for g-series problems

It is seen in Figure 3.1 that the convergence rate of IDEA is better than

NSGA-II for problems g01, g02, g04, g06, g07 and g10. As for problems g08, g09 and g12, the average convergence was found to be almost identical; and the figures for these problems are not presented.

2. **Converged values**: The mean, best and worst objective values obtained across all runs are listed in Table 3.3. It is seen that for the given number of function evaluations, IDEA consistently obtains better or equal objective values than NSGA-II does, for all problems. For the problems g08 and g12, the average objective values obtained by both algorithms are the same.

Table 3.3: Results for g-series functions

	NSGA-II Results			IDEA Results		
	Best	Mean	Worst	Best	Mean	Worst
g01	-14.9999	-14.9322	-12.9995	-15	**-14.9333**	-12.9999
g02	0.803283	0.791741	0.759453	0.803145	**0.80241**	0.794207
g04	-30665.1	-30660.7	-30649.8	-30665.5	**-30665.5**	-30665.4
g06	-6945.72	-6913.94	-6882.47	-6961.79	**-6961.55**	-6959.85
g07	24.5051	25.65	27.9891	24.3772	**25.2732**	27.7785
g08	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825	-0.095825
g09	680.681	681.508	683.304	680.716	**681.052**	682.153
g10	7085.13	8006.19	9616.54	7058.42	**7390.4**	7990.62
g12	1	1	1	1	1	1

It is worth mentioning here that although the reported values for the g-series functions using IDEA are an improvement over those of NSGA-II, some recent studies have reported better results (than both these algorithms) for g-series functions [58, 59, 39]. However, the scope of most of these studies is limited to single-objective optimization.

3. **Trade-off solutions**: The problem g06 is chosen to illustrate the trade-off solutions obtained from IDEA. The constraint boundary for g06 is formed by two intersecting circles, shown in Figure 3.2. The feasible space is formed by the narrow region enclosed between the points of intersection of the two

circles. The magnified view of the search space near the intersection is shown in Figure 3.3. The optimum objective value occurs at the intersection (14.095, 0.84296). The best solutions obtained by IDEA and NSGA-II for problem g06 across all runs are shown in Figure 3.3. In addition, the high ranked infeasible solutions in the final IDEA population are also shown. All NSGA-II solutions lie in the feasible region as expected, whereas the (feasible) IDEA solutions are converged very near to the intersection (optimum) point. It is clear that NSGA-II has difficulty in searching along the narrow region of the feasible space. The population of IDEA, however, approaches the optimum solution from various directions (as apparent from the distribution of the infeasible solutions around the intersection) and manages to reach the optimum consistently.

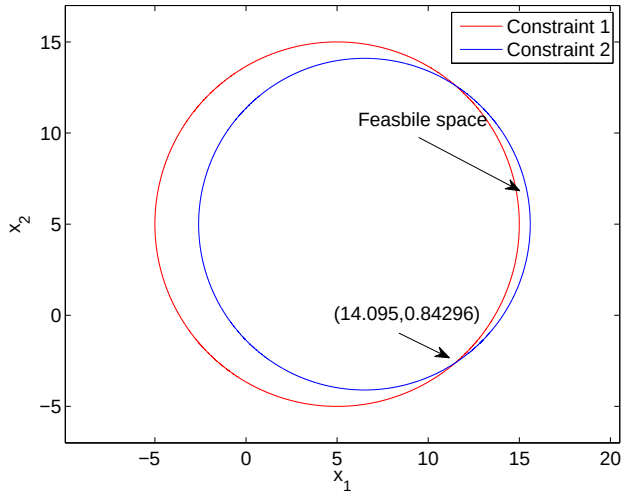


Figure 3.2: Search space and constraints for g06. The optimum solution is (14.0295,0.84296). Figure not to scale.

Some of the infeasible solutions in the final population obtained using IDEA are listed in Table 3.4. The optimum objective value for g06 is -6961.81388. The

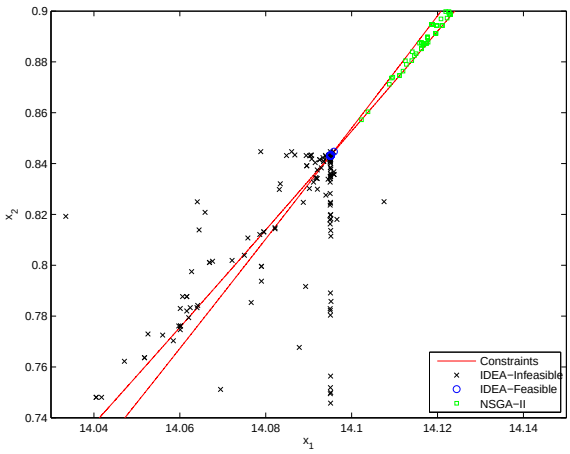


Figure 3.3: Final population for g06

objective can be improved substantially by relaxing one or both the constraints marginally, as seen from the table.

Table 3.4: Marginally infeasible solutions for g06 obtained using IDEA

			Violations	
x_1	x_2	$f(\mathbf{x})$	C_1	C_2
13.603	4.906e-15	-7953.2	0.98347	3.9606e-05
14.062	0.77939	-7033.7	0.065318	0.00054101
14.075	0.80391	-7005.9	0.036576	0.0033569
14.093	0.83829	-6967.1	0.0015724	0.0056683
14.064	0.7842	-7028.2	0.067479	0.0058579
13.609	0.011038	-7939.8	0.99474	0.022815
14.072	0.80192	-7008.2	0.071612	0.025974
14.09	0.83017	-6976.1	0.018567	0.028237
14.082	0.81432	-6994	0.0057509	0.031406
14.031	0.72543	-7095.2	0.1635	0.036114

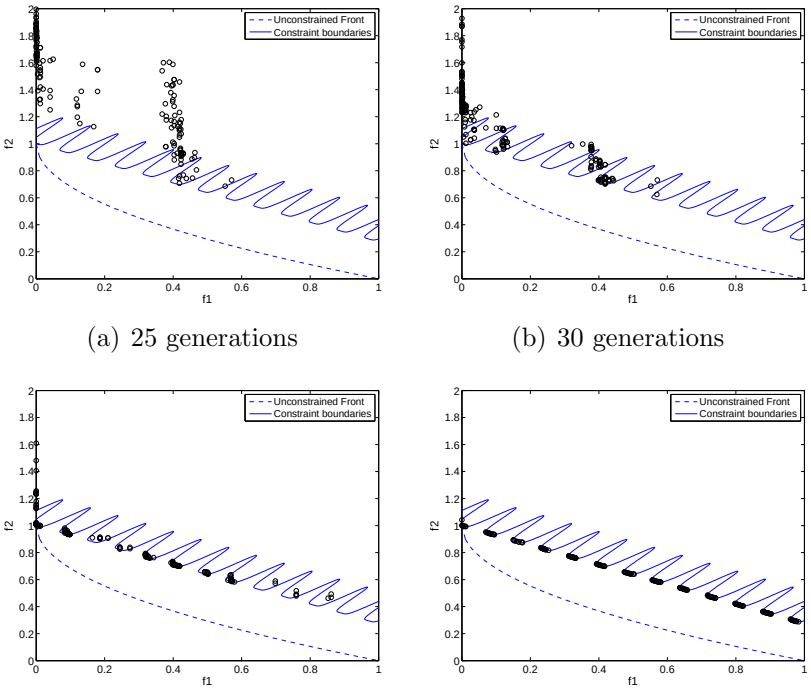
Results: CTP-series test problems

1. **Evolution**: The progress plots of NSGA-II and IDEA populations for up to 200 generations for test problem CTP2 are shown in Figure 3.4 and Figure 3.5 respectively. For NSGA-II, the population approaches the Pareto front from the feasible space and has difficulty in searching close to the constraint boundary. On the other hand, IDEA maintains the population

in both the feasible and the infeasible spaces, thus capturing the entire Pareto front much faster (See Figure 3.5). IDEA is able to cover most of the disconnected segments of Pareto front for CTP2 by generation 50.

Test problem CTP2 has a disconnected Pareto front. Any population based method that searches through the feasible space for a disconnected Pareto front, is likely to face difficulty capturing the whole front unless a reasonably large population size is chosen to maintain diversity. Once the entire population converges to fewer regions of the Pareto front, NSGA-II has to rely predominantly on mutation to spread the solutions to other regions of the Pareto front. On the other hand, the IDEA population can move through the infeasible regions, thus avoiding 'detours' through feasible space. Hence, even with a relatively small population, IDEA is able to capture the entire Pareto front.

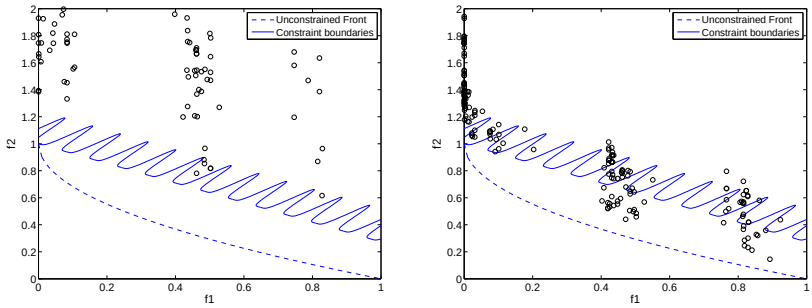
To illustrate this, a single run is performed for CTP2 with both NSGA-II and IDEA with a population size of 100 and 200 generations. The crossover and mutation parameters are kept fixed as in the earlier experiments. The results of the run are shown in Figure 3.6. It can be seen that IDEA solutions are spread much more evenly across the entire Pareto front as compared to NSGA-II solutions.



(a) 25 generations (b) 30 generations

(c) 50 generations (d) 200 generations

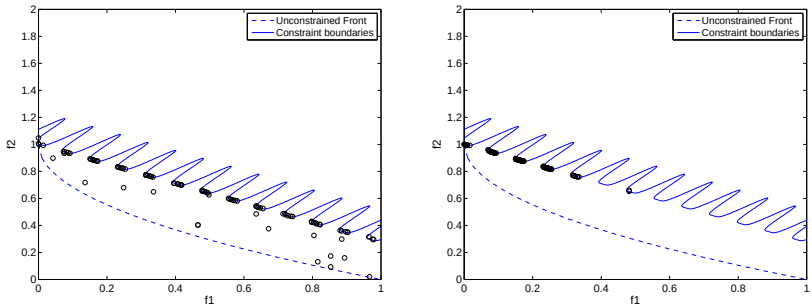
Figure 3.4: Evolution of NSGA-II population over generations for CTP2 test run (population size is 200)



(a) 25 generations (b) 30 generations

(c) 50 generations (d) 200 generations

Figure 3.5: Evolution of IDEA population over generations for CTP2 test run (population size is 200)



(a) IDEA (b) NSGA-II

Figure 3.6: Final fronts obtained for CTP2 using IDEA and NSGA-II (population size of 100 evolved over 200 generations)

2. **Performance metrics**: The performance metrics are calculated using the non-dominated solutions obtained by NSGA-II and IDEA. In the case of

Table 3.5: Displacement metrics for CTP problems

	IDEA Results			NSGA-II Results		
	Best	Mean	S.D.	Best	Mean	S.D.
CTP2	0.000101	**0.001254**	0.004286	0.000085	0.006415	0.009226
CTP3	0.001696	**0.006992**	0.017147	0.001820	0.024493	0.035827
CTP4	0.017625	**0.028247**	0.015927	0.019386	0.061003	0.047692
CTP5	0.000266	**0.001369**	0.003965	0.000268	0.003856	0.004762
CTP6	0.000463	**0.000689**	0.000924	0.000395	0.012739	0.050726
CTP7	0.000049	**0.004057**	0.013118	0.000041	0.008308	0.013071
CTP8	0.000251	**0.004592**	0.010069	0.000225	0.104338	0.137467

IDEA, only the feasible solutions in the final population are considered. The displacement and hypervolume metrics for NSGA-II and IDEA are listed in Tables 3.5 and 3.6 respectively for problems CTP2-CTP8. Shown are the show the best, mean, and Standard Deviation (S.D.) of the metric values across 30 runs. In the Table 3.5 it can be seen that IDEA obtains significantly better mean displacement metric values than NSGA-II for all CTP problems. The worse mean values of NSGA-II are due to its tendency to converge to sub-optimal fronts for CTP problems. For the best runs, the values of displacement metrics obtained by both algorithms are quite close. It is also observed that IDEA results exhibit lower S.D., indicating better consistency compared to NSGA-II results over multiple runs. Only for CTP7, the S.D. value obtained using IDEA is higher than that obtained NSGA-II. It is so because in *one* of the runs, IDEA converged to a sub-optimal front which is far away from the Pareto-optimal front. NSGA-II on the other hand, converged to a sub-optimal that was closer to the Pareto-optimal front, but went sub-optimal more times than IDEA. Hence, for CTP7, the mean value using IDEA is still better than that using NSGA-II, where as the S.D. is worse for IDEA.

The best, mean and the S.D. values for the hypervolume metric (across

all 30 runs) are shown in Table 3.6. It is seen that the average metric values obtained by IDEA are higher than those from NSGA-II for all CTP problems. The small values of S.D. from IDEA suggest that the performance of the algorithm is consistent. Also, it is seen that for the case of CTP7, the IDEA results have a higher S.D. than NSGA-II results, due to the same reason as given for the case of displacement metric results.

Table 3.6: Hypervolume metrics for CTP problems

	IDEA Results			NSGA-II Results		
	Best	Mean	S.D.	Best	Mean	S.D.
CTP2	3.059180	**3.011390**	0.177100	3.059344	2.870703	0.270056
CTP3	3.015969	**2.960771**	0.163811	3.010435	2.828100	0.254690
CTP4	2.919011	**2.744736**	0.139271	2.848500	2.438106	0.352716
CTP5	3.024724	**2.952929**	0.162089	3.020914	2.723520	0.292627
CTP6	36.819072	**36.787826**	0.075797	36.822693	36.182922	2.187300
CTP7	3.617663	**3.435931**	0.594506	3.617716	3.240162	0.594118
CTP8	36.180362	**35.970564**	0.434540	36.170783	32.085918	5.176305

3. **Trade-off solutions**: As shown in Figure 3.5(d), the final population for CTP2 evolved using IDEA contains infeasible points close to the constraint boundary. One can evaluate the benefits in the objective values by relaxing the constraints as done for g-series problems. For multi-objective optimization problems, significant benefits may be derived in multiple objectives at the cost of relaxing the constraints marginally.

3.2.6 Variations in performance with infeasibility ratio

To observe the effect of the infeasibility ratio (α) on the performance of IDEA, fifteen independent runs are conducted with different values of α, for the problem g06. The average convergence plots are shown in Figure 3.7. The parameters used are the same as listed in Table 3.2. It is seen that the performance of IDEA is consistent over a wide range of α. Even by maintaining a small proportion ($\alpha =$

0.05) of infeasible solutions in the population, significant improvement can be achieved in the convergence rate. For $\alpha = 0$, the performance of IDEA will be the same as that of NSGA-II. For multi-objective problems, a high value of α would lead to fewer solutions covering the Pareto front, and hence is not recommended.

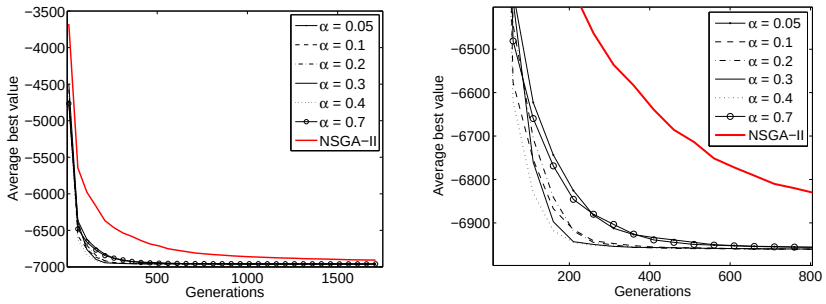


Figure 3.7: Variation of IDEA performance for problem g06 with change in α: (a) over all the generations, (b) during initial generations.

3.3 Constrained Pareto Simulated Annealing

Simulated Annealing is a heuristic that draws an analogy from the slow cooling process of metal atoms. It is an established robust optimization technique with a strong mathematical basis. Additionally, it has been proven that it converges to the global optimum if the annealing is sufficiently slow [13]. Although convergence is guaranteed only for a very slow cooling rate, in practice it has been observed that good results are obtained even with reasonably rapid cooling rates [52, 53]. The robustness of SA as an optimizer stems from its ability to accept unfavorable solutions probabilistically, which aids it in escaping local minima. However, in spite of its merits, the formulation of conventional SA is inherently suited for single-objective optimization problems as the definition of the "energy function"

can be directly associated with the objective to be minimized in this case. For multi-objective problems, defining the energy function becomes a problem in itself. Also, there is no implicit constraint handling mechanism available in conventional SA. There have been recent efforts to enhance SA to eliminate both the limitations, *i.e.*, constraint handling and multiple objectives.

For constraint handling in SA, the following approaches have been commonly used.

1. **Penalty function:** As with other algorithms, a penalty function approach is often employed (*e.g.* [54]) in the paradigm of SA. The penalty function method can be easily implemented by merely modifying the objective function, which makes it a simple and effective constraint handling technique. However, the drawback of using a penalty function is that a number of penalty parameters have to be chosen a priori in order to formulate it. The results obtained are often sensitive to the choice of these parameters.

2. **Rejection of infeasible solutions:** Another, more extreme way to weed out infeasible solutions is the "hard constraint" approach, in which all the proposals during the search that land on an infeasible solution [50] are rejected. While this technique can work for continuous feasible spaces, it is very likely to get stuck during the search if the feasible spaces are discontinuous.

3. **Approximate descent direction (ADD):** Recently, a derivative-free filtered simulated annealing (FSA) has been proposed by Hedar and Fukushima [2] in which filter set concept [60] coupled with ADD[61] is employed for constraint handling. In this approach, the trial solution is generated depending on whether the current solution is feasible or infeasible. If the

current solution is feasible, ADD is used to determine a direction in which the function value is likely to decrease, and subsequently, a trial solution is generated in that direction. If the current solution is infeasible, then ADD is used to determine a direction in which the constraint violation is likely to decrease, and a step is taken in that direction. A filter set is maintained and continuously updated, and employed to decide upon the acceptance of trial solutions. The efficacy of the algorithm was demonstrated on a number of single-objective g-series test problems.

In order to enhance conventional SA for handling multiple objectives, a number of studies have been undertaken in recent years. They fall into three broad categories:

1. **Scalarizing function scheme:** In this scheme, a scalar objective is formed by a weighted aggregation of the multiple objectives, which is then used as an energy function in SA [62]. Although solutions forming the Pareto front can be found using the scalarizing function method, one run of SA renders only one solution and, hence, a number of runs have to be made in order to get the required number of solutions on the Pareto front. A more prominent limitation of using a weighted sum approach is that the non-convex part of the front cannot be recovered using any combination of weights.

2. **Composite energy difference:** As an alternate to scalarizing, methods based on composite energy difference calculation [50, 63, 16] have been suggested, in which the probability of acceptance of a trial solution is calculated as a function of acceptance probabilities for individual objectives. Nam and Park [51] studied six different criteria for calculating the cost function from the change in the individual objective values. While using

composite energy techniques, care must be taken to scale the individual objectives (or cost criterion) such that the search doesn't become biased towards particular objective(s). This can also be handled by using different annealing temperatures for different objectives, as in [50].

3. **Pareto-dominance based schemes:** To overcome the difficulties associated with scalarizing techniques and composite functions, attempts have been made by various researchers to incorporate the concept of Pareto-dominance in SA [16, 64, 53, 52, 54, 14].

   In the earlier proposals [64, 54, 53] the energy function is calculated based on the number of solutions (in the non-dominated set) that dominate a trial solution under consideration. In the recently proposed Archive based Multi-objective Simulated Annealing (AMOSA) [14], the amount of dominance is defined in terms of hypervolume in function space enclosed by the points to be compared. The acceptance scheme takes into account all possible cases of the domination status between the current solution, trial solution and the solutions in the existing archive of non-dominated solutions. Comparisons were done with NSGA-II [65], PAES [66] and MOSA [64] on a number of benchmark problems in order to establish the benefits of AMOSA.

Although studies have been undertaken on handling constraints and multiple objectives separately, very few studies have focused on solving difficult constrained MO problems. As mentioned earlier, schemes such as penalty function or a "hard constraint" approach, which are either dependent on a number of parameters or are very limited in scope, have been used. Therefore, they not suitable for a number of constraint optimization problems. In the present work, a SA

algorithm is proposed for constrained multi-objective problems. The acceptance criteria of a trial solution includes the feasibility status of the current and the trial solutions, in addition to the domination status as considered in AMOSA. A comparison with two MOEAs, IDEA (discussed in the previous section), and NSGA-II [65] is presented on a benchmark set of problems (CTP series), in order to highlight the benefits of the proposed approach.

Before discussing the C-PSA algorithm, a brief background of AMOSA [14] is given, as similar concepts of domination are incorporated in the proposed algorithm.

Archive based multi-objective simulated annealing (AMOSA)

AMOSA is a dominance based SA method. A prominent difference between AMOSA and most of the earlier SA techniques is that for acceptance of a trial solution, the domination status of the point is considered not only with respect to the current solution, but also the archive of non-dominated solutions found during the search. Another salient feature of AMOSA is that it compares the solutions based on a dominance measure instead of on the number of solutions that dominate the points as done in previous studies [54, 64]. Given two solutions a and b, where a dominates b, the amount of domination is defined by Equation 3.6.

$$\Delta dom_{a,b} = \prod_{i=1, f_i(a) \neq f_i(b)}^{n_{obj}} \left(\frac{|f_i(a) - f_i(b)|}{R_i} \right), \tag{3.6}$$

where n_{obj} is the number of objectives and R_i the range of i^{th} objective. R_i is determined using the solutions in the archive (set of non-dominated solutions found so far in the search), the current and the proposed trial solution.

The AMOSA search is initialized with a certain number of solutions, which

are refined over a few iterations by using a simple strategy in which a solution is accepted only if it dominates the previous solution. The non-dominated solutions obtained using this technique are used to initialize the archive of non-dominated solutions. The main search is then started from a random solution in the archive. Laplacian perturbation is used to create a trial solution. Thereafter, the domination status of the current solution, proposed solution and the archive is analyzed, based on which a number of cases can arise. From the exhaustive list of cases, acceptance is calculated based on the one which applies. The set of non-dominated solutions (archive) is maintained and continuously updated during the search.

Whenever an unfavorable move is considered for acceptance, the probability of acceptance is calculated as

$$prob = 1/(1 + \exp(\Delta dom \times T)),$$

where T is the temperature, and Δdom is calculated based on the dominance relations between the current and trial solutions, and the archive.

During the search, to enforce diversity and limit the number of solutions in the archive to a desired value, clustering is performed if needed. The archive is allowed to grow to a prescribed *soft limit* SL, after which the clustering is done to reduce the number of solutions to a prescribed *hard limit* HL.

Following this brief description of the AMOSA algorithm, the proposed C-PSA, which is outlined in Algorithm 3.3, is discussed next. While operating in the feasible region, acceptance of the trial point is determined based on the domination status of the current solution, the trial solution as well as the archive of non-dominated solutions, similar to AMOSA (but with a key difference, as described later). In a situation where one or both the solutions are infeasible,

the acceptance criterion takes into account the feasibility status of the current and trial solutions, as well as the constraint violation. Details are discussed in the following subsections.

Initialization of archive

At the start of the search, the archive does not contain any solution. For simplicity, a random solution within the variable space is chosen as a starting point of the search. If the starting point is a feasible solution, the archive is initialized with this solution. Otherwise, the archive is initialized when the first feasible solution is found. More sophisticated initialization techniques, such as the scatter search diversification generation method [67, 68, 2] also exist in the literature, and can be used to initialize the archive instead.

Trial solution generation

Two different procedures are used to generate a trial solution, depending on whether the current solution is feasible or infeasible.

1. If the current solution is feasible, Laplacian mutation is performed as suggested in [64, 14]. One decision variable is chosen at random and perturbed by a random variable ϵ drawn from the Laplacian distribution $p(\epsilon) \propto e^{-\|\sigma\epsilon\|}$, where σ represents the spread of the perturbation. To effectively search the variable space, the σ is scaled throughout the search. Initially it is set to be sufficiently high for the algorithm to traverse the whole range of a variable. As the iterations proceed, σ is exponentially reduced to a small value.

   This scheme of updating σ is aimed at aiding the search in the following ways: (a) during the initial stages, it will help the algorithm to conduct

Algorithm 3.3 Constrained Pareto Simulated Annealing (C-PSA)

Require: N, M, T_{max}, T_{min}, T_G, HL, SL, α, P_i, P_f, σ_i, σ_f

Initialize archive
Set $T = T_{max}$, $\mathbf{x_{old}}$ = random solution in search space
for $i = 1$ to N **do**
  for $j = 1$ to M **do**
    $\mathbf{x_{new}} = perturb(\mathbf{x_{old}})$ {Laplacian perturbation if $\mathbf{x_{old}}$ is feasible, ADD perturbation otherwise}
    if both $\mathbf{x_{old}}$, $\mathbf{x_{new}}$ are feasible **then**
      Follow procedure described in Algorithm 3.4.
    else if $\mathbf{x_{old}}$ is feasible, $\mathbf{x_{new}}$ is infeasible **then**
      $prob = P_i * (P_f/P_i)^{i/N}$
      Set $\mathbf{x_{old}} = \mathbf{x_{new}}$ with a probability $prob$
    else if $\mathbf{x_{old}}$ is infeasible, $\mathbf{x_{new}}$ is feasible **then**
      Set $\mathbf{x_{old}} = \mathbf{x_{new}}$
      if $\mathbf{x_{new}}$ is non-dominated w.r.t archive **then**
        Add $\mathbf{x_{new}}$ to the archive
      else if $\mathbf{x_{new}}$ dominates points in archive **then**
        Add $\mathbf{x_{new}}$ to the archive.
        Remove all dominated points from archive
      end if
    else if both $\mathbf{x_{old}}$, $\mathbf{x_{new}}$ are infeasible **then**
      if $g_{old} \leq g_{new}$ **then**
        Set $\mathbf{x_{old}} = \mathbf{x_{new}}$
      else
        $prob = exp(-(g_{new} - g_{old})/T_G)$
        Set $\mathbf{x_{old}} = \mathbf{x_{new}}$ with a probability $prob$
      end if
    end if
    if K_{max} solutions are rejected consecutively **then**
      Restart.
    end if
    If |archive| $\geq SL$, cluster to HL
  end for
  Update T, T_G, σ
end for
If |archive| $\geq SL$, cluster to HL

a *random walk* through the search space, thereby identifying region(s) of potential optima very swiftly; since it can be expected that the starting point may be far from the global minimum and the search may waste a considerable amount of time getting to an optimal region if the step size is always small. (b) during the later stages of the SA run, it will be required to fine-tune the variable values to get as close to the Pareto front as possible. At this stage, a large step size will be detrimental to the search as it may not produce any better solutions than those already existing in the archive. For this reason, a small value of σ is used to increase the chances of improving the solutions in the final stages.

2. If the current solution is infeasible, an effort is made to move along a direction that will reduce the constraint violation. The ADD method, suggested in [61, 2], is used to achieve this. Given a scalar function F, ADD is a derivative-free method which estimates the direction in which F is likely to decrease, by exploring a few solutions around the current solution. If the current solution is denoted by $\mathbf{x}$, then ADD first generates p solutions $\{\mathbf{y_i}\}_{i=1}^{p}$, within a small radius r from $\mathbf{x}$. The descent direction v is calculated as

$$
\begin{aligned}
v &= \sum_{i=1}^{p} w_i \mathbf{e_i} \\
&\text{where} \\
w_i &= \Delta F_i / \sum_{j=1}^{p} |\Delta F_j| \\
\mathbf{e_i} &= -(\mathbf{y_i} - \mathbf{x})/(||\mathbf{y_i} - \mathbf{x}||) \\
\Delta F_i &= F(\mathbf{y_i}) - F(\mathbf{x})
\end{aligned}
\tag{3.7}
$$

It can be seen from Equation 3.7 that if at an exploring solution $\mathbf{y_i}$, $F(\mathbf{y_i}) > F(\mathbf{x})$, $w_i\mathbf{e_i}$ constitutes a direction *opposite* to $(\mathbf{y_i} - \mathbf{x})$, since ΔF is negative. On the other hand, if $F(\mathbf{y_i}) < F(\mathbf{x})$, then $w_i\mathbf{e_i}$ constitutes a direction inline with $(\mathbf{y_i} - \mathbf{x})$. Thus, for each $\mathbf{y_i}$, $w_i\mathbf{e_i}$ is a direction in which F is likely to decrease. The approximate descent direction v is calculated as a linear combination of all these directions. The weights w_i are calculated based on the magnitude of the change in function at a given point $\mathbf{y_i}$ from $\mathbf{x}$, normalized with the total magnitude of change summed over all exploring points, as shown in Equation 3.7. An example of constructing an approximate descent direction using two exploring points is shown in Figure 3.8.

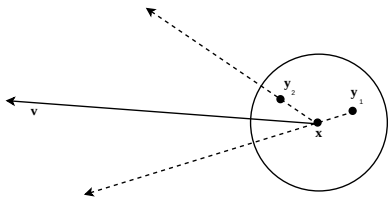


Figure 3.8: Illustration of calculation of approximate descent direction. Here $F(\mathbf{y_1}) > F(\mathbf{x})$, and $F(\mathbf{y_2}) < F(\mathbf{x})$. (Figure taken from [2])

In the case of optimization problems in which multiple constraints are present, it is a common practice to use one composite constraint value $g(\mathbf{x})$ for a solution $\mathbf{x}$ instead. It can be calculated by either :

- summing up all the constraint violation values : $g(\mathbf{x}) = \sum_{i=1}^{N_g} g_i(\mathbf{x})$, where N_g is the number of constraints, or
- taking the maximum constraint violation as the overall constraint violation of the solution :$g(\mathbf{x}) = \max(g_i(\mathbf{x})), i = 1, 2, \ldots N_g$.

For a constraint that is not violated, the value $g_i(\mathbf{x})$ is zero. Hence, for a feasible solution (which does not violate any of the constraints), the constraint violation value $g(\mathbf{x})$ is zero, whereas for a solution that violates any of the constraints, $g(\mathbf{x})$ is positive.

In the present studies, the **maximum constraint violation** is considered as the overall constraint violation ($g(\mathbf{x})$ of a solution $\mathbf{x}$. If the current solution is infeasible, it is desirable that the search progress so as to reduce the constraint violation, in order to eventually get to a feasible region. Therefore, whenever the current solution is infeasible, ADD is used to determine a direction which will reduce the constraint violation $g(\mathbf{x})$.

Solution acceptance criterion

For the case in which both the current solution ($\mathbf{x_{old}}$) and the new solution ($\mathbf{x_{new}}$) are feasible, the acceptance criteria as described in Algorithm 3.4 are used. For the cases in which the temperature (T) parameter does not play a role in the acceptance, it can be seen that the acceptance criteria are the same as those used in AMOSA. However, there is a significant difference in the case in which a potentially worse solution is being considered for acceptance. In AMOSA, the probability of acceptance is calculated as $1/(1 + exp(\Delta dom \times T))$. Since both Δdom and T are positive quantities, this expression implies that when T is high, the probability of acceptance is very low (behaving as a greedy search[1]) whereas, for the same given value of Δdom, when T is very low (towards the end of the SA run), the probability of acceptance will asymptotically converge to 0.5. The rationale behind using such a scheme is that, initially, the algorithm attempts to

[1] A greedy search is one which *always* prefers the locally optimal transitions in order to find the global optimum

obtain good solutions in the archive using a greedy search and, later (assuming a certain number of good solutions are already in the archive), an effort is made to explore the search space more exhaustively by increasing the probability of acceptance of worse solutions. This is contrary to the concept of a conventional SA in which the algorithm initially conducts a global search to identify a good region and later performs a (relatively) greedy search in a focused region in order to get close to the Pareto Front.

A greedy approach at the beginning of the algorithm (such as that used in AMOSA) may result in the search being attracted to a local optimum if the objective functions are multimodal. For some cases, this may be a beneficial approach, if the local optimum is not far from the global optimum, and the local optimum (or the global optimum itself) can be found during the early stages of the algorithm using a greedy search. However, this may not always be the case. Considering a more generic scenario in which convergence to a local optimum is detrimental to the global search (as it may require significant computational effort to escape the local optimum and then converge to the global optimum), a more conventional SA strategy is adopted here by setting the acceptance probability to $exp(-\Delta dom/T)$. The initial probability of acceptance is high, thereby giving even worse solutions a good chance of acceptance, while ensuring that the algorithm doesn't get stuck in any local optimum. In the later stages of the search, the probability of acceptance of worse solutions decreases, and the algorithm becomes greedy in nature in order to focus the search on the potentially optimal region of the search space. The variation in the probability of acceptance with SA iterations (for a fixed value of Δdom) is shown in Figure 3.9.

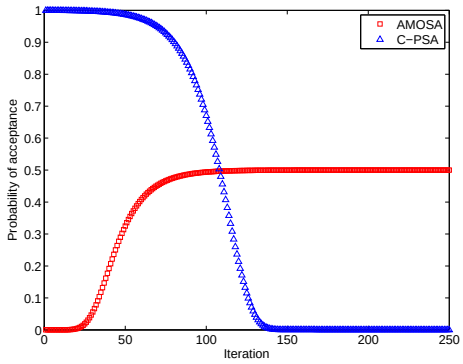


Figure 3.9: Probability of acceptance of a worse solution ($\Delta dom = 0.1$)

If one or both of the current solution ($\mathbf{x_{old}}$) and trial solution ($\mathbf{x_{new}}$) are infeasible, the constraint violation value (g) for multiple constraints is taken as the maximum constraint violation.

1. If $\mathbf{x_{old}}$ is infeasible and $\mathbf{x_{new}}$ is feasible, $\mathbf{x_{new}}$ is accepted as the current point. If none of the solutions in the archive dominates $\mathbf{x_{new}}$, then $\mathbf{x_{new}}$ is added to the archive and the solutions dominated by $\mathbf{x_{new}}$ are deleted from the archive.

2. If $\mathbf{x_{old}}$ is feasible and $\mathbf{x_{new}}$ is infeasible, then $\mathbf{x_{new}}$ is accepted with a probability given by Equation 3.8.

$$prob = P_i \times (P_f/P_i)^{i/N} \tag{3.8}$$

where, P_i and P_f are prescribed values of the initial and final probabilities of acceptance, i the current generation, and N the total number of generations. The algorithm is started with a high value of acceptance so that the search space can be adequately explored. Thereafter, the probability is reduced exponentially to a low value towards the end so that convergence is not

Algorithm 3.4 Acceptance Criterion for feasible → feasible jump

Check the domination status of $\mathbf{x_{old}}$ and $\mathbf{x_{new}}$.
if $\mathbf{x_{old}}$ dominates $\mathbf{x_{new}}$ **then**
  $\Delta dom_{avg} = \frac{(\sum_{i=1}^{k} \Delta dom_{i,new}) + \Delta dom_{old,new}}{k+1}$
  (k = number of points in the archive that dominate x_{new})
  $prob = exp(-\Delta dom_{avg}/T)$ (T = Current temperature)
  Set $\mathbf{x_{old}} = \mathbf{x_{new}}$ with a probability $prob$
else if $\mathbf{x_{old}}$ and $\mathbf{x_{new}}$ are non-dominating **then**
  if $\mathbf{x_{new}}$ is dominated by k (≥ 1) points in the archive **then**
    $\Delta dom_{avg} = \frac{(\sum_{i=1}^{k} \Delta dom_{i,new})}{k}$
    $prob = exp(-\Delta dom_{avg}/T)$
  else if $\mathbf{x_{new}}$ is non-dominating with respect to all points in the archive **then**
    Set $\mathbf{x_{old}} = \mathbf{x_{new}}$ and add it to the archive
  else if $\mathbf{x_{new}}$ dominates k (≥ 1) points in the archive **then**
    Set $\mathbf{x_{old}} = \mathbf{x_{new}}$ and add it to the archive
    Remove the k dominated points from the archive.
  end if
else if $\mathbf{x_{new}}$ dominates $\mathbf{x_{old}}$ **then**
  if $\mathbf{x_{new}}$ is dominated by k (≥ 1) solutions in the archive **then**
    Δdom_{min} = minimum of the difference of domination amounts between $\mathbf{x_{new}}$ and the k points
    $prob = \frac{1}{1+exp(-\Delta dom_{min})}$
    Set the point of the archive corresponding to Δdom_{min} as $\mathbf{x_{old}}$ with probability $prob$, else set $\mathbf{x_{old}} = \mathbf{x_{new}}$.
  else if $\mathbf{x_{new}}$ is non-dominating with all points in archive **then**
    Set $\mathbf{x_{old}} = \mathbf{x_{new}}$ and add it to the archive.
    If $\mathbf{x_{old}}$ is in the archive, remove it from the archive.
  else if $\mathbf{x_{new}}$ dominates k other points in the archive **then**
    Set $\mathbf{x_{old}} = \mathbf{x_{new}}$ and add it to the archive.
    Remove all dominated points from the archive.
  end if
end if

delayed due to the acceptance of infeasible solutions.

A probabilistic jump to an infeasible solution, even though the current solution is feasible, aids the search in escaping from the sub-optimal feasible regions. If the feasible regions appear as *islands* in the objective space, the only way the search can avoid getting trapped is by employing a sufficiently large step size, or traversing through the infeasible region by accepting the infeasible solutions. The drawback of using a large step size (throughout the search) is that it loosely translates to a random search, which is inefficient.

3. If both $\mathbf{x_{old}}$ and $\mathbf{x_{new}}$ are infeasible, the trial solution is accepted based on similar principles to those used in SA, and the probabilities are calculated using the constraint violation values.

   (a) If the constraint violation of $\mathbf{x_{new}}$ is less than that of $\mathbf{x_{old}}$ ($g_{new} \leq g_{old}$), the trial solution is accepted.

   (b) If the constraint violation of $\mathbf{x_{new}}$ is more than that of $\mathbf{x_{old}}$ ($g_{new} \geq g_{old}$), the trial solution is accepted with a probability $exp(-(g_{new} - g_{old})/T_G)$. Here, the parameter T_G here is synonymous to annealing temperature T, but its initial value is calculated based on the constraint violations of a few random trial solutions, instead of the objective values. Similar to that for T, an exponential decay schedule is used for T_G.

Restart mechanism

As opposed to population based methods, single-point methods do not have implicit means of maintaining diversity, which may result in the algorithm being unable to escape from a local minimum. To mitigate this problem, a restart

mechanism is added to SA. In C-PSA, the search is restarted from a solution in the archive whenever a new proposal is rejected a prescribed number of times ($Kmax$). Unlike some other restart methods in which temperature is reset [50, 2], the annealing temperature is kept unchanged for the restart solution. This is done in an effort to prevent the acceptance probability from increasing abruptly up due to high temperature, which might cause the search to accept too many unfavorable solutions. The restart point is chosen with equal probability as:

1. a random point from the archive; or
2. the most isolated point in the archive, which is identified as the one with the largest Euclidean distance from the rest of the points in the archive.

Archive update

The archive is updated whenever a new feasible solution is accepted. Unless the new point is dominated by points in the archive, it is added to the archive and all the dominated solutions are removed. If the number of solutions in the archive exceed a prescribed limit (soft limit SL), clustering is used to reduce the size of archive to a hard limit HL, similar to the procedure adopted in AMOSA. The limits can be set depending upon the computational resources available and/or the number of solutions desired in the final archive. A larger archive size will require more run-time and memory, primarily due to the increased number of non-domination checks.

3.3.1 Numerical experiments

To evaluate the performance of the proposed algorithm, thirty independent runs of C-PSA are conducted on CTP test problems. The average number of evaluations used by C-PSA are observed. Thereafter, comparisons are performed (for the same number of evaluations) with two MOEAs, namely NSGA-II [9] and IDEA (proposed in the previous section).

To compare the performance of the above mentioned algorithms with that of C-PSA, they are run for the average number of function evaluations taken by C-PSA[2]. The parameters used for C-PSA are listed in Table 3.7, and those used for IDEA and NSGA-II are listed in Table 3.8.

Table 3.7: Parameters used for the C-PSA

Parameter	Value
Initial probability of feasible to infeasible jump (P_i)	0.5
Final probability of feasible to infeasible jump (P_f)	0.01
Probability of acceptance used for calculating initial temperature (P_{T_i})	0.9
Final temperature (T_f)	1e-5
No. of exploring solutions for ADD (N_{add})	1
Exploration radius for ADD (r)	1e-3
Initial scaling factor for Laplacian mutation (σ_i)	1
Final scaling factor for Laplacian mutation (σ_f)	0.1
Epoch length (M)	$20 \times n_{var}$
Iterations (N)	200
Hard limit on no. of solutions in archive (HL)	200
Soft limit on no. of solutions in archive (SL)	300
Number of solutions rejected consecutively for restart ($Kmax$)	10

[2]In the current implementation, C-PSA is not tuned to complete the annealing schedule in a *given number of function evaluations*. A step from a feasible solution requires one evaluation, while that from an infeasible solution requires two. Since the numbers of feasible and infeasible moves is not known beforehand, it is not possible to predict the number of function evaluations taken by C-PSA before the run.

Table 3.8: Parameters used for NSGA-II and IDEA

Parameter	Value
Population Size	200
Number of generations	for CTP2, CTP3, CTP5, CTP7 : 150 for CTP4 : 400 for CTP6 : 200 for CTP8 : 500
Crossover probability	0.9
Crossover distribution index	15
Mutation probability	0.1
Mutation distribution index	20

A comparison of the hypervolume metric obtained using the various algorithms is shown in Table 3.9. It is seen that C-PSA outperforms the other two algorithms in terms of hypervolume for all problems except CTP8 for which the performance of IDEA is marginally better than C-PSA.

Table 3.9: Comparison of hypervolume metric

	C-PSA		NSGA-II		IDEA	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
CTP2	**3.055**	0.001	2.846	0.291	3.008	0.176
CTP3	**2.987**	0.008	2.810	0.250	2.936	0.162
CTP4	**2.847**	0.061	2.497	0.314	2.799	0.143
CTP5	**2.997**	0.019	2.709	0.304	2.940	0.159
CTP6	**36.790**	0.076	36.183	2.187	36.788	0.076
CTP7	**3.617**	0.000	3.212	0.581	3.436	0.594
CTP8	36.150	0.025	34.706	2.904	**36.180**	0.006

The displacement values obtained using various algorithms are listed in Table 3.10. From the table, it is seen that C-PSA is able to outperform the other algorithms for CTP2, CTP5 and CTP7. IDEA obtains the best results for CTP6 and CTP8, whereas NSGA-II obtains the best results for CTP3 and CTP4.

Apart from the competitive average metric values, a commendable feature of C-PSA's performance, as seen from the metrics, is the extremely low values of S.D. This suggests that C-PSA obtains good quality solutions much more

Table 3.10: Comparison of displacement metric

	C-PSA		NSGA-II		IDEA	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
CTP2	**0.0003**	4.53e-5	0.0048	0.0135	0.0032	0.0083
CTP3	0.0084	0.0009	**0.0063**	0.0072	0.0065	0.0071
CTP4	0.0267	0.0030	**0.0126**	0.0091	0.0127	0.0111
CTP5	**0.0008**	0.0001	0.0038	0.0083	0.0034	0.0076
CTP6	0.0010	0.0008	0.0124	0.0628	**0.0008**	0.0007
CTP7	**4.49e-5**	5.60e-6	0.0100	0.0166	0.0063	0.0203
CTP8	0.0004	0.0001	0.0392	0.0865	**0.0002**	8.0e-6

consistently as compared to the other two algorithms.

3.3.2 Greedy v/s non-greedy search

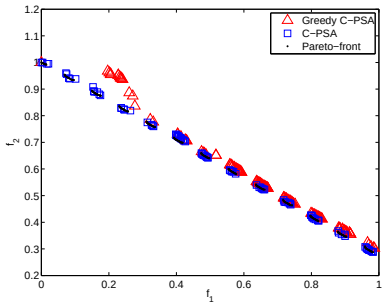
As shown in some previous studies [53], a greedy search doesn't necessarily mean that the search will become trapped in a local optimum. If the step size for perturbation is large enough to escape the local optimum, even a greedy search can converge to a global optimum without any difficulties. This has been demonstrated through studies on various DTLZ test problems in [53]. Thereafter, two test problems, for which it is difficult (or with certain parameter settings, impossible) to escape a local minimum without accepting a worse move, were formulated. As a number of optimization problems may have such characteristics, it is wise to use a technique such as SA whose ability to accept worse solutions makes it sufficiently robust to cope with such problem attributes. Extending this idea to constraint optimization, studies are presented in this section on CTP problems using a *greedy* version of the proposed C-PSA. In this case, greedy applies to both, the objectives and the constraints. The greedy version of the proposed C-PSA is obtained by setting $T, T_G = 0$ in Algorithms 3.3 and 3.4. The probability of a jump from a feasible to an infeasible solution is also set to 0.

Experiments are conducted on problems CTP2-CTP8 using both the greedy

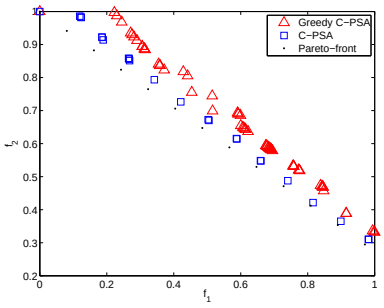
and non-greedy versions of C-PSA. Thirty independent runs are conducted for each problem, using the same parameters as in the previous subsection. A summary of the results is shown in Table 3.11. It can be clearly seen that the greedy C-PSA performs worse than its non-greedy counterpart, as reflected in both the hypervolume and displacement metrics. The greedy algorithm has difficulty escaping from the suboptimal regions, which results in inadequate convergence within given function evaluations. The non-dominated solutions obtained from the median run (based on the displacement metric) using the two algorithms are shown in Figure 3.10. The figures echo the same trends as suggested by the comparison metrics. It is seen that the solutions obtained using the non-greedy approach are consistently closer to the true Pareto fronts as compared to those from the greedy approach.

Table 3.11: Comparison of greedy and non-greedy C-PSA algorithm

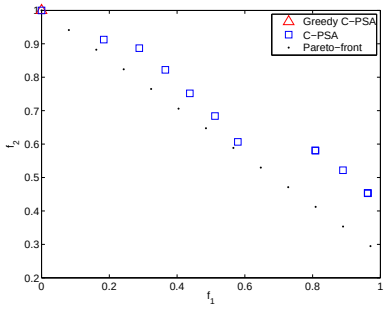
	Hypervolume				Displacement			
	C-PSA		Greedy C-PSA		C-PSA		Greedy C-PSA	
	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
CTP2	**3.0546**	0.0013	2.8501	0.3339	**0.0003**	0.0001	0.0075	0.0053
CTP3	**2.9875**	0.0085	2.7927	0.2958	**0.0084**	0.0009	0.0421	0.0041
CTP4	**2.8468**	0.0607	2.0976	0.2314	**0.0267**	0.0030	0.1731	0.0134
CTP5	**2.9974**	0.0191	2.7464	0.2867	**0.0008**	0.0008	0.0064	0.0055
CTP6	**36.7901**	0.0765	34.1873	5.5596	**0.0010**	0.0009	0.0643	1.8548
CTP7	**3.6167**	0.0004	3.6142	0.0140	**4.49e-5**	5.60e-6	7.93e-5	0.0002
CTP8	**36.1496**	0.0246	31.1527	6.6538	**0.0004**	0.0001	0.1162	2.0766



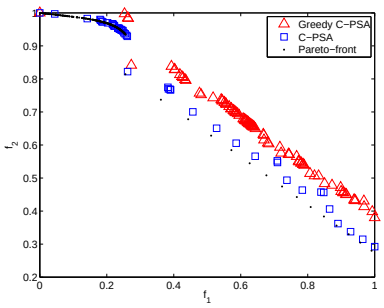
(a) CTP2



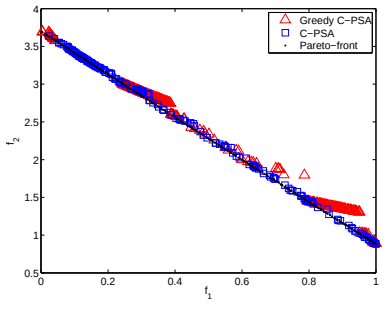
(b) CTP3



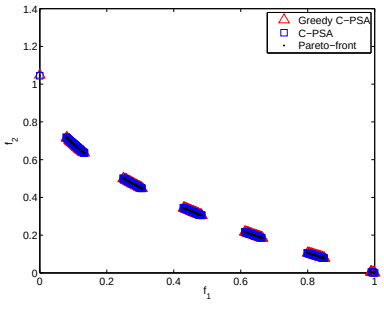
(c) CTP4



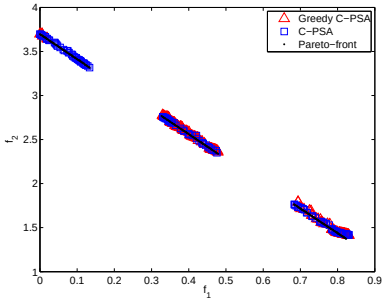
(d) CTP5



(e) CTP6



(f) CTP7



(g) CTP8

Figure 3.10: Comparison of greedy and non-greedy approaches. The plot shows the non-dominated set obtained for a median run based on displacement metric

3.3.3 Discussion of C-PSA parameters

One of the limitations of using a SA is that it involves a number of parameters which have to be appropriately chosen. In the proposed algorithm, the parameters used are T_{max} (calculated based on P_{T_i}), T_G, T_{min}, N, M, HL, SL, P_i, P_f, σ_i, and σ_f. It is often difficult to find the best match for so many parameter values and some reasonable values have to be approximately chosen.

The choice of T_{max} and T_{min} are based on the amount of jump allowed in the algorithm at the beginning and towards the final phases of convergence. Typically, SA is started with a high value for the initial temperature T_{max} in order to explore the design space. A small value for the final temperature T_{min} ensures low acceptance probabilities for unfavorable solutions during the final stages. T_{max} can be set to infinity as done in [50], to accept all trial solutions initially. It can also be determined based on a short "burn in" period as suggested in [64]. In the proposed algorithm, T_{max} is set using the method proposed by Ray *et al.* [69], in which the initial temperature is set by allowing a large perturbation with a high probability P_{T_i}. In that case, the perturbation is calculated based on evaluations of a few random solutions. In the present studies, the range of the energy function (Δdom) is known, *i.e.*, 0 to 1. The temperature T_{max} is calculated by allowing a large perturbation ($\Delta dom = 0.9$) with a high probability $P_{T_i} = 0.9$.

$$T_{max} = -\frac{\Delta dom}{ln(P_{T_i})}$$

The initial value for the temperature T_G (used for the acceptance probability for infeasible $\rightarrow$ infeasible jump), is calculated based on the constraint violations of a few random trial solutions in the same way by replacing Δdom with the maximum constraint violation (CV_{max}), found among those trial solutions. T_{min}

is set to a very low value of 1e-5, which allows almost zero probability of an uphill move towards the termination of the algorithm. The minimum value of T_G is similarly set to 1e-5.

In the proposed algorithm, the parameters P_i and P_f are introduced to calculate the acceptance probability of a jump from a feasible to an infeasible solution. It is desirable that the acceptance probability should monotonically decrease with the temperature. Starting from a relatively high value of P_i (=0.5) initially to accept infeasible solutions, it is exponentially reduced to a low value of P_f (0.01). The jump from a feasible to an infeasible solution is assigned a finite probability of acceptance in order to search the space (including feasible and infeasible) effectively and avoid getting trapped in local optima, especially where the objective space consists of a disconnected feasible space. The benefit derived by probabilistically accepting infeasible solutions is intended to be similar to that obtained by maintaining infeasible solutions to expedite convergence in IDEA [70].

The value of σ determines the spread of perturbation of the Laplacian distribution. It is exponentially reduced from σ_i to σ_f over the iterations. A high value of σ can generate solutions with large jumps that have a better chance of acceptance initially (due to high values of temperature). As the temperature reduces, smaller jumps are favored and a lower value of σ enforces smaller jumps. This is done in an effort to keep up with the probabilities of acceptance *i.e.,* when large jumps have high probabilities of acceptance, the value of σ is high in order to create trial solutions with large steps. This aids the algorithm in doing a *random walk* to identify optimal regions swiftly. When the acceptance of large jumps becomes less probable, the value of σ is reduced, so that shorter jumps, which have more probability of acceptance and of improving the objective values as the

algorithm comes close to convergence, are attempted. In the present studies, the initial value σ_i is set to 1, which implies that the algorithm can traverse the whole range of a chosen variable in a single jump. The value is exponentially reduced to a low final value of $\sigma_f = 0.1$.

The number of iterations N can be either prescribed or calculated based on the prescribed initial temperature, final temperature and decay schedule. The value of M should be chosen to allow for sufficient exploration at a given temperature. In the present studies, N is set. For M, it is recommended that the value be chosen based on the number of design variables. In the present studies, M is calculated as $20 \times n_{var}$, where n_{var} is the number of variables in the problem.

The temperature schedule is kept exponential for simplicity. Other rules, such as logarithmic decrement, also exist in the literature. A more sophisticated temperature schedule is used in [50] in which the decrement rule is adaptive during the algorithm. The decay parameter for C-PSA is calculated based on the initial and final temperatures, and the number of prescribed iterations N.

The radius of exploration r for the ADD method is kept the same as that used in previous studies by Hedar and Fukushima [2]. As far as the number of exploring solutions (p) is concerned, the number of function evaluations required to create a trial solution increases by one for every exploring solution. Thus, this number (p) is conservatively set to 1 so that a move can be made in the ADD with minimum expense (*i.e.* two evaluations).

The hard limit HL and soft limit SL are user-defined choices and depend on the number of non-dominated solutions desired. In this study the value of HL is fixed to the same value as the population size used for NSGA-II and IDEA to ensure a fair comparison. In order to avoid frequent clustering, SL should not be too close to HL. The upper limit on SL is dictated by the available computational

resources, as an increase in archive size will result in higher expenditure for non-domination checks.

3.4 Performance on CEC 2009 Benchmarks

Before concluding this chapter, the performance of IDEA and C-PSA is reported on a set of difficult constrained multi-objective test problems, proposed recently by Zhang *et al.* [71] for the special session and competition in the IEEE Congress on Evolutionary Computation (CEC) 2009. The mathematical formulation of the problems is given in Appendix C. The final report on the performances of the participating algorithms in the competition can be found in [72].

To compare the performance IDEA and C-PSA with these recent state-of-the-art algorithms, the results for seven problems (CF1-CF7) from the CEC 2009 test suite are reported. The evaluation criterion used in the competition is the average inverse generational distance or IGD metric (which is the same as the displacement metric used earlier in the chapter), averaged over thirty independent runs.

For both algorithms, the parameters used are the same as those for the CTP problems reported in previous sections. In accordance with the rules of the CEC 2009 competition, the maximum number of function evaluations is set to 300,000 and the maximum number of solutions used for evaluating the IGD is 100. If the number of solutions in the final non-dominated set obtained for any run is more than 100, the solutions are clustered into 100 clusters, and the solutions closest to the centroids of these clusters are taken as representative solutions from the non-dominated set for evaluating the IGD.

For C-PSA, the number of iterations is set to 1500 so that the algorithm

can reach the prescribed maximum limit of 300000 function evaluations. The number of variables for these problems is $n_{var} = 10$, hence the epoch length M is $20 \times 10 = 200$, which implies that the minimum number of evaluations done by the C-PSA algorithm is $1500 \times 200 = 300000$. However, as two function evaluations are required for generating a trial solution from an infeasible solution, total evaluations reach the prescribed limit (300,000) sooner than the 1500^{th} iteration, at which point the algorithm is terminated. Also, HL is set to 100 which is the same as the maximum number of solutions allowed for calculating the IGD metric. The soft limit SL is set to 150.

The mean IGD values obtained using IDEA and C-PSA are reported in Table 3.12, along with the reported values using other algorithms. It is seen that with the exception of CF4, C-PSA obtains competitive results (albeit not the best) for most problems. The best performance by C-PSA is for the CF1 problem for which it is ranked 2^{nd}. For CF3 and CF7, it shows a median performance (ranked at 4^{th} / 5^{th}), whereas for CF2 and CF5 its performance is second worst. It is worth mentioning here that the proposed C-PSA, unlike other algorithms in the competition, is not a population-based algorithm. To the best of the author's knowledge, there is no other SA-based algorithm whose results are reported on difficult constrained problems such as those presented in this work.

The performance of IDEA is also found to be quite competitive. Again, the best performance is for CF1 for which it is ranked 3^{rd}. For other problems, it is ranked either 5^{th} or 6^{th}. While the implementation of the constraint handling mechanism proposed here is implemented in an EA framework, it is possible to integrate it with other algorithms (such as DE, PSO, etc.) in order to further enhance the performance of multi-objective algorithms for constraint optimization.

Table 3.12: Performance of C-PSA and IDEA on CEC 2009 benchmarks

Rank	CF1		CF2		CF3	
1	LiuLiAlgorithm	0.00085	DMOEADD	0.0021	DMOEADD	0.056305
2	**C-PSA**	0.00342	LiuLiAlgorithm	0.0042	MTS	0.10446
3	**IDEA**	0.00525	MOEADGM	0.008	GDE3	0.127506
4	NSGAIILS	0.00692	NSGAIILS	0.01183	**C-PSA**	0.1443
5	MOEADGM	0.0108	**IDEA**	0.01297	LiuLiAlgorithm	0.182905
6	DMOEADD	0.01131	GDE3	0.01597	**IDEA**	0.193059
7	MTS	0.01918	MTS	0.02677	NSGAIILS	0.23994
8	GDE3	0.0294	**C-PSA**	0.03403	MOEADGM	0.5134
9	DECMOSA-SQP	0.10773	DECMOSA-SQP	0.0946	DECMOSA-SQP	1000000
Rank	CF4		CF5		CF6	
1	DMOEADD	0.00699	DMOEADD	0.01577	LiuLiAlgorithm	0.013948
2	GDE3	0.00799	MTS	0.02077	DMOEADD	0.01502
3	MTS	0.011009	GDE3	0.06799	MTS	0.01616
4	LiuLiAlgorithm	0.01423	LiuLiAlgorithm	0.10973	NSGAIILS	0.02013
5	NSGAIILS	0.01576	NSGAIILS	0.1842	**IDEA**	0.037702
6	**IDEA**	0.061212	**IDEA**	0.239511	**C-PSA**	0.0591969
7	MOEADGM	0.0707	DECMOSA-SQP	0.41275	GDE3	0.06199
8	DECMOSA-SQP	0.15265	**C-PSA**	0.447294	DECMOSA-SQP	0.14782
9	**C-PSA**	5.88743	MOEADGM	0.5446	MOEADGM	0.2071
Rank	CF7					
1	DMOEADD	0.01905				
2	MTS	0.02469				
3	GDE3	0.04169				
4	LiuLiAlgorithm	0.10446				
5	**C-PSA**	0.185705				
6	**IDEA**	0.186596				
7	NSGAIILS	0.23345				
8	DECMOSA-SQP	0.26049				
9	MOEADGM	0.5356				

3.5 Summary

In this chapter, two key contributions are made in terms of constraint handling in optimization. While one is in the paradigm of EA, the other is an enhanced SA. The two contributions are summarized below.

1. **IDEA**: A novel algorithm, IDEA, for constrained optimization problems is proposed. The algorithm maintains infeasible solutions during the evolution, thereby searching the space through both the feasible and the infeasible regions. The original constrained optimization problem is reformulated as an unconstrained optimization problem with one additional objective

which is based on the constraint violation level of the solutions. In addition, the infeasible solutions are ranked higher than the feasible solutions in order to focus the search near the constraint boundary. The search through the infeasible space improves the convergence rate of IDEA over NSGA-II which only searches through the feasible regions, as shown by the experiments on g-series and CTP test problems. Furthermore, IDEA has the additional advantage of providing marginally infeasible solutions for beneficial trade-offs for design considerations.

The faster convergence rate of IDEA also makes it an attractive choice for solving *dynamic* constrained optimization problems, in which the objective and/or constraint functions can vary with time. To validate this, a study on a set of dynamic constrained problems using IDEA is reported in [73]. However, since dynamic problems are not considered in the scope of this thesis, the details are not discussed here.

2. **C-PSA**: SA is extended as C-PSA for constrained MO problems, and tested on CTP test problems. The results obtained using C-PSA are compared with those from NSGA-II and IDEA. C-PSA is found to deliver competitive, and in some cases significantly better, results in terms of displacement and hypervolume metrics. In addition, the consistency of the proposed algorithm is reflected in very low deviations in results across multiple runs. The study substantiates the fact that, in spite of being a single-point method, SA (with a few enhancements) can be used to efficiently solve constrained MO problems. The ability of C-PSA to accept uphill moves makes it less prone to being trapped in a local minimum, thereby delivering solutions close to the true Pareto front more consistently. The advantage of

using a SA-based technique is also emphasized in this work by comparing its performance with that of a greedy search.

Comparisons of the proposed algorithms with some of the most recent multi-objective algorithms on the set of CEC 2009 benchmark problems are also reported. Overall, the proposed algorithms, C-PSA and IDEA show great promise for solving constrained optimization problems.

Chapter 4

Large-scale Optimization I: Large Number of Objectives

Abstract

The existing Multi-objective Optimization (MO) algorithms can solve two- and three-objective problems very efficiently. However, their performances deteriorate rapidly for problems with a higher number of objectives, predominantly because of the failure of Pareto-dominance sorting to induce selection pressure. To deal with such problems, the ranking procedure has to be appropriately modified. Alternatively, some problems can be reduced to fewer objectives, which can be handled using conventional Multi-objective Evolutionary Algorithms (MOEAs). In this chapter, three proposals are made to solve many-objective problems efficiently. The first two are modified secondary ranking procedures for improving convergence and diversity, while the third is a novel way of dimensionality reduction, which requires nominal computational expense compared to existing techniques.

4.1 Overview of many-objective optimization

While conventional MOEAs perform reasonably well for two- and three-objective optimization problems, their performances do not scale well with an increasing number of objectives. This observation has been reported in a number of previous studies [74, 75, 76]. These problems, in which the number of objectives is large, are known as *many-objective* optimization problems. While there is no strict agreement on how many objectives a problem should have in order to be classified as a many-objective problem, it is understood that the term refers to problems which are difficult to solve using the conventional Pareto-dominance methods (typically containing four or more objectives).

Recently, many-objective optimization has attracted significant attention from the Evolutionary Multi-objective Optimization (EMO) community. Many-objective optimization problems are considered far more challenging than traditional two- or three-objective optimization problems, for the following reasons:

1. **Convergence**: The main difficulty arises from the inability of Pareto- dominance based schemes to generate selection pressure to drive the solutions to the Pareto front. For a large number of objectives, most of the solutions in the population become non-dominated very early in the evolutionary search; a characteristic well reported in the literature [74, 75, 76].

   This problem is illustrated in Figure 4.1, which shows the numbers of non-dominated solutions in randomly initialized populations of 100 individuals for scalable test problems DTLZ1-DTLZ4[77]. It is clear from this figure that 70% or more of the solutions of 20 objective DTLZ1-DTLZ4 problems are non-dominated even at the initialization stage. Since all non-dominated solutions have essentially the same rank in terms of con-

vergence, the selection pressure towards the optimum solution is lost if all solutions (or most of them) become non-dominated.

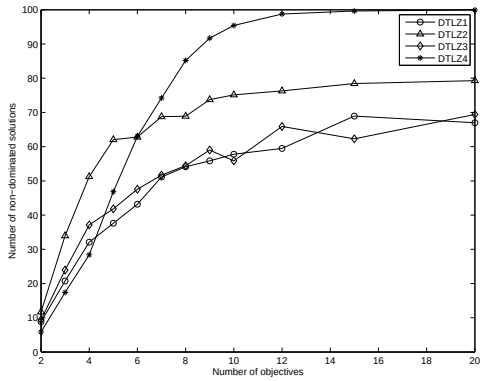


Figure 4.1: Numbers of non-dominated solutions in randomly initialized populations for DTLZ. Population size used is 100.

2. **Coverage**: Many-objective optimization has difficulty associated with coverage of the Pareto front. The number of solutions required to approximate the Pareto front grows exponentially with the number of objectives. This makes it very difficult to capture the whole front for a large number of objectives, even with a reasonably large population size.

3. **Visualization**: There is a problem of visualization of multi-dimensional data. The practicality of solving many-objective optimization is often questioned owing to the fact that, even if the whole Pareto front is available, there are no suitable means for the decision maker to visualize the front, and thus, it is difficult to choose a preferred solution out of the innumerable Pareto-optimal solutions.

Over the last few years, a number of efforts have been made to deal with these issues of convergence, coverage and visualization. Some of them are highlighted below.

1. **Convergence**: To improve convergence, there have been proposals such as average ranking [78], modifying dominance relations [79], indicator-based ranking [80] and substitute distance assignments [76].

2. **Coverage**: On the issue of the coverage of the Pareto front, it is often argued that the decision maker might be interested in focusing on a specific region of interest instead of the whole front. To aid such a preference based search, proposals based on a reference point method [81] and a preference articulation method [82] have been formulated.

3. **Visualization**: For the visualization of solutions, a number of methods, such as Self-organizing maps [83], parallel plots [82], heatmaps [84] and web diagrams [85], have been developed.

In spite of these improvements, the existing methods for dealing with many-objective problems are still not nearly as efficient as they are for two- or three-objective problems. Therefore, another direction that is relevant in the context of many-objective optimization is *dimensionality reduction*. For many problems, it may be possible to solve or analyze solutions to a many-objective problem by reducing the objective set to a much smaller set of *relevant* objectives. The rest of the objectives are innately termed as *redundant* objectives. Identification of such redundant objectives is an actively pursued area by the contemporary EMO community.

In the present work, efforts are made to improve upon two promising proposals for many-objective optimization, namely improvement in secondary ranking and dimensionality reduction:

1. **Secondary ranking procedures**: Conventionally, MO algorithms contain a secondary ranking procedure for promoting diversity among the

non-dominated solutions (e.g. crowding distance ranking in NSGA-II). However, for many-objective optimization, as non-dominance is not sufficient to create convergence pressure, the secondary ranking procedures need to be modified in order to enhance convergence as well. Two new secondary ranking methods are proposed in order to improve both the convergence and diversity of the solutions.

2. **Dimensionality reduction**: The dimensionality reduction techniques in practice currently are often computationally overbearing. A novel method for dimensionality reduction, which estimates the true dimensionality of the optimization problems with relatively low computational expense, is proposed.

The rest of the chapter is organized as follows. In Section 4.2, some of the existing secondary ranking techniques are discussed, followed by the description of the two proposed secondary ranking methods in Section 4.3. Numerical experiments using various secondary ranking methods are presented in Section 4.4. Similarly, the existing and proposed dimensionality reduction methods are discussed in Section 4.5 and Section 4.6 respectively; followed by numerical experiments using various dimensionality reduction techniques in Section 4.7. The findings of these studies are summarized in Section 4.8.

4.2 Existing Secondary Ranking Methods

Pareto-dominance based algorithms often use what is known as a *secondary ranking* technique. A secondary ranking technique is used to differentiate solutions *amongst* those with a same non-dominance rank. In the present studies, NSGA-II is considered as an example. The primary ranking in NSGA-II is the

non-dominated sorting of the solutions. The secondary ranking is called *crowding distance ranking* (given in Chapter 3, Algorithm 3.2), which is intended to give a higher rank to more isolated solutions, thereby preserving and promoting diversity among the solutions. However, while the crowding distance can give a good estimate of diversity for two- or three-objective problems, it is not able to do so for many-objective problems [86]. This is because crowding distance assigns the highest ranks to the solutions corresponding to the extreme values of each objective. For a many-objective problem, since all (or most of) the solutions are non-dominated, the search is driven mainly by the crowding distance which tries to preserve all the extreme solutions. While preserving the extreme solutions may be advantageous for creating a well-spread population for cases when the population is already near the Pareto front, it is not so for the many-objective problems. Consequently, the forced preservation of these extreme solutions may actually hamper the convergence of the algorithm.

To overcome the drawback of selection pressure reduction due to a large number of non-dominated solutions, a number of substitute distance assignment measures have been suggested recently [76]. The idea is to devise measures that would differentiate *amongst* the non-dominated solutions. This is in contrast to NSGA-II, in which all the non-dominated solutions are considered equally good in terms of convergence. Also, even though the focus is on convergence, the secondary distance assignments also try to ensure that diversity is not lost during the run. The following substitute distance assignment measures were suggested in [76]:

1. **Subvector dominance (SV-DOM)**: SV-DOM counts the number of objectives in which a solution is better than other solutions. A solution better in a larger number of objectives is considered a better solution. For

a given solution i in the population, procedure $svd(i, j)$ counts the number of objectives that another solution j is better than i. The values of $svd(i, j)$ are calculated for all $j \neq i$, and the largest among these values is assigned as the distance $dist(i)$ to the solution i. The smaller the value of $dist(i)$, the better is the solution.

2. **-eps-dominance (-ϵ-DOM)**: -ϵ-DOM ranks a given solution based on the smallest amount that should be subtracted from all objectives of the other solutions in order to make them dominate the given solution. For a solution i, the value $mepsd(i, j)$ denotes the smallest amount to be subtracted from all the objectives of solution j, so that it dominates solution i. The smallest such value among all solutions $j \neq i$ is the distance $dist(i)$ assigned to the solution i. A larger $dist(i)$ value implies a better solution, as it means that a large amount has to be subtracted from the objectives of the other solutions to make them dominate the given solution.

3. **Fuzzy Pareto dominance (FPD)**: FPD assignment is based on the Fuzzy Pareto dominance relationship as proposed in [87]. For a solution i in the population, the product of the bounded quotients $F(i).m/F(j).m$ of all objectives is calculated for all $j \neq i$ (where $F(i).m$ represents the m^{th} objective value for the i^{th} solution). The largest value of the product among all other solutions (such that $j \neq i$) is assigned to i as the distance value. A smaller $dist(i)$ value implies a better solution.

4. **Sub-objective dominance count (SOD-CNT)**: SOD-CNT ranks the solutions based on two separate rankings: SV-DOM and -ϵ-DOM. For each solution i in the non-dominated set, a set S_i is constructed as all pairs of two single criterion distance measures: $M - svd(i, j)$ (where M is the number

of objectives), and $mepsd(i, j)$, for all $j \neq i$. The number of solutions in the Pareto set PSO_i of the set S_i is assigned as $dist(i)$ to the solution. A larger $dist(i)$ value implies a better solution.

A comparative study of these substitute distance assignments along with the crowding distance was reported in [76]. It was observed that SV-DOM and SOD-CNT have the best convergence properties, but attain poor diversity. On the other hand, FPD has the worst convergence among the substitute distance assignments (still better than the crowding distance), but the diversity obtained among the solutions is good. The -ϵ-DOM was found to be a good compromise of both convergence and diversity. It is to be noted that, unlike in conventional practice, the diversity of solutions was compared in the variable space (for the P* problem, described in Section 4.4) whereas the diversity is usually desired in the objective space. In spite of its good performance, the -ϵ-DOM has a limitation, which is described later in this chapter.

4.3 Proposed Secondary Ranking Methods

4.3.1 Cluster-sort

The first secondary ranking procedure in this chapter is referred to as Cluster-sort. This method can be used as an alternate scheme for preserving diversity. The proposed method doesn't show any bias to the corner points, unlike crowding distance. The preference for corner solutions may be disadvantageous for many-objective problems, as a large number of solutions in the front may be "extreme" solutions in one or more of the objectives. In such cases, the distance assigned using the crowding distance to each of those points will be infinity

and hence, there will be no preferred point amongst them. The drawback of maintaining corner solutions has also been highlighted in [86].

The Cluster-sort is outlined in Algorithm 4.1. The non-dominated front F to be ranked is first clustered into half the number of points it contains, i.e. $|F|/2$. (if $|F|$ is odd, then ceiling($|F|/2$) is used). Then within each cluster, a point closest to its centroid is identified. The solutions closest to the centroids are assigned a lower rank than the rest of the solutions in F. The procedure is shown in Algorithm 4.1. For the present studies, *hierarchical clustering* is used. The motivation behind clustering into $|F|/2$ points is that during the evolution, when the (parent+child) population is largely non-dominated (rank 1), solutions with good diversity could be chosen as the parent population for the next generation.

Algorithm 4.1 Cluster-sort

Require: Front data {non-dominated rank, corresponding set F}
1: **for** $i = 1$ to $|F|$ **do**
2: Assign $dist(i)$ = non-dominated rank + 0.5.
3: **end for**
4: $A = cluster(F \rightarrow |F|/2)$ {cluster set F into $|F|/2$ points}
5: $C \leftarrow$ Centroids(A)
6: **for** $i = 1$ to $|F|/2$ **do**
7: $id = min(norm(x_{id \in C(i)} - C(i)))$ {Find id of closest point to i^{th} cluster}
8: Assign $dist(id)$ = non-dominated rank. {The points closest to centroids get smaller $dist$}
9: **end for**
10: {Smaller $dist \Rightarrow$ Better solution.}

4.3.2 Modified-ϵ-DOM

The second method proposed in this chapter focuses on improving convergence as well as diversity. A close look at the -ϵ-DOM ranking (shown in Algorithm 4.2) reveals that if two solutions in a non-dominated set are located very close to each other, there is a good chance that both of them will get depleted in the next

generation. For a given solution i, -ϵ-DOM ranking tries to identify a solution $j \neq i$ in the population, such that $mepsd(i, j)$ is the minimum. Thereafter, this value is assigned as $dist(i)$. In other words, point j is used by point i for distance assignment. The drawback of assigning the distances this way is illustrated in Table 4.1 using a sample population containing 12 solutions. A plot of the solutions is shown in Figure 4.2. Solutions 1 and 2 (which are almost overlapping in the figure), receive very low *dist* values because the *dist* to Solution 1 is assigned using Solution 2 and vice-versa. As a result, both solutions receive low ranks, and will most likely be deleted in the next generation. Ideally, if a diverse set of points were to be chosen for the next generation, it should contain at least one of these points (solution 1 or 2), which does not happen if the conventional -ϵ-DOM is used.

To overcome this limitation of -ϵ-DOM, the following strategy is proposed. While assigning the *dist* value to any solution A, solution B which will determine the *dist* value for it is identified. Then, it is checked if solution A was used to assign *dist* value to B. If not, then *dist* is assigned to A using solution B. Otherwise, the *dist* value is calculated based on the set of points excluding B. The same process is repeated until a solution is found which has not been used to assign *dist* value to A. A pseudo-code outlining this process is shown in Algorithm 4.3.

This method is expected to perform better than the original -ϵ-DOM for the following two reasons:

1. During the search, -ϵ-DOM can loose two (or more) solutions if they are very close in objective space, even if both (or all) of them are very good solutions in terms of convergence. The modified -ϵ-DOM ensures that at least one of the solutions among them obtains a good rank and is carried

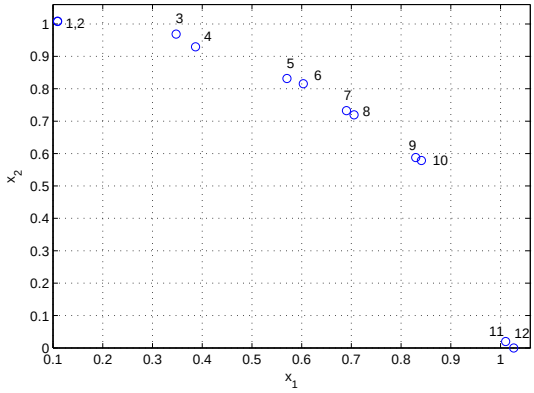


Figure 4.2: Sample population of 12 non-dominated points

Table 4.1: Ranking of a sample population of 12 non-dominated points

Sol. id	f_1	f_2	conv ($\|f\|-1$)	-ϵ-DOM *dist*	-ϵ-DOM sol. used	-ϵ-DOM rank	Mod-ϵ-DOM *dist*	Mod-ϵ-DOM sol. used	Mod-ϵ-DOM rank
1	0.1090	1.0086	0.0145	0.0001	2	12	0.0001	2	12
2	0.1091	1.0075	0.0134	0.0011	1	11	0.2384	3	2
3	0.3475	0.9686	0.0290	0.0389	2	2	0.0391	4	7
4	0.3866	0.9295	0.0067	0.0391	3	1	0.0780	2	6
5	0.5705	0.8316	0.0085	0.0329	6	3	0.0329	6	8
6	0.6034	0.8156	0.0145	0.0160	5	6	0.0868	7	5
7	0.6902	0.7323	0.0063	0.0153	8	7	0.0153	8	10
8	0.7055	0.7193	0.0075	0.0130	7	8	0.0963	6	4
9	0.8295	0.5876	0.0165	0.0113	10	9	0.0113	10	11
10	0.8408	0.5786	0.0206	0.0090	9	10	0.1407	8	3
11	1.0103	0.0197	0.0105	0.0163	12	5	0.0163	12	9
12	1.0266	0	0.0266	0.0197	11	4	0.5786	10	1

on to succeeding generations to generate even better solutions.

2. From the perspective of diversity, if two points are close in the objective space, it is desirable that at least one of them is carried forward to the next generation, so that diverse solutions on the Pareto front can be captured. The modified -ϵ-DOM ensures this by eliminating the assignments of *mutual* distance values (*i.e.* if solution A is used to assign *dist* to solution B, then B will not be considered for A).

Table 4.1 shows the difference introduced by eliminating the mutual ranking.

Algorithm 4.2 -eps-dominance assignment (-ϵ-DOM)

Require: F {Non-dominated set}
  $N_s = |F|$ {Number of solutions in the non-dominated set}
  M = Number of objectives
  Define $mepsd(i, j)$
  $max = 0$
  for $m = 1$ to M **do**
    $max = max(F[j].m - F[i].m, max)$
  end for
  return max
  for $i = 1$ to N_s **do**
    $F[i].dist = \infty$
    for all $j \neq i$ **do**
      $v = mepsd(i, j)$
      if $F[i].dist > v$ then $F[i].dist = v$
    end for
  end for
  Larger $dist \Rightarrow$ Higher rank

It can be seen that the solutions very close to each other no longer receive similar distances or ranks; e.g., since solution 1 uses solution 2 for its *dist* value, solution 2 uses the next best candidate, i.e. solution 3 for its *dist* assignment. As a result, solution 1 gets a poor rank, but solution 2 gets a good rank. Hence, one representative solution is preserved to be carried over into the next generation. On the other hand, in the conventional -ϵ-DOM, solution 1 uses solution 2 and vice versa and in the process, both points are "killed". From the convergence point of view, it can be seen that points 1 and 2 have good convergence values of 0.0145 and 0.0134 respectively, which are the 7^{th} and 6^{th} best values respectively among the 12 solutions. Using Mod-ϵ-DOM ranking, it is seen that solution 2 receives a good rank, which is also desirable for the convergence. At the same time, it can be observed that solution 12 obtains a better rank than solution 11 which is disadvantageous in terms of convergence. However, the important thing to note here is that, since both solutions are close, even by carrying solution

Algorithm 4.3 Modified-eps-DOM assignment (Mod-ϵ-DOM)

Require: F {Non-dominated set}
  $N_s = |F|$ {Number of solutions in the non-dominated set}
  M = Number of objectives
  for all $i = 1, 2, \ldots N_s$ **do**
    Set $F[i].used = 0$ {$F[i].used$ denotes id used by Solution i to calculate $F[i].dist$}
  end for
  for $i = 1$ to N_s **do**
    $F[i].dist = \infty$
    for all $j \neq i$ **do**
      if $F[j].used \neq i$ **then**
        $v = mepsd(i, j)$
        if $F[i].dist > v$ then $F[i].dist = v$
        $F[i].used = j$
      end if
    end for
  end for
  Larger $dist \Rightarrow$ Higher rank

12 into further generations, using crossover and mutation, solutions that are at par or better than solution 11 will be generated. However, if the conventional -ϵ-DOM is used, both solution 11 and 12 will receive very low rank owing to their proximity in the objective space.

4.4 Numerical Experiments (Secondary Ranking)

Following the description of the proposed secondary ranking procedures in the previous sections, the numerical experiments to study their performance on many-objective test problems are detailed in this section.

4.4.1 Test problems studied

To demonstrate the performance of the proposed secondary ranking procedures, two sets of scalable problems (P* problems and DTLZ problems) are chosen:

P* problems

For most problems, as the number of objectives grows, it becomes increasingly difficult to visually analyze the quality of the obtained non-dominated set. Koppen and Yoshida [76] designed a set of problems for which the Pareto front is easy to visualize, even for a high number of objectives. These problems are referred to as P* problems. The problem definition is as follows.

Given a set of m fixed points $(P_1, P_2, ..., P_m)$ in Euclidean place, the objective values $f_1, f_2, ..., f_m$ at a given point x_i are $d(x_i, P_1)$, $d(x_i, P_2)$,..., $d(x_i, P_m)$ respectively, where $d(A, B)$ denotes the Euclidean distance between the two points A and B. The aim is to minimize all the objectives $f_1, f_2, ..., f_m$.

The Pareto set (solutions in the variable space) of the problem is given by the convex enclosure of the points P_i. The proof and a detailed discussion can be found in [85]. In the present work, P* problems are studied in order to *qualitatively* illustrate the performance of the proposed secondary ranking methods. Thereafter, more rigorous numerical studies are conducted with DTLZ test problems.

DTLZ problems

DTLZ is a set of multi-objective test problems [77] which are scalable in terms of the number of objectives and the number of variables. Problems DTLZ2 and DTLZ3 with up to 30 objectives are used for the present study. They are chosen from the DTLZ suite because the distance of any solution from the Pareto front

for these two problems can be calculated as $\|f\|$-1, which can be used as an unambiguous measure of convergence. The definitions of DTLZ2 and DTLZ3 problems are given in Appendix D.1 and D.2. Performances of different secondary ranking methods are compared using various performance metrics as described in the following subsections.

4.4.2 Performance metrics

Since the optimum solution to a MO problem consists of not one solution but a set of solutions, performance measures are required to evaluate the quality of the obtained non-dominated set. Performance metrics can be broadly classified into two categories: (a) metrics for convergence and (b) metrics for diversity. A number of metrics to measure one or both of these qualities have been proposed in the literature [88, 17]. Many of them essentially compare the obtained non-dominated set with a true or representative Pareto front. Convergence is usually measured by calculating distance of the obtained front from the reference set. Diversity is usually measured by comparing the distribution of the non-dominated set with that of the reference set. However, when the Pareto front is not known a priori, the choice or generation of a reference set becomes a problem in itself. One of the suggestions in such a case [88] is to merge all the non-dominated solutions from all the generations of EA and find the reference set by non-dominated sorting of the combined pool. However, if the algorithm has not converged sufficiently, the reference set so obtained will be far away from the Pareto front and comparing the non-dominated set with this reference set may give misleading results. This problem is more pronounced for many-objective problems since the performance of MO algorithms is known to deteriorate as the number of objectives increase. Test problems DTLZ2 and DTLZ3 are chosen

for the present studies as their Pareto front is known and the convergence of a solution can be evaluated unambiguously.

Convergence metric

The problems DTLZ2 and DTLZ3 have a spherical Pareto front, given by $\|f\| = 1$, where $\|f\| = \sum_{i=1}^{M} f_i,\ f_i > 0 \forall i$, M denotes the number of objectives, and f_i denotes the i^{th} objective value corresponding to the solution. The distance of any solution from the Pareto front can be found using ($\|f\|$-1). This value is computed for all solutions in the population and the mean value is then used as a measure for convergence.

Diversity metrics

In addition to convergence to the Pareto front, it is desirable that the obtained set possesses good diversity, *i.e.*, the non-dominated set is adequately and evenly spread in the objective space. A number of diversity metrics exist in the literature. Again, many of them require a reference set which is difficult to obtain, especially for many-objective problems. Three diversity metrics that do not use a reference set are considered here. These metrics measure "pure" diversity, *i.e.*, they measure sparseness of points in the set disregarding the range of the set or closeness to the Pareto front. When examined in isolation, these diversity measures alone may not be of great significance as a non-dominated set far away from the Pareto front may have a much bigger spread than a set near to it. However, combined with convergence metrics, they can provide a reasonable measure of the overall quality of the non-dominated set. A brief description of the three diversity metrics follows.

1. **Standard deviation of the crowding distances**: The Standard De-

viation (S.D.) of the crowding distance values of solutions in the final population can give an indication of how regularly the points are spaced. The idea was proposed in [89]. The same metric is used here, but with a slight modification. The S.D. will be large if the non-dominated set is spread over a larger volume (even though it may not be a better set) as compared to a set that is spread over a smaller volume. Therefore, the S.D. is normalized using the mean crowding distance of the population. Hence, the metric used is given by Equation 4.1

$$SDC = \left(\sqrt{\frac{1}{|F|} \sum_{i=1}^{|F|} (d - \bar{d})^2} \right) / \bar{d} \tag{4.1}$$

The lower the value of the metric, the better the solutions are distributed in the objective space.

2. **Grid-count (D2) diversity metric**: This metric was proposed by Deb and Jain [88]. The obtained non-dominated points for an M objective problem are projected to an $(M-1)$ dimensional plane. Thereafter, a uniform grid is generated in the projected plane, and the number of occupied grids is counted; the higher the count, the better the distribution of the points. The recommended value for the number of grids for each objective is suggested as $G_i = N^{1/M-1}$, where N is the population size.

3. **Sammon mapping-based diversity measure**: In addition to the above metrics, another metric that can be used to visualize the distribution of points for multi-objective problems, is proposed. Sammon mapping is a data projection technique for reducing the dimensionality of data so that the relative topology of the points is preserved. A detailed discussion of

Sammon mapping can be found in [90]. The proposed metric is calculated as follows.

(a) Firstly, the M-dimensional data is projected via Sammon mapping to a 2D space. As Sammon mapping preserves the distance relations among the multidimensional data, the distribution of the projected data D on a 2D plane closely resembles the distribution of the points in the objective space.

(b) The bounds of the data, as well as their orientation along any particular axes, are disregarded during the mapping. To take care of the bounds, the 2D data is scaled using the ratio between the maximum distance among the points in the objective and the 2D spaces. Hence, if r_1 is the maximum Euclidean distance between the data points in objective space and s_1 is the corresponding distance in the projected space, then the scaled data values are calculated as $D_1 = D * (r_1/s_1)$.

(c) Principle component analysis (PCA) of the scaled 2D data D_1 is performed to take care of the orientation of the mapped data.

(d) Thereafter, the 2D plane within the bounds of the data is divided into a regular grid and the number of grids occupied by the transformed points in the 2D plane is counted. The higher the value of this count, the better the distribution of the points. In the present studies, a 10×10 grid is used, resulting in 100 boxes.

The calculation of this proposed diversity metric for a 5-dimensional sample data containing 100 points is illustrated in Figure 4.3.

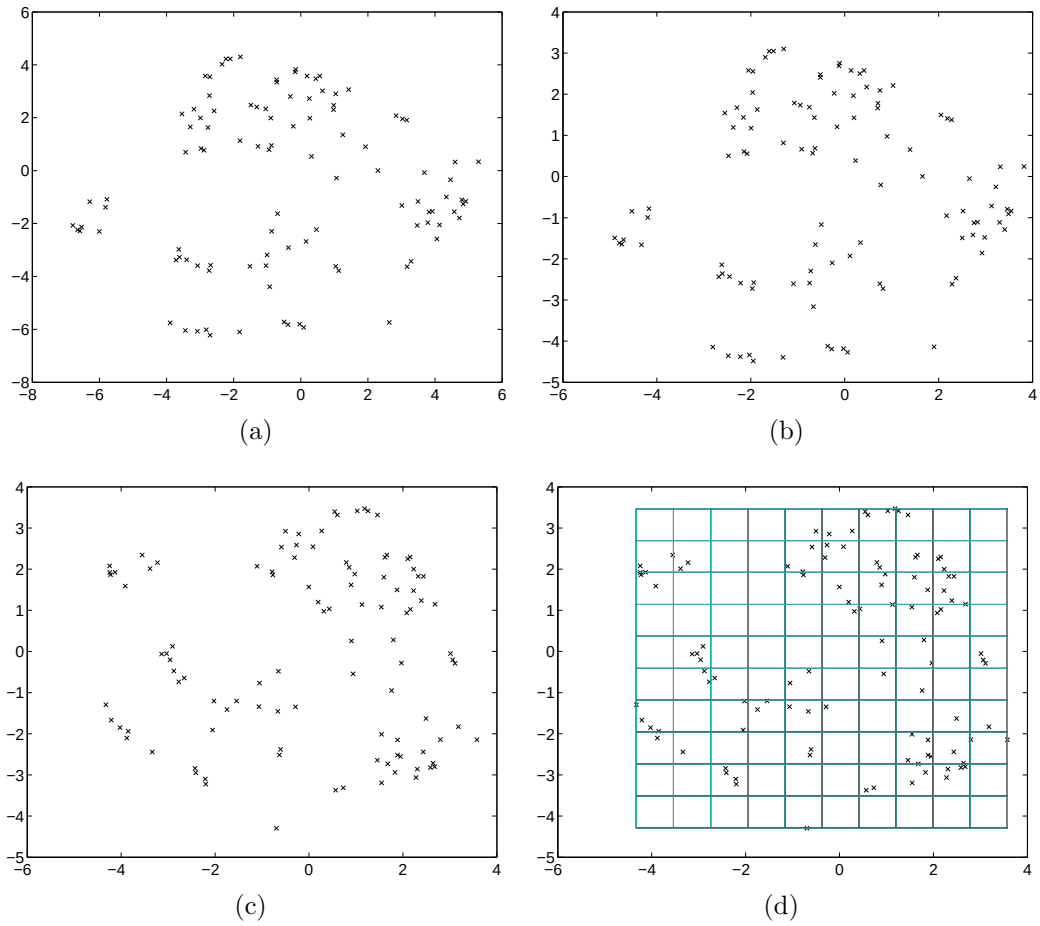


Figure 4.3: Calculation of Sammon mapping based diversity metric: (a) 5D data projected in 2D using Sammon mapping; (b) 2D data scaled using ratio of maximum Euclidean distance in objective space and mapped data; (c) scaled 2D data after PCA analysis; and (d) counting of occupied grids

4.4.3 Experimental setup and results

P* problems

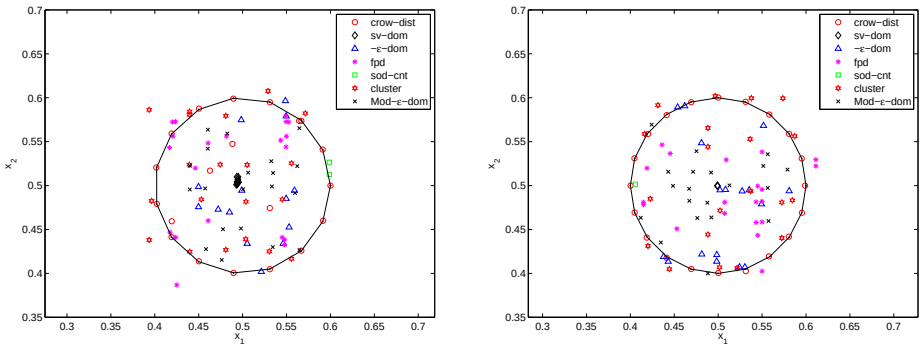
For P* problems, the fixed points used to calculate the objectives are chosen as the corners of an M-sided regular polygon in the variable (2D) space, where M is the number of objectives (see Figure 4.4). The parameters used are: probability of crossover $= 0.9$, probability of mutation $= 0.1$, crossover index $= 15$, and mutation

index = 20. A population of 20 solutions is evolved over 100 generations. The results are shown for 15 and 20 objective P* problems.

As previously discussed, the Pareto front for a P* problem consists of all the solutions bounded by the polygon. This implies that the more solutions an algorithm obtains lying inside the polygon, the better its convergence. Also, the more evenly the solutions are distributed inside the polygon, the better the diversity among them.

The final populations from a typical run using each of the 7 strategies are shown in Figure 4.4. It is observed that all the solutions obtained using SV-DOM and SOD-CNT converge near to one point inside the polygon, indicating a good convergence but poor diversity. Crowding distance, due to its preference for extreme points, obtains solutions in the periphery of the polygon (also observed in [76]). Cluster-sort obtains a good distribution of solutions inside the polygon indicating good diversity, but poor convergence as indicated by the presence of a number of solutions outside the polygon. Mod-ϵ-DOM and -ϵ-DOM show the best trade-off between convergence and diversity, with convergence of Mod-ϵ-DOM observed to be better than that of -ϵ-DOM. The performance of FPD is also close to that of -ϵ-DOM.

From the preceding discussion, it is observed that Mod-ϵ-DOM is able to improve upon the existing -ϵ-DOM ranking method. Cluster-sort mechanism is able to obtain good diversity among the solutions but, since it is also merely a mechanism for preserving diversity, no benefit is derived in terms of convergence. Therefore, it would be more suitable to use this scheme in conjunction with other fast convergence schemes (with poor diversity) such as SV-DOM and SOD-CNT. Further studies are conducted for 20-objective P* problem by using a combination of Cluster-sort with SV-DOM and SOD-CNT. The parameters used are the same



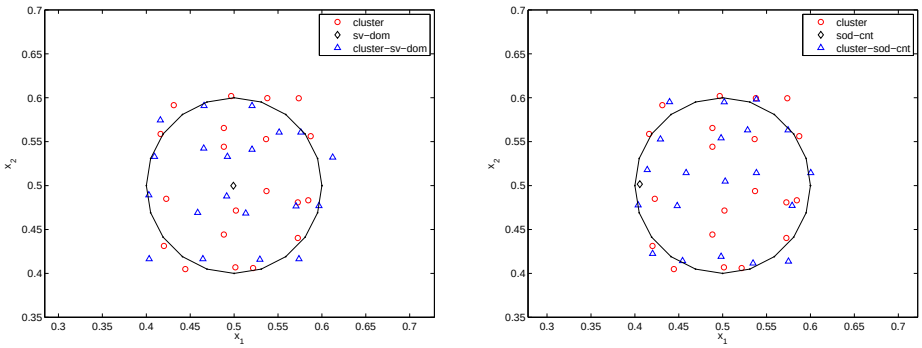
(a) 15-objectives (b) 20-objectives

Figure 4.4: Final populations obtained for P* problems (in variable space) using various secondary ranking methods with NSGA-II

as those used previously in this section. Results obtained using SV-DOM for the first 40 generations and Cluster-sort thereafter (up to 100 generations) are shown in Figure 4.5(a). Figure 4.5(b) shows the results obtained using SOD-CNT for the first 40 generations and Cluster-sort thereafter. Also shown are the results obtained using the schemes individually. It can be seen that, if SV-DOM or SOD-CNT alone are used, the solutions show poor diversity. On the other hand, if Cluster-sort alone is used, the convergence is not good, as reflected in the presence of a number of solutions outside the 20-sided polygon. However, when the schemes are used in combination, the solutions have better convergence (fewer solutions outside the polygon). At the same time, the solutions show more even distribution, indicating better diversity.

DTLZ problems

For DTLZ2 and DTLZ3, experiments are undertaken for 5-, 10-, 15-, 20-, 25- and 30-objective problems. Twenty independent runs are performed using the 7 different secondary ranking assignments, *viz.*, crowding distance, Cluster-sort, -ϵ-DOM, Mod-ϵ-DOM, FPD, SV-DOM and SOD-CNT, and their results are



(a) (b)

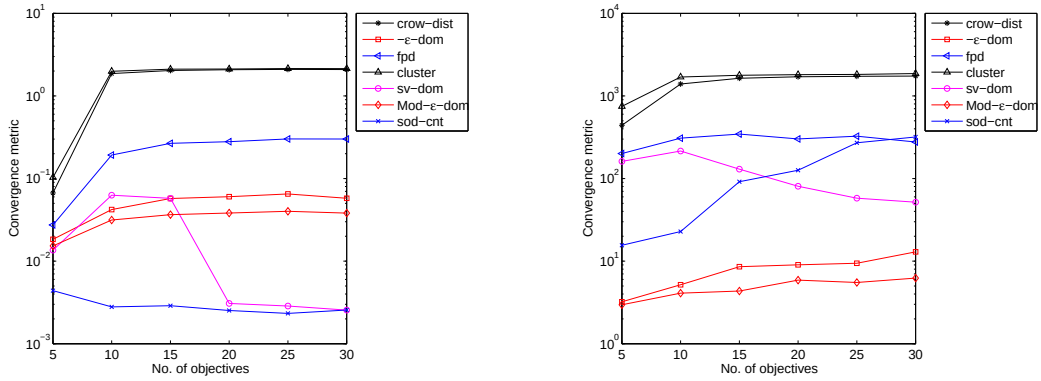
Figure 4.5: Distributions of final populations obtained for 20 objective P* problem (in variable space) using combination of schemes: (a) SV-DOM with Cluster-sort; and (b) SOD-CNT with Cluster-sort.

compared in terms of convergence and diversity of the obtained non-dominated sets. The crossover and mutation parameters are kept the same for each run and across all strategies, and are listed in Table 4.2. A population size of 100 is evolved over 200 generations. The same number of function evaluations are used for all strategies to ensure a fair comparison.

Table 4.2: Crossover and mutation parameters

Parameter	Values
Crossover probability	1.0
Mutation probability	$1/n, n$ = number of variables
Crossover distribution index	15
Mutation distribution index	20

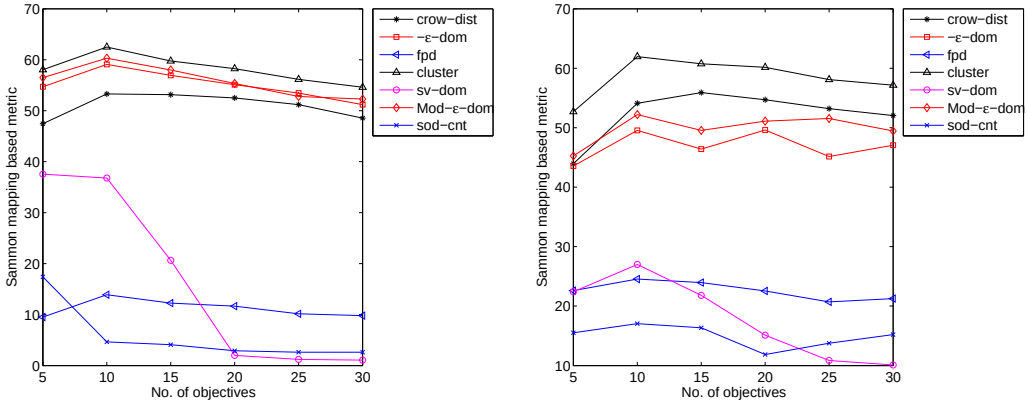
Convergence values obtained using various substitute distance assignments are shown in Figure 4.6. For DTLZ2, it is seen that SOD-CNT obtains the best convergence values except in the 30-objective case, for which SV-DOM has a marginally better value. However, both SV-DOM and SOD-CNT have very poor diversity values, as evident from Figures 4.7- 4.9. Cluster-sort, crowding distance and FPD have the poorest convergence values among all the assignments.



(a) DTLZ2 (b) DTLZ3

Figure 4.6: Convergence metrics averaged over 20 runs

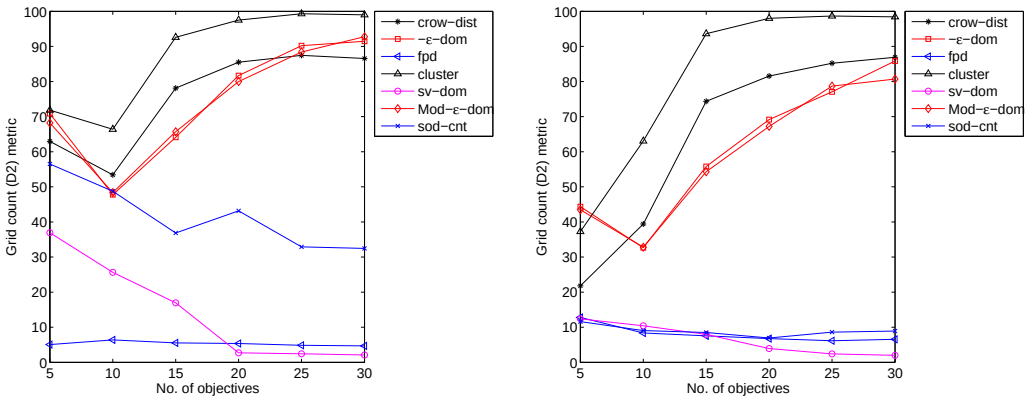
However, while Cluster-sort and crowding distance perform the best in terms of diversity, FPD performs poorly in diversity as well. -ϵ-DOM and Mod-ϵ-DOM perform well in terms of both convergence and diversity. The convergence of these two assignments are not as good as that of SOD-CNT, but they achieve much more diverse sets of non-dominated points compared to SOD-CNT which tends to obtain solutions in a very concentrated region. Mod-ϵ-dom shows better convergence values than -ϵ-DOM.



(a) DTLZ2 (b) DTLZ3

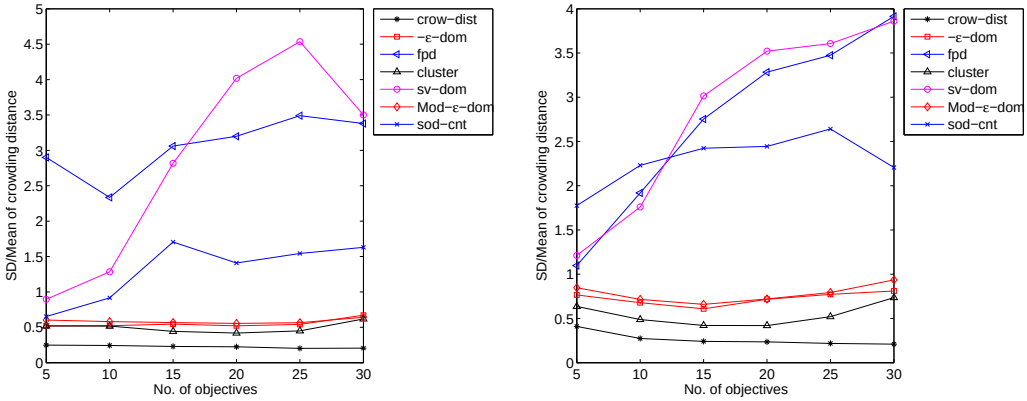
Figure 4.7: Diversity metrics based on Sammon mapping, averaged over 20 runs

DTLZ3, which is a more difficult problem to solve compared to DTLZ2, puts



(a) DTLZ2 (b) DTLZ3

Figure 4.8: Grid count (D2) diversity metrics, averaged over 20 runs



(a) DTLZ2 (b) DTLZ3

Figure 4.9: Diversity metrics SDC (SD/mean of crowding distance), averaged over 20 runs

the convergence ability of the various algorithms to a tougher test. From Figure 4.6, it can be seen that -ϵ-DOM and Mod-ϵ-DOM comprehensively outperform all other assignments in terms of convergence. Among these two, Mod-ϵ-DOM shows better values than -ϵ-DOM. Also, the difference in their convergence values becomes greater with increasing numbers of objectives. In terms of diversity, crowding distance and Cluster-sort give the best results, but they have very poor convergence. Mod-ϵ-DOM and -ϵ-DOM have comparable performances in terms of all the three diversity metrics considered. The convergence values of

SV-DOM show an interesting property. As opposed to the other assignments in which convergence values tend to become worse as the number of objectives is increased, SV-DOM shows an improvement in performance as the number of objectives is increased. This is because it assigns the *dist* value based on the number of objectives for which a solution is better than others. Hence, the *dist* value can take at the most M unique values, where M is the number of objectives. Therefore, as the number of objectives is increased, the probability of the solutions in the population to get different *dist* value increases, thereby creating a selection pressure for better solutions. However, when the number of objectives is few, a number of solutions are likely to get the same *dist*, thereby losing the selection pressure. But in spite of good convergence, SV-DOM loses diversity, because unlike -ϵ-DOM, it does not have an implicit diversity-preserving mechanism.

Individually, the performance of each secondary ranking assignment is as follows:

1. -ϵ-DOM shows extremely good convergence values, especially for DTLZ3. At the same time, it also maintains relatively good diversity, which makes it an attractive ranking process for many-objective problems.

2. Mod-ϵ-DOM performs even better than -ϵ-DOM in terms of convergence. Elimination of the mutual ranking helps it to preserve good solutions irrespective of closeness to other points. At the same time, it helps to maintain good diversity by retaining at least one representative point from a "group" of points.

3. Crowding distance shows very good values in terms of all the diversity metrics but has the worst convergence. This is expected, as the crowding

distance operator is designed to improve the diversity, taking into consideration only the sparseness of points, not convergence.

4. Cluster-sort also shows good values of the diversity metrics. It does not have a preference for the extreme solutions unlike crowding distance. However, the convergence is still poor, and therefore it is recommended that it be used in conjunction with other schemes that have good convergence.

5. The performance of FPD is found to vary across the experiments. In terms of convergence, it is better than Cluster-sort and crowding distance, but worse than the other schemes. It is able to obtain better diversity than SV-DOM and SOD-CNT for most, but not all, test cases. The diversity obtained using FPD is poorer than that of the other schemes (except SV-DOM and SOD-CNT) for all test cases.

6. SV-DOM obtains good convergence values, especially for higher objectives, but fails to maintain diversity among the population members.

7. SOD-CNT shows also good values of convergence, but it deteriorates rapidly as the number of objectives is increased. For lower objective problems, its convergence is better than SV-DOM but, for higher objective problems it is worse. However, like SV-DOM, SOD-CNT also shows very poor diversity.

This concludes the discussion on the secondary ranking procedures for many-objective optimization problems. For a number of many-objective problems, significant benefits can also be derived by the reduction of the objectives to fewer *relevant* objectives (dimensionality reduction); which is discussed in next three sections.

4.5 Existing Dimensionality Reduction Methods

Although many of the studies described above show great promise, in general, MOEAs need to improve a lot before they can be deemed nearly as efficient for many-objective optimization problems as they are for two- and three-objective problems. Given the limitation on the availability of computational resources, it is prudent to first investigate whether the many-objective problem *is*, in fact, a many-objective problem. More often than not, the problem at hand may have many objectives, but they may be reducible to fewer *relevant* objectives which can be easily handled by existing MOEAs. The *relevant* objectives, in the context of the presented studies, are the ones which are sufficient to generate the Pareto front of the many-objective optimization problem. The rest of the objectives, which are not necessary to obtain the Pareto front, are innately termed *redundant* objectives. As pointed out in earlier studies by Saxena *et al.* [91], the redundancy of objectives can result from two scenarios: (a) the objectives are of a non-conflicting nature; or (b) the objectives are conflicting, but their removal from the problem makes a statistically insignificant difference to the Pareto front.

Dimensionality reduction has been a widely researched topic in data analysis; e.g., in image and signal processing, the raw data can often have very high dimensionality. A number of schemes for transforming high-dimensional data into lower-dimensional data have been suggested in the literature [92]. By finding a subset of features that adequately represent high-dimensional data, these techniques aid in data compression, visualization and classification. A comparative review of such dimensionality reduction procedures is reported by Maaten *et al.* [92]. In many-objective optimization, dimensionality refers to the number of objec-

tives in the problem, and the term "dimensionality reduction" refers to finding a (potential) subset of the original set of objectives which can represent the optimization problem adequately. While the aim of reducing dimensionality remains the same for both cases (*i.e.* data analysis data and many-objective optimization), the difference arises due to the involvement of Pareto-dominance in the case of many-objective optimization. As the Pareto-dominance needs to be taken into account for dimensionality analysis, the standard dimensionality reduction techniques cannot be directly applied to many-objective optimization problems [93]. However, many concepts from data analysis can still be useful for identifying a relevant set of objectives in many-objective optimization.

The early seminal work discussing the issue of redundancy of objectives is by Gal and Leberling [94], in the context of linear optimization problems. Another work, also dealing with linear problems relating to Multi-criteria Decision Making (MCDM) is by Agrell [95]. More extensive studies in the domain, especially in the context of evolutionary optimization, have taken place only in the last decade or so. Some recent significant contributions in the field of dimensionality reduction in many-objective optimization are summarized below.

4.5.1 Correlation-based objective reduction

One line of thought for dimensionality reduction relies on information regarding correlations among various objectives. PCA-based reduction proposed by Saxena and Deb [96] is among the early works using correlation. In their proposal (referred as PCA-NSGA-II), a representative set of solutions for dimensionality analysis is obtained by running NSGA-II for a large number of generations. Thereafter, the correlation matrix $\mathbf{R}$ (w.r.t. objectives) is computed using the objective values of the final population. The eigenvalues and corresponding

eigenvectors are then analyzed in order to reduce the objectives. The procedure starts with the original set of objectives which are eliminated iteratively based on their contribution to the principal components and their degree of conflict with other objectives. Once the set of objectives cannot be reduced further, the procedure is stopped and the reduced set of objectives is declared.

Further studies revealed that the linear PCA technique has the drawback of misinterpreting the data when it lies on sub-manifolds. To mitigate this problem, techniques based on non-linear correlations were proposed by Saxena and Deb [97]. While the reduction procedure remains largely the same, the key difference lies in the way in which the correlations were calculated. Instead of the linear PCA, two new proposals were put forward: Correntropy PCA (C-PCA), and Maximum variance unfolding (MVU). The corresponding algorithms were named C-PCA-NSGA-II and MCU-PCA-NSGA-II respectively. Using these non-linear techniques, more accurate results were obtained for up to 50-objective DTLZ-(2,M) problems [96]. Even so, the algorithms were found to be ineffective for a number of other problems (such as DTLZ5-(5,10) and DTLZ5-(5,20) [96]), since the population for dimensionality analysis was not converged near enough to the Pareto front for a meaningful analysis. To overcome this problem, Saxena *et al.* [91] suggested the use of two different algorithms in place of NSGA-II, namely NSGA-II (ϵ-dom) [76] and Infeasibility driven evolutionary algorithm (IDEA) [98]. The representative sets of non-dominated solutions obtained using these two algorithms were found to be much closer to the Pareto front as compared with those obtained using NSGA-II, and DTLZ5-(5,M) test problems with up to 30 objectives were solved accurately. A concept of dimensionality reduction was also similarly extended to reducing the number of *constraints* by Saxena and Deb [99, 100].

4.5.2 Dominance structure-based reduction

Another salient direction in dimensionality reduction for optimization problems has been explored by Brockhoff and Zitzler [101, 102, 103]. In their studies, they investigated how adding and omitting an objective affects problem characteristics, and discuss formal definitions of conflict and redundancy of objective sets. Subsequently, a quantification (δ) for measuring the change in the dominance structure of a problem based on ϵ-dominance was introduced. Thereafter, two following problems related to dimensionality reduction were identified:

1. δ-`MOSS` problem: If the original set of objectives is denoted by $\mathcal{F}$, a δ-`MOSS` problem involves finding a minimum subset $\mathcal{F}' \subset \mathcal{F}$ given an error δ.

2. `k-EMOSS` problem: If the original objective set $\mathcal{F}$ has k objectives, a `k-EMOSS` problem involves finding a subset $\mathcal{F}'$ containing $\mathtt{k} < k$ objectives, with the smallest possible δ error.

An exact algorithm for solving the above problems was proposed by the authors of [101]–[103]. However, the time complexity of the exact algorithm limits its practical usage. Therefore, a few heuristic techniques, with relatively quick run-times but no guarantee of achieving the exact solutions, were also suggested.

In their approach, the dimensionality reduction is achieved by solving one of the two problems defined above using a set of representative Pareto solutions. A population obtained from the Indicator Based Evolutionary Algorithm (IBEA) [80] is used as a representative Pareto front. The dimensionality reduction makes the decision-making process easier for the user by providing a reduced set of objectives, without significant loss of information regarding Pareto-dominance. This *offline* technique was further extended in their later works to address *online*

dimensionality reduction by combining them with hypervolume based EAs [104, 105]. To this effect, three kinds of dimensionality reduction schemes were embedded into the baseline algorithm Simple IBEA (SIBEA) [106]: a) the random selection of a reduced set of objectives, b) the selection of reduced objectives based on `k-EMOSS`, and c) the selection of reduced objectives based on δ-`MOSS`. In their experiments, in which a fixed run-time was allotted to each algorithm, the algorithms which try to use reduced dimensionality are able to outperform the baseline SIBEA in terms of the hypervolume indicator, for DTLZ2_{BZ} and DTLZ7 test problems with up to 9 objectives.

4.5.3 Feature based selection

Recently, Jaimes, Coello and Chakraborty [107] developed a dimensionality reduction scheme based on an unsupervised feature selection technique by Mitra *et al.* [108]. In their approach, the objective set is first divided into homogeneous neighborhoods based on a correlation matrix of a non-dominated set obtained using an EA. Conflict between the objectives takes the role of distance, meaning that the more conflict between the objectives, the more distant they are in the objective "conflict" space. Thereafter, the most compact neighborhood is chosen and all the objectives in it, except the center one, are dropped as they are the least conflicting. Similar to previously mentioned studies, two algorithms were developed. The first finds the minimum set of objectives with the minimum error possible whereas the second finds a subset of objectives of a given size with minimum error possible. The algorithm is claimed to be computationally less expensive than the earlier proposed algorithms [102]. This dimensionality reduction scheme was later integrated into a MOEA by Jaimes, Coello and Barrientos [109] to form the Reduction Genetic Algorithm (RedGA). Two different schemes for

integration were studied. In the first, the objectives are successively reduced every few generations of the MOEA and, towards the end of the search, all objectives are considered again. In the second scheme, both the reduced and original sets of objectives are considered alternately every few generations during the search. The results obtained using these reduction techniques, combined with NSGA-II, are found to outperform the baseline NSGA-II for DTLZ2_{BZ} test problems with up to 10 objectives.

At this juncture, it is important to point out a key difference between the approaches mentioned in subsection 4.5.1 versus those mentioned in subsections 4.5.2 and 4.5.3. While all the approaches do emphasize the importance of reducing objectives, they try to solve two fundamentally different kinds of problems:

1. Problem 1 (approaches given in subsection (4.5.1)): Given a problem with a set of k objectives $\mathcal{F}$, is there a subset $\mathcal{F}' \subset \mathcal{F}$ of size $\mathtt{k} < k$ sufficient to solve the problem instead? If so, which $\mathtt{k}$ objectives form the subset $\mathcal{F}'$? This question can also be posed as: what is the *dimensionality* of the Pareto front? The approaches mentioned in subsection 4.5.1 try to find a reduced *relevant* set of objectives in order to expedite the search in lieu of using the complete set of objectives.

2. Problem 2 (approaches given in subsections 4.5.2 and 4.5.3): Given a Pareto front obtained for a problem with a set of k objectives $\mathcal{F}$, is it possible to analyze the solutions with respect to an objective set $\mathcal{F}' \subset \mathcal{F}$ of size $\mathtt{k} < k$, without losing much information regarding the dominance structure? If so, which $\mathtt{k}$ objectives form the subset $\mathcal{F}$? The primary aim of such approaches is to aid in the decision making process by using fewer objectives.

   Extensions to the *online* dimensionality reduction address the problem

of reducing objectives during the search, but are still not equivalent to solving Problem 1, as the search is not for a *relevant* subset(s) of objectives, optimizing which alone will give the solution to the original problem.

As a simple example, consider the scalable DTLZ2 test problem proposed by Deb *et al.* [77]. If Problem 1 is attempted (as in [96, 97]) for an M-dimensional DTLZ2, then the result should indicate that *all* M objectives are relevant, and the problem should be solved by considering all objectives.

On the other hand, Problem 2 was solved in [103, 102], where it was indicated that it is possible to *analyze* the set of solutions with respect to a smaller subset of original objectives while preserving the dominance structure (*e.g.* use of 18 objectives instead of the original 25 for the 25-objective DTLZ2). However, in such a case, it should be noted that the reduced set of identified objectives is *with respect to* the set of solutions being analyzed. Identifying 18 objectives (with $\delta = 0$) instead of 25 indicates that solutions analyzed do not cover the complete Pareto front (since the true Pareto front lies in the 25-objective space).

In this work, an attempt is made to solve Problem 1, i.e. to identify the dimensionality of the Pareto front. A new approach to identify a reduced set of objectives (if it exists), whose optimization alone should be sufficient to provide the solution to the original optimization problem, is proposed. In this approach, the true dimensionality of a many-objective optimization problem is identified with a relatively low computational cost by searching for a specific set of non-dominated solutions on the Pareto front for dimensionality analysis instead of an approximation of the *whole* Pareto front. Earlier studies in this field have emphasized that both – *convergence* and *coverage* are difficult to achieve for any many-objective optimization problem using non-dominance based techniques. Therefore, an alternate ranking scheme that provides solutions on the boundaries

of the Pareto front is used in the proposed approach. These solutions lying on the Pareto front are diverse in terms of the range of their objective values and should aid in the accurate estimation of the true dimensionality of the Pareto front.

The proposed approach, in which the dimensionality of a Pareto front is identified using a specialized set of solutions, is discussed in detail in the next section.

4.6 Pareto Corner Search for Dimensionality Reduction

4.6.1 Motivation

All dimensionality reduction schemes mentioned in the previous section essentially have two major components: (a) the generation of an approximation of the Pareto front using an MOEA, and (b) the dimensionality analysis of the obtained approximation of the Pareto front. These components may be applied either once or repetitively to come up with a reduced set of objectives. The approaches used in the studies mentioned in Section 4.5 for these two components are summarized in Table 4.3.

Table 4.3: Summary of previous works in dimensionality reduction

Reference work	Pareto front approximation	Dimensionality reduction	Focus of work
Saxena and Deb ([96, 97])	NSGA-II	PCA (linear/non-linear)	Problem 1
Saxena *et al.* ([91])	IDEA, NSGA-II+ϵ-dominance	PCA (non-linear)	Problem 1
Brockhoff and Zitzler ([102, 103])	IBEA	δ-`MOSS`, `k-MOSS`	Problem 2
Jaimes *et al.* ([107, 109])	NSGA-II	Feature selection	Problem 2

The methods used for Pareto front approximations in the literature suffer

from (at least one of) the following drawbacks.

1. If a conventional MOEA (*e.g.* NSGA-II) is used to generate a representative Pareto front, it usually has to be run for a large number of generations, thereby resulting in huge numbers of function evaluations; e.g., in the previous studies on DTLZ5-(I, M) problems ([96, 97]), the number of evaluations used for generating an approximate Pareto front run in millions for every iteration of dimensionality reduction. Such large numbers of function evaluations may not be practical, especially if each function evaluation is expensive. More importantly, even after running for so long, the population may still be far away from the Pareto front, which might render extraction of any information (with respect to dimensionality reduction) meaningless. Although the studies on DTLZ5-(I, M) problem by Saxena *et al.* [91] emphasize this problem and also partly overcome it by using better convergence techniques, the number of evaluations used is still substantially large (500,000 per iteration).

2. On the other hand, if an indicator-based algorithm based on hypervolume is used to generate the Pareto front, the number of evaluations required to get close to the Pareto front is less, but the time for the hypervolume calculation itself grows significantly with the number of objectives. This makes it impractical to use for problems with more than 10 objectives, as mentioned in the studies by Brockhoff and Zitzler [104]. However, some recent studies [110] have used a faster method of hypervolume estimation based on Monte-Carlo sampling which makes it viable to study problems with higher numbers of objectives using hypervolume.

3. Another impediment in using a conventional MOEA to generate an ap-

proximation to the Pareto front is that it seeks an approximation to the *entire* Pareto front which becomes impossible for high numbers of objectives even with a large population size; as the number of solutions required to approximate the Pareto front grows exponentially with the number of objectives.

Using the current state-of-the-art MOEAs, it is difficult to get a representative set of solutions that approximates the entire Pareto front. Therefore, it is worthwhile exploring whether a smaller set of solutions that capture the dimensionality information of the Pareto front can be found. If so, these solutions should have the following properties:

- Convergence: The solutions should be sufficiently close to the Pareto front to capture the dimensionality correctly.
- Diversity: The solutions should have a good spread to ensure that variations in all objective values are captured.

In the proposed approach, it is hypothesized that if sufficient solutions are found on the boundaries of the Pareto front, the dimensionality of the Pareto front can be predicted using those solutions. The search is focused on a few *key* solutions on the boundaries of the Pareto front. These solutions are termed the "corner" solutions of the Pareto front where the boundaries intersect.

Definition 3. For an M-dimensional minimization problem, an *ideal point* is the point in the M-objective space corresponding to the minimum value of each objective.

$$P_{ideal} = (f_1^*, f_2^*, \ldots, f_M^*)$$

Definition 4. Consider an optimization problem with M objectives in which the objective set is denoted by F_M. Now, consider a subset F_k of the original objectives set, with $k < M$ objectives. If minimizing the k-objectives in this subset results in a single solution in the M-objective space, this solution is referred to as a *corner* (or Pareto corner) solution.

For a bi-objective optimization problem, it is straightforward to visualize the corner solutions of the Pareto front. The corner solutions correspond to the minimum value of each objective, as shown in Figure 4.10. However, for a generic M-objective problem, k can take values from 1 to M-1, and hence there are 2^M -1 possible corners to an M-objective optimization problem. This number corresponds to all possible sets of objectives which, when minimized, result in a single solution. The case in which all objectives when simultaneously minimized result in a single solution is omitted, because then the problem will reduce to a single-objective optimization problem.

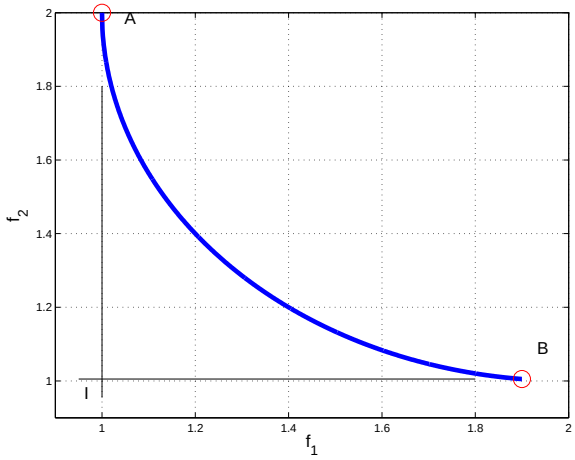
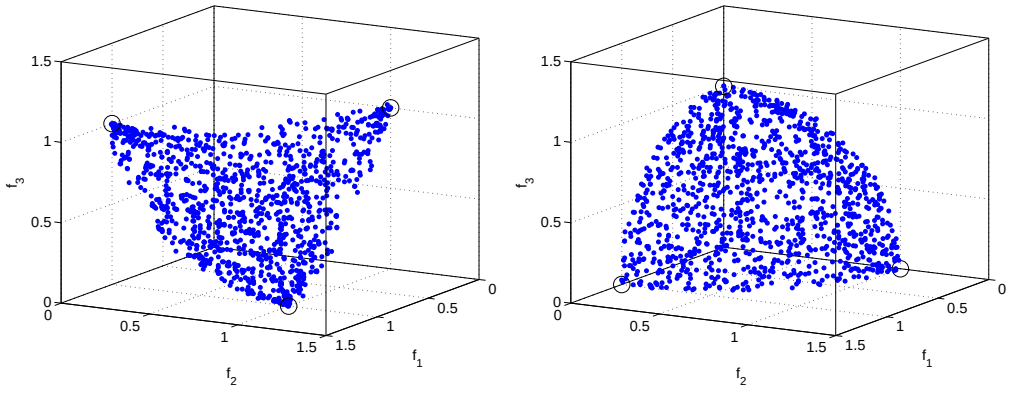


Figure 4.10: Pareto corners of a bi-objective problem (A and B are the corners, I is the ideal point)

Out of all possible 2^M-1 sets, two extreme cases can be identified as follows:

- Case 1: The corner solutions lie on a hyperplane where only one of the objectives f_m is minimum ($f_m = f_m^*$). An example of which is shown in Figure 4.11(a), in which it can be seen that minimization of each objective individually gives a single solution lying on the plane $f_m = f_m^*$ (the corners of the Pareto front are shown by circles).

- Case 2 : The corner solutions lie on an objective axis with origin at the *ideal point*. This happens in a case in which $M-1$ objectives are simultaneously minimized to their minimum value ($f_i = f_i^*,\ \forall i \neq m$). A simple example is shown in Figure 4.11(b). For the corner solution lying on the f_1 axis, both f_2 and f_3 are at their minimum value, and so on.



(a) Case 1 (b) Case 2

Figure 4.11: Different types of Pareto corner solutions (corners shown using circles)

Figure 4.12 shows a case in which the Pareto front has both of the above described cases of corner solutions, with one of the corner points lying on the f_3 axis, whereas the other is on the plane $f_3 = f_3^*$.

As previously mentioned, the number of possible corner solutions increases exponentially with the number of objectives (M). If all these corner solutions are to be explicitly searched, the search effort required will become impractical as M

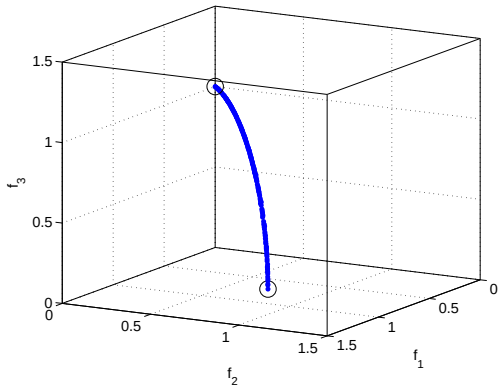


Figure 4.12: Combination of two different types of Pareto corner solutions (corners shown using circles)

grows. However, in reality the actual number of corner solutions of the Pareto front is often much less; e.g., the Pareto fronts for the widely studied scalable test problems DTLZ [77] and WFG [111] have a maximum of M corner solutions, for M-objective problems.

The proposed approach comprises two steps– a) finding the corner solutions, and (b) dimensionality reduction using the corner solutions. Pareto corner search evolutionary algorithm (PCSEA) is proposed to search for the corner solutions. As PCSEA does not use dominance-based ranking, it does not suffer from the limitation of convergence pressure loss while handling a large number of objectives. As the evolutionary search in PCSEA is focused on finding only a few solutions on the Pareto front, it can converge quite close to the corner solutions within a reasonable number of evaluations. In the dimensionality reduction step, the final population of PCSEA is used to determine whether any of the objectives are redundant. Although the proposed dimensionality reduction is unable to predict the exact dimensionality in some cases (as discussed later), it gives a good estimate for a number of problems shown in the following section.

4.6.2 Pareto Corner Search Evolutionary Algorithm (PCSEA)

PCSEA is designed to search for the corners of the Pareto front. However, not all possible $2^M - 1$ corners are explicitly searched for. In PCSEA, the solutions which minimize either one of the objectives or the rest of the objectives simultaneously are preferred and ranked higher. The minimization of $M-1$ objectives is achieved using a composite scalar function, the L_2 norm of these objectives. Any other norm that tries to simultaneously minimize these objectives could also be used instead.

The idea behind minimizing $M-1$ objectives using the L_2 norm is that, for the problems for which it is possible to minimize $k < M-1$ objectives simultaneously, it is likely that minimizing all $M-1$ objectives will give a solution close to such a corner, provided all values of the objectives are of approximately the same order of magnitude[1]. Thus, it might be possible to search for the corner solutions corresponding to all sets of $k < M-1$ objectives in a single pass. Also, it should be kept in mind that minimizing the L_2 norm will minimize the objective values only if the objective values are positive. If the objectives can assume negative values, a constant quantity can be added to each of them in order to ensure they take only positive values.

For example, consider the objective functions of the single-variable problem shown in Figure 4.13. Of the six objectives, four (f_2, f_4, f_5, f_6) have minima at $x = 0.5$, making this point a corner solution as per the considered definition. Minimization of the L_2 norm of $M-1$ objectives (excluding f_3) gives the solution 0.535 which is quite close to the actual corner solution, as shown in the figure.

[1]However, no scaling has been used in the presented studies in order to convert the objective values to the same order. The objective values have been used as is.

Although this may not always be the case, it is likely to be for problems with large numbers of objectives, many of which are redundant (which can be minimized simultaneously).

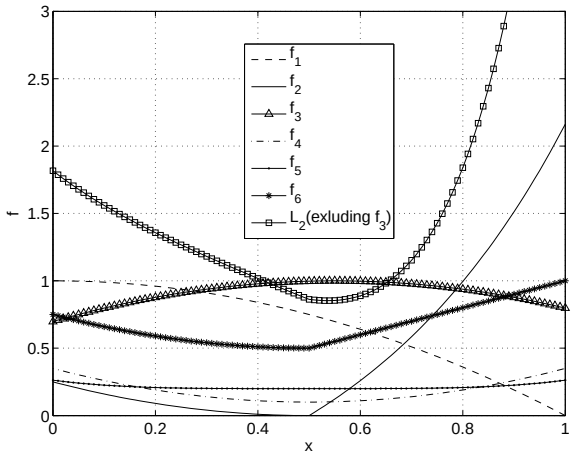


Figure 4.13: Minimization of $M-1$ objectives (excluding f_3) gives a solution close to the corner where f_2, f_4, f_5, f_6 are minimum. The problem considered here has a single variable, shown on x-axis.

PCSEA is outlined in Algorithm 4.4. Like NSGA-II, it uses SBX crossover and polynomial mutation operators to create offspring solutions. However, unlike NSGA-II, which uses non-dominated sorting and crowding distance based ranking, PCSEA uses the corner-sort ranking procedure in which the solutions are ranked based on the individual objective values and L_2 norms of all-but-one objectives.

The corner-sort ranking procedure is described below. Firstly, the solutions are sorted based on the increasing individual objective values and increasing all-but-one objective values (converted to a single composite value using the L_2 norm).

1. Sort the solutions in increasing order of the objective values f_i, $i = 1, \ldots, M$. For M objectives, this corresponds to M sorted lists.

Algorithm 4.4 Pareto Corner Search Evolutionary Algorithm (PCSEA)

Require: N {Population size}
Require: $N_G > 1$ {Number of generations}
1: Initialize(pop_1)
2: Evaluate(pop_1)
3: **for** $i = 2$ to N_G **do**
4: $childpop_i$ = Evolve(pop_{i-1})
5: Evaluate($childpop_i$)
6: $S \leftarrow$ corner-sort(pop_{i-1}+$childpop_i$)
7: $pop_i = S(1 : N)$
8: **end for**

2. Sort the solutions in increasing order based on the values of the rest of the objectives except the current one. For more than two objectives ($M > 2$) a single composite value is calculated using the L_2 norm of the rest of the objectives ($\sum_{j=1,j\neq i}^{M} f_j^2$ where $i = 1, \ldots, M$). For M objectives, this corresponds to M sorted lists.

This gives a total of $2M$ sorted lists. The solutions are picked up from each sorted list in turn and are assigned ranks based on the order in which they are picked up. This way, the corner solutions are promoted over generations in order to eventually capture the corners of the Pareto front.

The corner-sort ranking procedure is illustrated using a sample population of 12 solutions for a 3-objective minimization problem (Table 4.4). The solutions are sorted using the single and all-but-one objectives (L_2 norm) and each of the $2 \times 3 = 6$ sorted lists are shown in Table 4.5. Each column shows the solution IDs corresponding to the sorting criteria. To assign the overall ranks, the solutions are picked in turn from each of the sorted lists.

The first rank is assigned to the first solution in the first sorted list. Thus, the solution ID 5 is picked as the rank 1 solution. The second rank is assigned to the first solution in the second sorted list, so the solution ID 3 is picked as rank

Table 4.4: Sample population for a 3-objective problem

Solution ID	f_1	f_2	f_3
1	0.0617	0.1561	0.1173
2	0.3924	0.5660	0.0336
3	0.5446	0.0183	0.4089
4	0.6359	0.2619	0.0731
5	0.0365	0.7365	0.6474
6	0.2322	0.4008	0.0357
7	0.2440	0.3225	0.1113
8	0.6014	0.0876	0.1886
9	0.9205	0.1960	0.1153
10	0.7453	0.0277	0.2315
11	0.1123	0.0914	0.3017
12	0.0551	0.5851	0.7805

Table 4.5: Corner-sort ranking (columns represent solution IDs sorted using objective values in Table 4.4)

Rank	f_1	f_2	f_3	$f_2^2 + f_3^2$	$f_1^2 + f_3^2$	$f_1^2 + f_2^2$
1	(5)	(3)	(2)	(1)	1	(11)
2	12	10	6	8	(6)	1
3	1	8	4	9	7	7
4	11	11	7	10	11	6
5	6	1	9	4	2	3
6	7	9	1	11	8	12
7	2	4	8	7	4	8
8	3	7	10	6	5	4
9	8	6	11	3	3	2
10	4	2	3	2	10	5
11	10	12	5	12	12	10
12	9	5	12	5	9	9

2 solution. The process is repeated, spanning all columns, and the corresponding solutions are assigned ranks in the increasing order. The solutions selected in the first pass are circled in Table 4.5. If a solution has already been picked, then the next solution in the same column is picked instead. In the example, as the solution ID 1 is already selected using $f_2^2 + f_3^2$ criterion, it is skipped and the next solution, ID 6 is selected for $f_1^2 + f_3^2$ criterion. If there are multiple copies of a solution in the population, the ranking is done based on the unique solutions. The time complexity of the ranking procedure can be calculated as

follows: each list can be sorted in $O(Nlog(N))$ time, where N is the number of solutions. Since there are $2M$ lists to be sorted, the complexity of ranking the solutions is $O(2MNlog(N))$.

In addition to the corner solutions, PCSEA may also identify additional solutions on the Pareto front boundaries. This is because the L_2 norm is used to combine minimizations of all-but-one objectives together. For exact corner points, true minimization of the objective set has to be carried out, but this is impractical for many-objective problems. These solutions may affect the dimensionality analysis in some cases, as described later.

4.6.3 Dimensionality reduction

Once the corner solutions of the Pareto front are identified using PCSEA, the dimensionality analysis of the obtained solutions is performed using a heuristic technique proposed in this section. While conducting the dimensionality analysis, it is assumed that the population obtained using PCSEA is converged close to the (corners of the) Pareto front, so as to appropriately represent the dependency of the Pareto front on various objectives. If this is the case, the Pareto-dominance among the solutions will be largely due to the *relevant* objectives. Therefore, if one of the redundant objectives is omitted from the set, there will be no (or negligible) change in the number of non-dominated solutions in the population. On the other hand, if a relevant objective is omitted, the number of non-dominated solutions will be significantly affected. Building on this idea, the reduction process is done as follows.

Firstly, a representative set F is formed using the non-dominated solutions present in the final population obtained using PCSEA. If multiple copies of a solution are present in the population, they are removed and only unique solutions

are considered[2]. This set F is used for the dimensionality reduction procedure. To quantify change in the number of non-dominated solutions, a parameter R is defined as follows:

$$R = N_{F_R - f_m}/N_F,$$

where

$$\begin{aligned} N_F &= \text{Number of non-dominated solutions} \\ &\quad \text{in the reference set } F \\ N_{F_R - f_m} &= \text{Number of non-dominated solutions} \\ &\quad \text{corresponding to the objective set obtained after} \\ &\quad \text{omitting } f_m \text{ from the set } F_R \end{aligned} \tag{4.2}$$

where F_R denotes the set of relevant objectives. If the value of R is very high for an objective f_m, this means that omitting that particular objective does not have a significant impact on the number of non-dominated solutions, and therefore it can be omitted from the set F_R.

The set of relevant objectives F_R is initialized as the original objective set $(f_1, f_2 \ldots f_M)$. Thereafter, starting from the first objective (f_1), each objective (f_m) is omitted in turn, and changes in the number of non-dominated solutions are observed. If the quantity R for an objective f_m exceeds a user defined threshold C, then the objective is deemed redundant and is removed from F_R. This process is repeated until all the objectives have been considered. Thereafter, F_R is declared as the final reduced objective set.

[2]For identifying unique solutions, a tolerance of 0.01 for the objectives is used consistently for the examples presented in this chapter. However, it is recommended that while dealing with objectives of different orders of magnitudes, the objectives should either be normalized before using a constant tolerance, or the tolerance value for each objective should be computed individually based on its range.

The result of dimensionality reduction is not entirely independent of the order in which the objectives are omitted. Depending on the sequence of the objectives considered, F_R may contain different objectives. The redundancy of an objective is a result of the non-conflict between the objective under consideration and one (or more) other objective(s). Therefore, any of the non-conflicting objectives can be included in the final set. It is essential to maintain at least one of these non-conflicting objectives in order to maintain the dimensionality of the Pareto front. Thus, even though each objective can be evaluated independently, not all the redundant objectives can be dropped simultaneously. Therefore, the objectives are dropped sequentially after considering the conflict between various objectives at every step.

4.7 Numerical Experiments (Dimensionality Reduction)

In this section, numerical experiments are conducted on a number of test cases to study the performance of Pareto corner search for dimensionality reduction.

4.7.1 Test cases

To test the performance of the proposed approach for dimensionality reduction, three different test problems with various numbers of objectives are studied. The test functions are DTLZ2, DTLZ5-(I, M) and WFG. The first two of these test functions have also been studied extensively in the literature [97, 96, 91] for dimensionality reduction.

DTLZ2

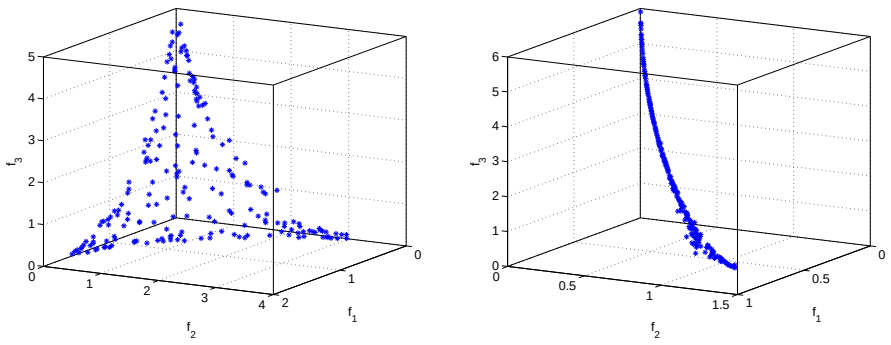
In order to build the basis for studying the performances of various MOEAs, a scalable test problem suite DTLZ was formulated by Deb *et al.* [77]. DTLZ2 is one of the test functions from the suite. The formulation of DTLZ2 can be found in Appendix D.1. The notation DTLZ2(M) is used to refer to an M-dimensional DTLZ2 problem.

DTLZ5-(I,M)

Another function from the DTLZ test suite, DTLZ5 was later modified to construct a set of test problems in which the dimensionality of the Pareto front is less than the original number of objectives [96]. These test problems are known as DTLZ5-(I, M) problems. Here, I denotes the actual dimensionality of the Pareto front and M is the original number of objectives for the problem. The intent of formulating these problems was to critically evaluate dimensionality reduction techniques for many-objective optimization problems. The formulation of DTLZ5-(I, M) is given in Appendix D.3.

WFG

Both problems considered in the above subsections have concave fronts. To demonstrate the performance of the proposed approach for problems with convex fronts, walking fish group (WFG) problems are chosen. WFG is a flexible toolkit for construction of scalable test problems developed by Huband *et al.* [111]. While constructing a problem using this toolkit, various attributes of the test problem can be controlled, including continuity, shape, dimensionality and modality of the Pareto front.



(a) $\text{WFG3}_{\text{conv1}}$ (b) $\text{WFG3}_{\text{conv2}}$

Figure 4.14: Approximations of Pareto fronts for $\text{WFG3}_{\text{conv1}}$ and $\text{WFG3}_{\text{conv2}}$ problems (generated by evolving 200 solutions for 200 generations using NSGA-II)

To study the performance of the proposed approach, two three-objective problems are formulated here. Both the problems are similar to the problem WFG3 defined in [111], but instead of linear shape function, the convex shape function is used ($h_{m=1:M} = \text{convex}_m$). In addition, for the first problem, $A_1 = 1, A_2 = 1$ is used, resulting in a *non-degenerate* Pareto front (one whose dimensionality is less than the number of objectives) whereas, for the second problem, $A_1 = 1, A_2 = 0$ is used, resulting in a degenerate Pareto front (dimensionality reduced by one). For convenience, these problems are referred to as $\text{WFG3}_{\text{conv1}}$ and $\text{WFG3}_{\text{conv2}}$. The approximate Pareto fronts for these problems are shown in Figure 4.14. The details regarding the construction of problems using the toolkit can be found in [111].

4.7.2 Experimental setup

The crossover and mutation parameters for PCSEA used in the numerical experiments are listed in Table 4.6. The experiments are performed on a number of values of M for each problem. A summary of the experiments and the

corresponding numbers of evaluations (population size × number of generations) are shown in Table 4.7. For the dimensionality reduction procedure described in subsection 4.6.3, the threshold value C is chosen as 0.8, implying that if $R \geq 0.8$ for a particular objective f_m, then f_m is considered to be redundant, and is discarded from the relevant set of objectives F_R. The experiments are repeated thirty times in order to establish the repeatability of performances.

Table 4.6: Parameters used for PCSEA

Parameter	Value
Crossover probability	0.9
Crossover distribution index	10
Mutation probability	0.1
Mutation distribution index	20

Table 4.7: Summary of numerical experiments

Test Problem	Number of objectives (M)	Evaluations used (pop size × generations)
DTLZ2	3	100× 200
	5,10	200 × 500
	15,20	200 × 1000
DTLZ5-(5, M)	10,20,30, ... 100	200 × 200
WFG	3	200 × 200

4.7.3 Results

The results obtained from experiments on the various test cases are summarized as follows.

DTLZ2

The final population for three-objective DTLZ2 obtained using PCSEA is shown in Figure 4.15. It can be seen that the whole population has converged to the

corners of the Pareto front of DTLZ2. The dimensionality analysis is carried out by omitting each objective in turn (see Table 4.8), as according to the procedure discussed in subsection 4.6.3. The first column in the table contains the objective under consideration of being dropped. Hence, the objectives considered for calculating R for the objective f_m belong to the set $F_R - f_m$, shown in the second column. The calculated value of R is shown in the third column. From the table, it is clear seen that eliminating any objective induces considerable change in the percentage of non-dominated solutions, as reflected in low values of R. Thus, none of the objectives are redundant and, consequently, the relevant set of objectives is identified as $\{f_1, f_2, f_3\}$.

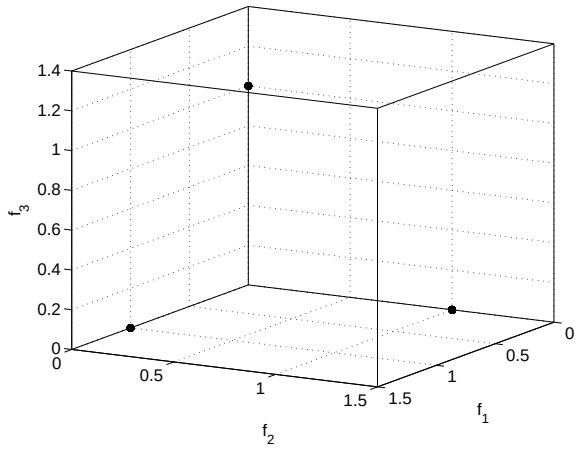


Figure 4.15: Final population obtained for DTLZ2 using PCSEA

Table 4.8: Dimensionality analysis for 3-objective DTLZ2 problem

f_m	Objectives considered $(F_R - f_m)$	R	Discard f_m ?
f_1	f_2, f_3	0.33	No
f_2	f_1, f_3	0.33	No
f_3	f_1, f_2	0.33	No

Similar experiments are performed on DTLZ2 with higher numbers of objectives. The proposed approach is able to accurately identify the dimensionality

of the Pareto front for up to 20 objectives studied. The results for DTLZ2(M) problems are summarized in Table 4.9. The efficiency of the approach can be inferred from the fact that a much higher number of function evaluations (400 $\times$ 2000) was used in the earlier study [97] to solve DTLZ2(5) accurately. Using the proposed approach, the same problem is solved accurately in a mere 200 $\times$ 500 evaluations. In addition, a problem with a much larger number of objectives, DTLZ2(20), is solved in only 200,000 evaluations. The last column in the table shows the success rate, *i.e.* the number of times correct dimensionality is identified, out of the 30 independent runs conducted. It is seen that the approach demonstrates consistent results over multiple trial runs.

Table 4.9: Results obtained for DTLZ2 (M) test problems

Test Problem	Critical set of objectives	Success rate
DTLZ2 (5)	f_1, f_2, f_3, f_4, f_5	30/30
DTLZ5 (10)	$f_1, f_2, \dots f_{10}$	30/30
DTLZ5 (15)	$f_1, f_2, \dots f_{15}$	30/30
DTLZ5 (20)	$f_1, f_2, \dots f_{20}$	30/30

DTLZ5 (I, M)

Next, the dimensionality of the Pareto front for DTLZ5-(2,3) is identified using the proposed approach. The Pareto front for DTLZ5-(2,3) is a 2-dimensional curve with endpoints (0,0,1) and ($1/\sqrt{2}$, $1/\sqrt{2}$, 0). The final population obtained using PCSEA for the problem is shown in Figure 4.16. It can be seen that the corners are accurately captured by the algorithm. Thereafter, the dimensionality analysis is performed on this converged population, as shown in Table 4.10. From the table, it is clear that omitting f_1 does not change the number of non-dominated solutions, thereby implying redundancy of this objective. Therefore, f_1 is removed from the set of relevant objectives. For the remaining set of

objectives (f_2 and f_3), it is observed that removing either objective significantly affects the number of non-dominated solutions, implying that both of them are non-redundant. Hence, the set of relevant objectives is identified as $\{f_2, f_3\}$. It is to be noted here that different sequences of omitting objectives might identify a different set of relevant objectives. In such a case, *either* of the reduced objective sets can be used for reconstructing the Pareto front; for example, in this case, if the sequence of dropping the objectives is reversed (*i.e.* f_3 is dropped first instead of f_1), the reduced set of objectives will be identified as $\{f_1, f_3\}$, as seen from Table 4.11. To validate this observation, NSGA-II is run using the reduced sets of objectives (both cases, $\{f_1, f_3\}$ and $\{f_2, f_3\}$). The recombination and mutation parameters used for NSGA-II are the same as those given in Table 4.6. For ease of visualization, a small population size (20) is used. The population is evolved for 200 generations. The original objectives are then reconstructed from the final population, and plotted in Figure 4.17. A population obtained by running NSGA-II on the original set of objectives is also plotted. It can be seen that the approximations of the Pareto front obtained for all three cases coincide, which confirms that the original 3D problem can be solved using reduced sets of objectives identified by the proposed methodology. However, in some cases, not all sets of reduced objectives may give the entire front, a limitation that will be highlighted in the next section.

Table 4.10: Dimensionality analysis for DTLZ5-(2,3)

f_m	Objectives considered ($F_R - f_m$)	R	Discard f_m ?
f_1	f_2, f_3	1.00	Yes
f_2	f_3	0.50	No
f_3	f_2	0.50	No

Next, the reducible set of test problems DTLZ-(5,M) is attempted for various

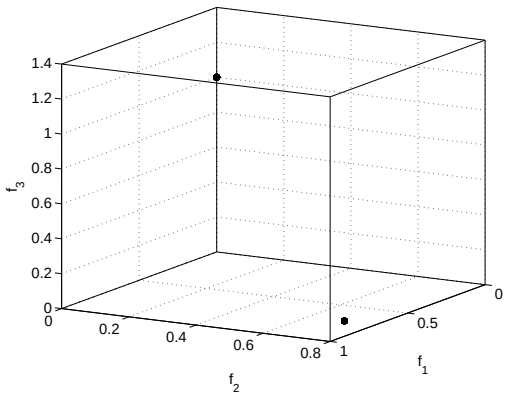


Figure 4.16: Final population obtained for DTLZ5-(2,3) using PCSEA

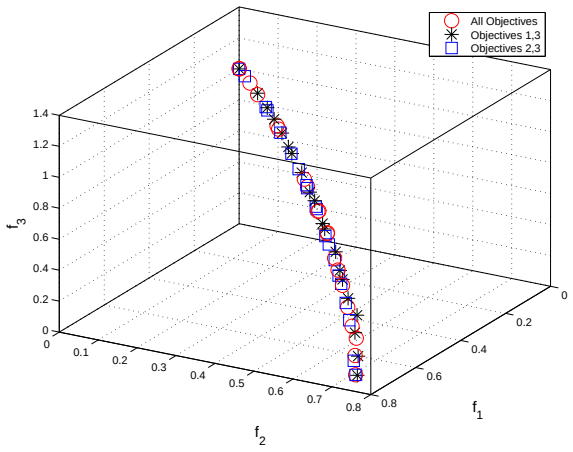


Figure 4.17: Non-dominated set obtained for DTLZ5-(2,3) using all objectives, compared with those using two relevant objectives

Table 4.11: Dimensionality analysis for DTLZ5-(2,3)

f_m	Objectives considered $(F_R - f_m)$	R	Discard f_m ?
f_3	f_1, f_2	0.50	No
f_2	f_1, f_3	1.00	Yes
f_1	f_3	0.50	No

values of M. The dimensionality analysis for one case, DTLZ-(5,10) is shown in Table 4.12. It can be seen that omitting the objectives does not substantially affect the number of non-dominated solutions until all the redundant objectives are dropped, as reflected in high R values for these objectives. However, reducing the objective set any further results in a substantial decrease in the non-dominated solutions, as seen from the low R value for these cases $(f_6, f_7, \ldots f_{10})$. Hence, the reduced objective set is identified as $\{f_6, f_7, f_8, f_9, f_{10}\}$. The results for the rest of the test problems are summarized in Table 4.13. As for the DTLZ2 test problems, it is seen that the algorithm is able to consistently identify the reduced sets of objectives across multiple runs. The success rate mentioned in the last column indicates the number of times the algorithm is able to identify these objectives in the reduced set, out of 30 runs. For a few runs for DTLZ5 (I, M), the dimensionality analysis identifies an additional objective apart from the last five (thus giving a total of six relevant objectives). These cases are limited in number, as seen in Table 4.13, and are counted as unsuccessful runs. While not the exact minimum subset, still a close approximation of it is found in such cases.

Once again, the efficacy of the proposed approach compared with existing approaches in the literature can be noted. The test problems shown in bold in Table 4.13 have been reported earlier in [91]. The numbers of evaluations used in these studies compared with those used in this study are listed in Table 4.14. It can be seen that the numbers of evaluations required to solve the problems using

Table 4.12: Dimensionality analysis for DTLZ-(5,10)

f_m	Objectives considered $(F_R - f_m)$	R	Discard f_m ?
f_1	$f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}$	1.00	Yes
f_2	$f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}$	1.00	Yes
f_3	$f_4, f_5, f_6, f_7, f_8, f_9, f_{10}$	1.00	Yes
f_4	$f_5, f_6, f_7, f_8, f_9, f_{10}$	1.00	Yes
f_5	$f_6, f_7, f_8, f_9, f_{10}$	0.83	Yes
f_6	f_7, f_8, f_9, f_{10}	0.33	No
f_7	f_6, f_8, f_9, f_{10}	0.06	No
f_8	f_6, f_7, f_9, f_{10}	0.06	No
f_9	f_6, f_7, f_8, f_{10}	0.06	No
f_{10}	f_6, f_7, f_8, f_9	0.06	No

Table 4.13: Results obtained for DTLZ-(I, M) test problems

Test Problem	Reduced set of objectives	Success rate
DTLZ5-(5-10)	$f_6, f_7, f_8, f_9, f_{10}$	30/30
DTLZ5-(5-20)	$f_{16}, f_{17}, f_{18}, f_{19}, f_{20}$	26/30
DTLZ5-(5-30)	$f_{26}, f_{27}, f_{28}, f_{29}, f_{30}$	24/30
DTLZ5-(5-40)	$f_{36}, f_{37}, f_{38}, f_{39}, f_{40}$	26/30
DTLZ5-(5,50)	$f_{46}, f_{47}, f_{48}, f_{49}, f_{50}$	27/30
DTLZ5-(5,60)	$f_{56}, f_{57}, f_{58}, f_{59}, f_{60}$	28/30
DTLZ5-(5,70)	$f_{66}, f_{67}, f_{68}, f_{69}, f_{70}$	25/30
DTLZ5-(5,80)	$f_{76}, f_{77}, f_{78}, f_{79}, f_{80}$	26/30
DTLZ5-(5,90)	$f_{86}, f_{87}, f_{88}, f_{89}, f_{90}$	26/30
DTLZ5-(5,100)	$f_{96}, f_{97}, f_{98}, f_{99}, f_{100}$	28/30

the proposed approach are significantly less than those used earlier. In addition, DTLZ-(I, M) problems with many more objectives (up to 100) are also solved using the proposed approach, using very few function evaluations.

Table 4.14: Number of evaluations used for DTLZ5-$(5, M)$ test problems

	Number of evaluations	
Test Problem	Earlier studies	Present studies
DTLZ5-(5-10) [91]	$2 \times 100 \times 5000$	200×200
DTLZ5-(5-20) [91]	$2 \times 100 \times 5000$	200×200
DTLZ5-(5-30) [91]	$2 \times 100 \times 5000$	200×200

WFG

The final population (of size 200, evolved over 200 generations) for $\text{WFG3}_{\text{conv1}}$ obtained using PCSEA is shown in Figure 4.18. The algorithm is able to capture the corner solutions, as seen from the figure. In addition, the algorithm obtains a few additional solutions on the curve due to the L_2 norm minimization. While these solutions do not lie on corners of the curve (intersection of the Pareto front boundaries), they are still close to the boundaries of the Pareto front. These solutions are present in the final population because they have low values of the L_2 norm compared with those on any of the corners. For concave problems, such as those discussed in the previous subsections, this case does not arise.

The dimensionality analysis of the problem is shown in Table 4.15. As the omission of any objective from the set significantly affects the number of non-dominated solutions, all objectives are identified as relevant.

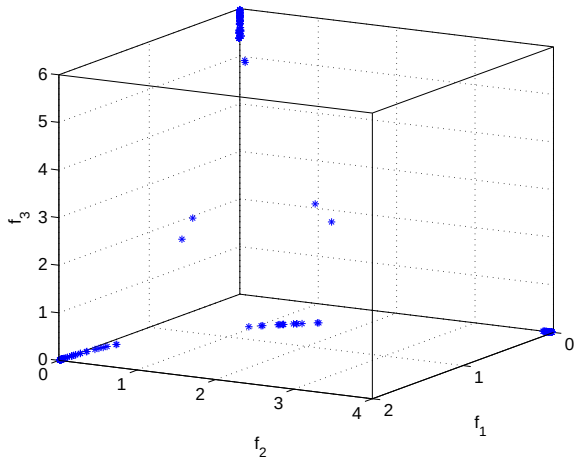


Figure 4.18: Final population obtained for $\text{WFG3}_{\text{conv1}}$ using PCSEA

The final population (of size 200, evolved over 200 generations) for $\text{WFG3}_{\text{conv2}}$ obtained using PCSEA is shown in Figure 4.19. Similar to the case of $\text{WFG3}_{\text{conv1}}$, PCSEA obtains the corner solutions along with some intermediate solutions on

Table 4.15: Dimensionality analysis for WFG3$_{\text{conv1}}$ problem

f_m	Objectives considered $(F_R - f_m)$	R	Discard f_m ?
f_1	f_2, f_3	0.03	No
f_2	f_1, f_3	0.03	No
f_3	f_1, f_2	0.03	No

the Pareto front boundaries. The dimensionality analysis of the obtained solutions is shown in Table 4.16. As mentioned before, the choice of the parameters $A_1 = 1$ and $A_2 = 0$ makes the Pareto front of the problem degenerate, as can be seen in the Table 4.16. Only two (f_2 and f_3) of the three objectives are identified as relevant whereas, with a different sequence, f_1 and f_3 can be identified as relevant. To verify the dimensionality analysis, the results using both the original and the reduced sets of objectives are shown in Figure 4.20. For better visualization, a small population size (20) is used. Since the population size is small, the algorithm is run for a larger number of generations (2000) to obtain the Pareto front. The number of evaluations is the same as that used to generate Figure 4.14(b). Also, the WFG$_{\text{conv2}}$ problem takes a larger number of evaluations than does the DTLZ5-(2,3) problem previously discussed.

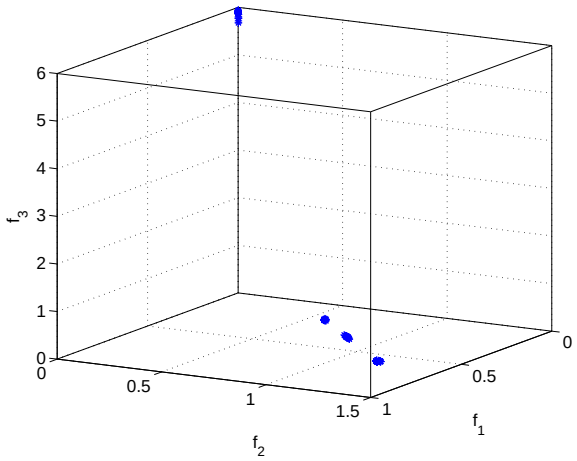


Figure 4.19: Final population obtained for WFG3$_{\text{conv2}}$ using PCSEA

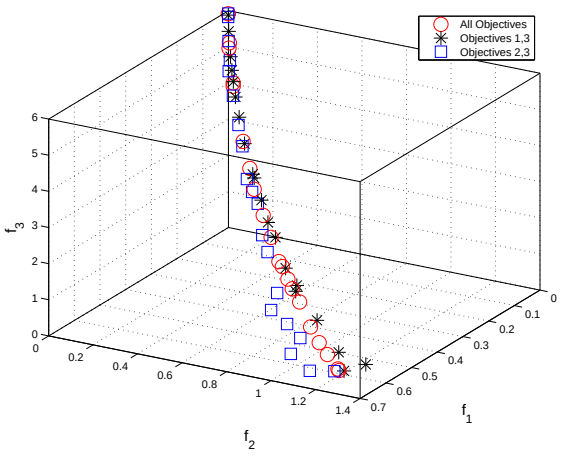


Figure 4.20: Approximation of the Pareto front obtained for WFG3$_{\text{conv2}}$ using all objectives, compared with the approximations obtained using two relevant objectives

Table 4.16: Dimensionality analysis for WFG3$_{\text{conv2}}$

f_m	Objectives considered $(F_R - f_m)$	R	Discard f_m ?
f_1	f_2, f_3	0.89	Yes
f_2	f_3	0.11	No
f_3	f_2	0.11	No

The value of C (cutoff value of R) used in the presented studies is set to 0.8. However, from the dimensionality analysis for the problems studied, it can be seen that, even if a cutoff value as low as 0.6 is used instead, the results would remain unchanged as there is a significant decrease in the number of non-dominated solutions when omitting any of the relevant objectives. This suggests that the proposed methodology is not very sensitive to the choice of the threshold. Also, if a higher value of threshold (say 0.9) is used, the elimination of objectives will be stricter, meaning more *conservative* estimate of the reduced dimensionality will be obtained, *i.e.* although the reduced set of objectives obtained will be able to capture the front, it may not be the smallest possible subset to do so.

4.7.4 Engineering design examples

After demonstrating the performance of the proposed dimensionality reduction procedure on commonly studied benchmark problems, its performance is investigated on two real life engineering design problems: a water resource problem [112] and a radar waveform optimization problem [113]. The experiments are detailed in the following subsections.

Water resource problem

The water resource problem, proposed by Musselman and Talavage [112] consists of 5 objectives and 7 constraints; and its formulation is given in Appendix E.9. A detailed description of the problem can be found in [112]. The problem was also studied earlier from a dimensionality reduction perspective by Saxena and Deb [100].

In an attempt to identify the true dimensionality of the problem, PCSEA is run on the problem, using the same crossover and mutation parameters as in the previous section. A population of 200 solutions is evolved for 200 generations. Table 4.17 shows the dimensionality reduction analysis of the population obtained using PCSEA. From the table, it is clear that omitting objectives f_1 and f_4 does not significantly affect the number of non-dominated solutions and, hence, they can be deemed redundant. The reduced set of objectives is identified as $\{f_1, f_2, f_5\}$. Omitting the objectives in a different order (for example, starting from f_5 and going down to f_1), identifies the reduced objective set as $\{f_2, f_3, f_5\}$, as shown in Table 4.18. Thus, either of these two sets can be used to reconstruct the Pareto front. To validate this result, the water resource problem is optimized using both sets of reduced objectives and the results compared with the those obtained using the original set of objectives. The function value plots for the

results obtained are shown in Figure 4.21. In order to aid visualization, the values shown in the plot are scaled appropriately by constant factors suggested in [69]. It can be seen that the plots corresponding to the reduced set of objectives closely match those obtained using the original set of objectives. This confirms that either of the reduced sets obtained is sufficient to obtain the Pareto front for the test problem.

Table 4.17: Dimensionality analysis for water resource problem

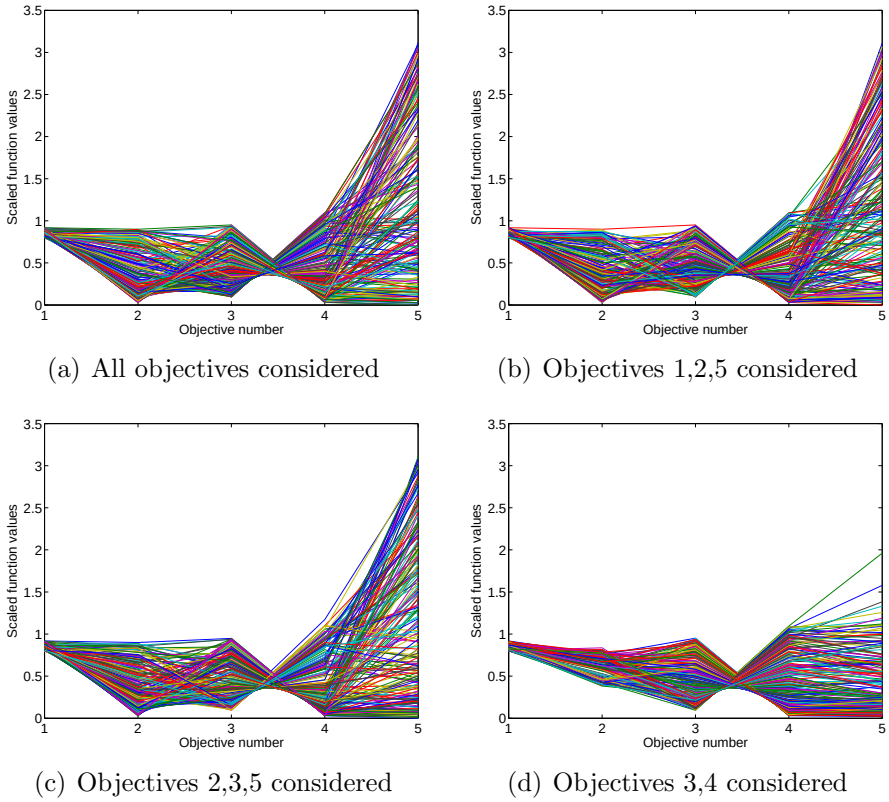
Objectives f_m	Objectives considered $(F_R - f_m)$	R	Discard f_m ?
f_1	f_2, f_3, f_4, f_5	1.00	Yes
f_2	f_3, f_4, f_5	0.20	No
f_3	f_2, f_4, f_5	0.39	No
f_4	f_2, f_3, f_5	0.99	Yes
f_5	f_2, f_3	0.05	No

Table 4.18: Dimensionality analysis for water resource problem

Objectives f_m	Objectives considered $(F_R - f_m)$	R	Discard f_m ?
f_5	f_1, f_2, f_3, f_4	0.20	No
f_4	f_1, f_2, f_3, f_5	0.99	Yes
f_3	f_1, f_2, f_5	1.00	Yes
f_2	f_1, f_5	0.04	No
f_1	f_2, f_5	0.40	No

Also shown in Figure 4.21 are the results obtained using the reduced set of objectives suggested by Saxena and Deb [100]. According to the PCA-based scheme employed by them, the reduced set consists of objectives $\{f_3, f_4\}$. However, from Figure 4.21, it can be observed that, when the problem is solved using these two objectives, certain regions of the front are not obtained, which is reflected as missing values in the function value plots.

Since the reduced set has fewer objectives than the original set, an MOEA is likely to converge faster to the front using the reduced set. This can be verified



(a) All objectives considered

(b) Objectives 1,2,5 considered

(c) Objectives 2,3,5 considered

(d) Objectives 3,4 considered

Figure 4.21: Function value plots of the final populations obtained for the water resource problem using various combination of objectives

by comparing the performances for different formulations (original objectives and the reduced sets). For this study, multiple (30) runs of NSGA-II are conducted for the above 4 formulations of the water resource problem, considering: (a) all objectives; (b) objectives 1,2,5; (c) objectives 2,3,5; and (d) objectives 3,4. For cases b-d, the rest of the objectives are reconstructed using the solutions in the final population. The results are compared using the hypervolume metric. To estimate the hypervolume, the MATLAB code available from [114] is used. A higher value of hypervolume indicates better performance. The reference point for the calculation of the hypervolume is set to $\max(f_1)$,$\max(f_2)$, $\ldots \max(f_5)$ found across all runs performed. A very limited number of evaluations (population size 100, evolved for 25 generations) are used for the study. The crossover and

mutation parameters used are the same as in Table 4.6. The statistics for the hypervolume metrics across all runs is given in Table 4.19. It can be seen that the mean hypervolume values obtained using the reduced sets of objectives (both $\{f_1, f_2, f_5\}$ and $\{f_2, f_3, f_5\}$) are better that those obtained using all the objectives. The mean hypervolume values obtained using objectives $\{f_3, f_4\}$ are inferior to all others, indicating that using these two objectives may not be sufficient to adequately represent the problem.

Table 4.19: Comparison of results (hypervolume) for water resource problem using original and reduced set of objectives

	f_1, f_2, f_3, f_4, f_5	f_2, f_3, f_5	f_1, f_2, f_5	f_3, f_4
Mean	0.061700	0.064700	0.063400	0.059000
Best	0.075000	0.088000	0.085000	0.078000
Worst	0.046000	0.047000	0.044000	0.044000
S.D.	0.006374	0.008226	0.009608	0.008204

Radar problem

The second example considered here is that of the radar waveform design problem [113]. The problem contains 9 objectives and 4 to 12 decision variables. Since objectives f_1 to f_8 are to be *maximized*, they are first multiplied by -1, in order to convert the problem to one with minimization of all objectives. The code to evaluate the objectives is taken from [115]. As in the previously reported study [113], an unconstrained problem is considered.

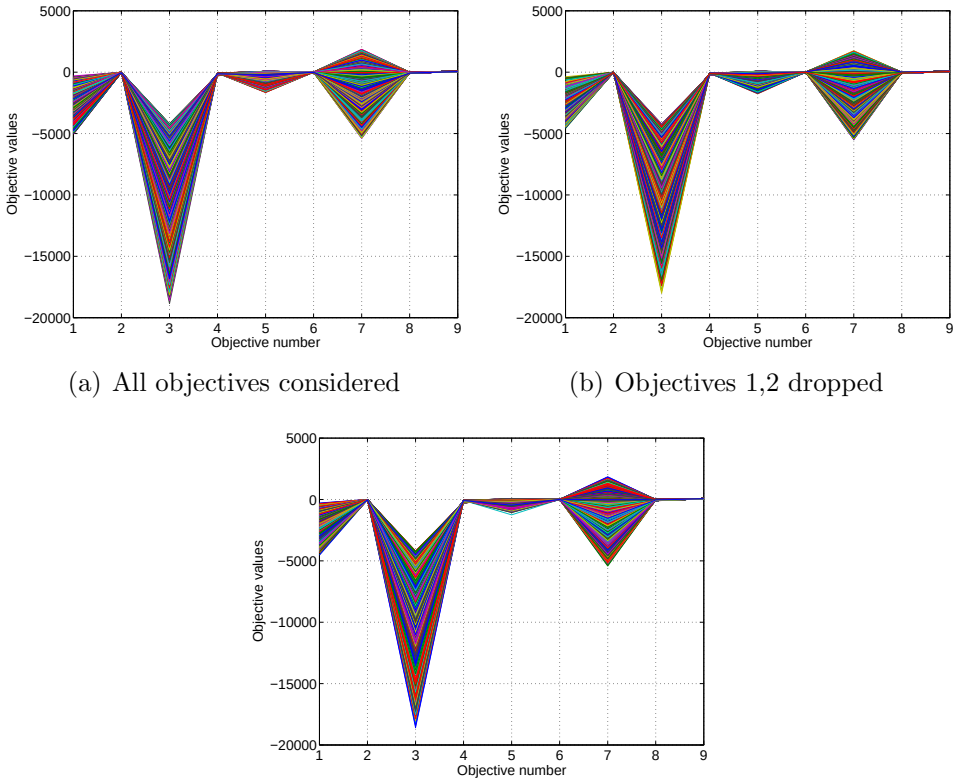
To identify possible redundant objectives, PCSEA is run on the problem. A population of 200 solutions is evolved for 100 generations, resulting in 20,000 evaluations as used in [113]. Also, since objectives f_1 to f_8 can assume negative values, a constant value of 20000 is added to all the objectives while running PCSEA, so that all the objectives take only positive values. Consequently, the

minimization of the L_2 norm corresponds to the minimization of objectives, as discussed in Section 4.6.2.

Of thirty independent runs of PCSEA, using a threshold $C = 0.8$, most result in identifying 6 or 7 objectives as relevant. The different relevant sets identified are listed in Table 4.20. Subsequently, two of them – $\{f_3, f_4, f_5, f_6, f_7, f_8, f_9\}$ and $\{f_3, f_4, f_6, f_7, f_8, f_9\}$ – are used to solve the problem. NSGA-II is run for 20,000 evaluations using the reduced sets (population of size 200 evolved for 100 generations). The non-dominated solutions collected from 10 independent NSGA-II runs are shown in Figure 4.22. Also shown in the figure are the solutions obtained from NSGA-II (10 runs) considering all 9 objectives. It can be seen that the function value plots obtained using the identified reduced set(s) of objectives are almost identical to that obtained using the original set.

Table 4.20: Reduced set of objectives obtained for Radar problem

Run	Reduced objectives	Run	Reduced objectives
1	$f_2, f_3, f_4, f_6, f_7, f_8, f_9$	16	$f_2, f_3, f_4, f_5, f_7, f_8, f_9$
2	$f_2, f_3, f_4, f_5, f_6, f_8, f_9$	17	$f_2, f_3, f_5, f_6, f_7, f_8, f_9$
3	$f_2, f_3, f_4, f_6, f_7, f_8, f_9$	18	$f_2, f_3, f_4, f_5, f_7, f_8, f_9$
4	$f_3, f_4, f_6, f_7, f_8, f_9$	19	$f_3, f_4, f_5, f_6, f_7, f_8, f_9$
5	$f_3, f_4, f_5, f_6, f_7, f_8, f_9$	20	$f_3, f_4, f_5, f_6, f_7, f_8, f_9$
6	$f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9$	21	$f_3, f_4, f_5, f_6, f_7, f_9$
7	$f_2, f_3, f_4, f_5, f_6, f_7, f_9$	22	$f_2, f_3, f_4, f_6, f_7, f_8, f_9$
8	$f_3, f_4, f_6, f_7, f_8, f_9$	23	$f_3, f_4, f_6, f_7, f_8, f_9$
9	$f_3, f_4, f_5, f_6, f_7, f_8, f_9$	24	$f_3, f_4, f_5, f_6, f_7, f_8, f_9$
10	$f_3, f_4, f_6, f_7, f_8, f_9$	25	$f_3, f_4, f_5, f_6, f_7, f_8, f_9$
11	$f_3, f_4, f_5, f_6, f_7, f_8, f_9$	26	$f_3, f_4, f_5, f_6, f_7, f_8, f_9$
12	$f_2, f_3, f_5, f_6, f_7, f_8, f_9$	27	$f_3, f_4, f_5, f_6, f_7, f_8, f_9$
13	$f_2, f_3, f_4, f_6, f_7, f_8, f_9$	28	$f_2, f_3, f_4, f_5, f_6, f_7, f_9$
14	$f_3, f_4, f_5, f_6, f_7, f_8, f_9$	29	$f_3, f_4, f_5, f_6, f_7, f_8, f_9$
15	$f_2, f_3, f_4, f_6, f_7, f_8, f_9$	30	$f_2, f_3, f_4, f_5, f_7, f_8, f_9$



(a) All objectives considered

(b) Objectives 1,2 dropped

(c) Objectives 1,2,5 dropped

Figure 4.22: Function value plots of the final populations obtained for the radar problem using various combination of objectives

Another observation worthwhile mentioning is the concurrence of the results with some of the predictions in [113]. From the problem formulation, it is expected that certain objectives have a degree of correlation with each other, *e.g.* the objective pairs $\{f_1, f_3\}$, $\{f_5, f_7\}$, $\{f_2, f_4\}$, and $\{f_6, f_7\}$ are not in strong conflict with each other. From Table 4.20, it is clear that any of these pairs of objectives are not dropped simultaneously during objective reduction. For all these runs, at least one objective of these non-conflicting sets is retained (*e.g.*, either f_1 or f_3 is always retained; similarly at least one objective out of f_2, f_4 and at least one out of f_5, f_6, f_7 is always retained).

4.7.5 Limitations of proposed dimensionality reduction approach

As indicated in [113], the non-dominated front obtained using NSGA-II with 20,000 evaluations for the original problem is likely to be quite close to its Pareto front. In such a case (*i.e.*, if the problem can be solved using the original set of objectives), the objective reduction may not provide a major advantage in terms of convergence. However, this experiment further verifies the validity of the presented approach for identifying possible redundant objectives for many-objective problems.

While the proposed approach for finding the redundant objectives works efficiently for the problems studied in this chapter, there are a number of limitations to the method which might deteriorate its performance for certain problems. While some of the limitations are due to the use of (only) corner solutions to identify dimensionality, others relate to the way in which PCSEA is implemented in the current work. These limitations are summarized below.

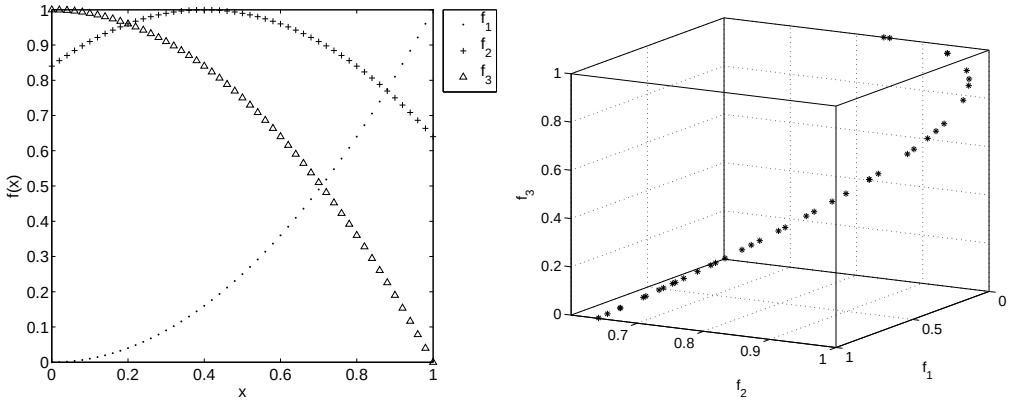
1. Firstly, since only the corner solutions are used for predicting the relevant objectives, the resulting set may not be able to capture the entire information regarding conflict among all the objectives. Consequently, some of the relevant sets identified using the proposed approach may not be able to capture the entire Pareto front. To illustrate this, consider the three-objective, one-variable example given in Equation 4.3.

$$\begin{aligned} &\text{Minimize} \\ &f_1(x) = x^2 \\ &f_2(x) = 1 - (x - 0.4)^2 \\ &f_3(x) = 1 - x^2 \\ &0 \leq x \leq 1. \end{aligned} \tag{4.3}$$

Individual variations in objectives are shown in Figure 4.23(a) and the Pareto front for the problem in Figure 4.23(b). According to the definition of the corner solutions considered here, the Pareto front for this problem has two corners, corresponding to $x = 0$ and $x = 1$. The dimensionality analysis using these two corner solutions reveals that either $\{f_1,f_3\}$ or $\{f_1,f_2\}$ can be used as relevant objectives for the problem. However, these two sets of objectives have different regions of conflict. f_1 and f_3 are conflicting in $x \in [0, 1]$, whereas f_1 and f_2 are conflicting for $x \in [0.4, 1]$. Thus, while one set of objectives (f_1, f_3) is able to generate the whole Pareto front, the second (f_1, f_2) is able to achieve only a part of it. Similarly, for certain problems, different sets may give different regions of the Pareto front, with or without some overlap. As only the corner solutions are being used here to predict dimensionality, such conflicts *cannot* be entirely captured.

However, on the other hand, it is also important to realize that for many-objective problems, even if only a part of the Pareto front is achievable using a reduced set of objectives, it is still more advantageous than using the entire set of objectives (and not getting any near-optimal solutions). In such a case, although a user preference for which part of the front is preferable is not currently built in to the proposed approach, it is definitely

an enhancement worth exploring.



(a) Objective functions (b) Pareto front

Figure 4.23: Test problem 1 (Equation 4.3)

2. Another concern regarding the presented approach is its performance for problems in which certain objective(s) do not contribute to the extremities of the Pareto front, but contribute elsewhere. While it might seem from the nomenclature that *corner* solutions only refer to the physical extremities of the Pareto front, it is to be noted from Definition 2 that the corners include the solutions with a minimum of each individual objective, irrespective of which part of the front they are contributing to. Hence, even if, in theory, it is possible to have a problem with objective(s) that contribute only to the non-extreme regions of the Pareto front, the solutions with extreme values of these objectives will still be preserved by the PCSEA (and used for dimensionality reduction) irrespective of their location on the front.

3. The use of the L_2 norm in the implementation of PCSEA may result in certain additional solutions which are not corner points; for example, if PCSEA is run for the above test problem 1, it yields three solutions, as shown in Figure 4.24. One of these solutions does not have any objective at

its minimum value. The presence of this artifact solution overestimates the dimensionality (as it results in identifying three relevant objectives). Such solutions, if present, should ideally be removed from the final population before the dimensionality analysis which, again, is a exponentially growing computational task because all possible sets of objectives have to be considered to locate all intermediate solutions. Alternatively, a more intelligent selection scheme, which does not result in such artifact solutions in the final population, is required.

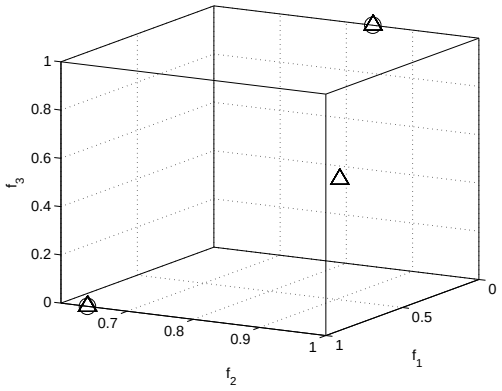


Figure 4.24: Solutions obtained using PCSEA for test problem 1. The actual corners for the Pareto front are shown with circles.

4. The use of a user-defined threshold to determine the relevance of the objectives is also an approximation. Ideally, *any* change in the non-dominated set due to exclusion of an objective under consideration should imply that it is a relevant objective. Similarly, only if $R = 1$, should an objective be deemed redundant. However, this will work only if the corner solutions are exactly obtained as opposed to the present case in which the solutions obtained are approximations to the Pareto corners.

   Consider an optimization problem in which all the N_c exact corners con-

sist of solutions where *only one* of the objectives is the minimum (*e.g.*, Figure 4.11(a)). Now, if an objective is dropped, the corner solution corresponding to that objective being minimum, will become dominated by other corner solutions. As only a single corner solution is dominated each time an objective is dropped, the R value for such a case would be 1 - $1/N_c$. Now, for a large number of corners (N_c), the value of R will approach unity. So, for a fixed value of the threshold C, the objectives will be marked redundant although all objectives are relevant. Thus, approximate analysis using a threshold may be misleading for such cases.

4.7.6 Applicability of proposed dimensionality reduction approach

Before concluding the discussion on the proposed dimensionality reduction technique, it is important to summarize some of the observations from the numerical experiments presented in the previous sections, in order to usefully apply the proposed approach to many-objective problems.

Firstly, it has to be kept in mind that the focus of the current work is to investigate if certain objectives can be omitted from a problem in order to improve its convergence with as little compromise as possible in terms of the final solution achieved. The term *redundancy* is also used here accordingly, which might differ from some other studies in which this term is used in the context of decision making. For the many-objective optimization problems in which the Pareto front is somehow achievable without prohibitive computational effort, the proposed approach does not offer any significant benefits in terms of convergence, other than perhaps reducing the number of evaluations by using a smaller set of

objectives.

Secondly, for some many-objective problems, it may be possible to find objective(s) that are completely redundant (meaning that the entire Pareto front can be achieved without considering them); e.g., if there are two objectives with strictly monotonic behavior throughout the search space (*i.e.*, if both are either increasing or decreasing), the exclusion of one of them will have no effect on the problem. This seems to be the case for the DTLZ5-(I, M) problems studied here, for which the exact Pareto fronts can be achieved using the reduced set. However, in other cases, the redundant objective (as identified by the proposed approach) may still have a contribution to the Pareto front and, thus, if that objective is removed, the Pareto front will be compromised, *i.e.*, some part of the Pareto front will not be achieved. In the proposed approach, since only the extreme values of the objectives are searched for (and used for dimensionality reduction), it is not possible to guarantee whether the complete front can be constructed using the identified reduced set of objectives. Evidently, this also means that the extreme solutions are *not* sufficient to characterize the *entire* front. Therefore, depending on the nature of conflict between the objectives, the applicability and benefits of the proposed approach can vary.

At the same time, as previously emphasized, it is also to be considered that it is difficult to predict such conflicts even by analyzing solutions obtained from a conventional MOEA. This is because a) since the Pareto-dominance sorting does not create any convergence pressure, it is difficult to obtain an approximation close to the front with all objectives even with a very large number of evaluations; and b) even a reasonably large population size is insufficient to approximate the front for a many-objective problem. On the other hand, preference-based methods may give solutions only in a specific region of interest and can, therefore, be

misleading for a dimensionality analysis of the entire front. In such cases, the offline dimensionality analysis using the PCSEA, which takes far fewer function evaluations, is certainly attractive. Even if a part of the front can be recovered using a reduced objectives (from the corner search), it is more desirable than solving the problem using all objectives and not achieving any near optimal solutions.

The experiments presented here on three widely studied benchmark problems (DTLZ2, DTLZ5-(I, M) and WFG) and two engineering design problems (water resource and radar waveform), and their comparison with previously reported studies clearly indicate that, by using the corner solutions, it is possible to estimate redundancy in the objectives with reasonable accuracy and, consequently, reduce the computational effort for achieving the Pareto front. However, like other heuristic methods, there are elements of uncertainty in the performance, and the best (or exact) results cannot be perpetually guaranteed.

4.8 Summary

In this chapter, the challenges involved in dealing with many-objective problems are highlighted and, to enhance existing methods, two different approaches are discussed. One is the use of improved secondary ranking and the other is identifying the reduced dimensionality of a problem (if applicable) with minimal computational expense. In addition, a new diversity metric, which can be used to evaluate various algorithms in terms of their performance in maintaining diversity, is proposed. The major contributions of the chapter are summarized below.

Secondary ranking : Cluster-sort

The Cluster-sort ranking is proposed and its performance studied in terms of convergence and diversity. It is observed that though it does not accelerate convergence, it is able to achieve a good diversity among the population members. This ranking method can be used in conjunction with approaches with faster convergence.

Secondary ranking : Mod-ϵ-DOM

Mod-ϵ-DOM is proposed to eliminate the drawback of the conventional -ϵ- DOM ranking procedure. The results obtained from using it on benchmark problems indicate that it has better convergence with similar levels of diversity when compared with -ϵ-DOM.

Dimensionality reduction using Pareto corner search

A novel methodology for predicting the true dimensionality of a Pareto front and identifying redundant objectives, if any, is presented. The key idea is that for many-objective problems, as it is impractical to search for an approximation for the *whole* front even with a reasonably huge population size, it may be worthwhile searching for a few key solutions instead, such as the corners of the Pareto front. By focusing the search on these key solutions, the search can get much closer to the front than if it were searching for the entire front. In addition, the corner solutions have maximum diversity. The obtained set of solutions is much more suitable for dimensionality analysis as it is closer to the Pareto front and has good diversity in terms of range of function values. Consequently, such a set is a better indicator of the dimensionality than that obtained by conventional MOEAs. PCSEA to search for the corners of the Pareto front, and a heuristic

technique identification of redundant objectives, are proposed.

The proposed methodology, while being simple, overcomes the following key issues.

1. As PCSEA does not use non-dominated sorting, it is not likely to suffer from the lack of selection pressure during the evolutionary search. Consequently, it takes fewer number of evaluations to find solutions close to the Pareto front.

2. The computational complexity of PCSEA does not grow exponentially with the number of objectives, unlike most hypervolume-based MOEAs. This makes it suitable for identifying dimensionality for problems with large numbers of objectives whereas the exponential complexity of hypervolume-based MOEAs limits their applicability to problems with up to 10 objectives or so.

3. A very large population size is not required with growing number of objectives. This is because the proposed approach does not attempt to approximate the entire Pareto front, but only a few characteristic solutions instead.

The efficacy of the proposed dimensionality reduction approach is demonstrated using three benchmark test problems and two engineering optimization problems. Dimensionality is predicted accurately for many-objective test problems containing up to 100 objectives (DTLZ5-(I, M), using significantly fewer function evaluations than those used previously in the literature. Comparison with previously reported studies clearly demonstrate the potential of the proposed approach. Although the approach is not without limitations, it provides a novel

way of evaluating many-objective optimization problems and approximating the Pareto front with, possibly, a smaller set of relevant objectives. Further studies are required to overcome the drawback of dimensionality reduction techniques using Pareto corner solutions (or some other key Pareto solutions), in order to correctly identify the relevant objectives for all types of problems.

Chapter 5

Large Scale Optimization II: Large number of variables

Abstract

While conventional Evolutionary Algorithms (EAs) can be used to efficiently solve problems with small numbers of design variables, their performance degrades for problems containing a large numbers of design variables because the search space grows exponentially. To efficiently solve such a problem, the "separability" of the problem can be effectively utilized in order to individually evolve smaller groups (collaborators) of variables. In this chapter, a cooperative coevolutionary algorithm with adaptive variable partitioning (CCEA-AVP) is proposed, which automatically partitions the variables using the correlation among them. The experiments demonstrate a competitive performance of CCEA-AVP across a range of separable and non-separable problems with large numbers of variables.

5.1 Overview

In the previous chapter, the challenges posed by large numbers of objectives in multi-objective optimization were discussed. In this chapter, another form of a "curse of dimensionality", which relates to the number of design variables in the problem, is discussed. Although EAs have proven to be quite competent for problems with a few variables, their performance tends to deteriorate for problems with a large number of design variables. This can be attributed to the exponential increase in the search space as the number of variables increases.

In order to efficiently solve optimization problems with a large number of variables, the notion of dividing the variable space into multiple partitions (or "subcomponents") of smaller dimensions, which are then evolved individually, was introduced by Potter and De Jong [116]. This approach is referred to as Cooperative Coevolutionary Algorithm (CCEA). These individual partitions also frequently interact (collaborate) with each other in order to improve global fitness. Using this approach, the modularity of an optimization problem can be exploited to generate solutions with higher fitness with relatively less computational expense. This approach is also commonly referred to as a "problem decomposition" or a "divide and conquer" strategy.

While the concept of evolving subcomponents of a problem independently and co-adaptively sounds natural and attractive, the dynamics of a CCEA is far more complex as compared to an EA [117]. In a CCEA, in addition to basic parameters (population size, number of generations, crossover and mutation rates), various other factors play a significant role in the algorithm's performance; such as the number of variable partitions, the number of generations for evolving each subpopulation, and collaboration strategies (single best, random etc). Without

a careful choice of these additional parameters, the performance of basic CCEA may exhibit a large variation in performance, ranging from good to poor, as echoed in the studies by Popovici and De Jong [118].

In an attempt to better understand the underlying dynamics of a CCEA, empirical studies have focused on its various aspects – choice of collaborators [119, 120], interaction frequency [121], sequential and parallel versions of information exchange (referred to as update timing) [122], and disconnection between the external goal and the internal behavior of CCEA [117], etc. All these studies have highlighted the complexity of the dynamics of CCEAs.

On a parallel front, CCEA and its variants such as coevolutionary particle swarms [123, 124], coevolutionary differential evolution [125], cooperative coevolutionary models based on self-adaptive neighborhood search differential evolution [126], non-dominance based cooperative coevolution, and the original CCEA [116] have been applied to a number of single- and multi-objective, separable and non-separable benchmark problems.

While excellent performances of CCEAs have been reported in many studies, very few techniques have been proposed for a methodical decomposition of a problem to capture the variable interactions appropriately. Apart from fixed partitioning, as proposed in the original CCEA, the most prominent scheme used for problem decomposition is known as *random grouping*[126, 124, 127] in which the variables are assigned randomly to different groups during the search, rather than assigning the variables into a fixed group for the whole search. Some of the previous studies [126] have suggested that the probability of assigning two interacting variables into a single sub-component is increased by random grouping. However, this probability decreases significantly if the number of interacting variables is higher. In the most recent studies [128], a decomposition

based on changes in improvement intervals has been proposed.

In this chapter, the drawbacks of the conventional CCEA are highlighted. Thereafter, a new CCEA with adaptive variable partitioning (CCEA-AVP) is proposed, which tries to identify the best collaborators adaptively during the search. Comparisons are conducted with conventional fixed-partition CCEAs in order to highlight the performance improvements attained. Numerical experiments on problems containing up to 100 variables are presented.

The remainder of this chapter is organized as follows. Details of the basic CCEA are discussed in Section 5.2. The proposed algorithm (CCEA-AVP) is described in Section 5.3. The numerical experiments and related discussions are presented in Section 5.4. A summary of the findings is given in Section 5.5.

5.2 Background

5.2.1 Basic CCEA

The CCEA proposed by Potter and De Jong [116][1] consists of 3 major steps:

1. Decompose the decision space into multiple subsystems of smaller dimensions. In the basic CCEA, the population is partitioned into s (>1) subpopulations, containing equal (or near equal) numbers of variables. The variables to be grouped into any particular subpopulation are chosen randomly at each CCEA generation.

2. Evolve each subsystem individually for a specified number of *subevolve* generations using a conventional EA. Since the complete set of variables

[1]Potter and De Jong used the term CCGA for their GA- based algorithm. CCEA has also been referred to as CCA in certain studies.

are required in order to evaluate objective value(s), collaborators are needed from other subpopulations in order to evolve any individual subpopulation. The following two strategies were suggested in [116]:

(a) Single best collaboration strategy: The subpopulation under consideration is combined with variable values corresponding to the best solution obtained so far. The subpopulations are evolved in a round-robin fashion, and the best function value is continuously updated.

(b) Random/best collaboration strategy: The subpopulation is combined with either the best *or* a random solution from remaining subpopulations, whichever has the better global fitness value.

3. After subpopulations are evolved for a specified number of generations, integrate them back together, so that the subpopulations can co-adapt and interact in order to obtain high global fitness value.

The pseudo-code of the basic CCEA is given in Algorithm 5.1.

5.2.2 Why are CCEAs attractive ?

As suggested by various studies in the past [116, 126], for many large-scale problems, CCEAs can provide much better solutions than conventional EAs. While dealing with a large number of variables, evolving different modules of a problem individually can expedite convergence rate appreciably as compared to traditional EAs which attempt to evolve all variables simultaneously. A simple illustration of this effect for four common scalable benchmark problems (*viz.* Rastrigin, Schwefel, Rosenbrock and Ackely, defined in Table 5.1) is shown in Table 5.3 (for 50 Dimension problem) and Table 5.4 (for 100 Dimension problems). Both algorithms are run for 100,000 evaluations and use identical crossover

Algorithm 5.1 Basic CCEA

Require: N_G (Number of CCEA Generations), N_V (Number of variables for the problem), N_P (Population size), CR (Crossover Rate), MR (Mutation Rate), Distribution index of Crossover and Distribution index of Mutation.

```
Initialize Population pop
Set Soln_best as the best individual in the population
for cgen = 2 to N_G do
    Split pop_cgen into s partitions pop_1,cgen, pop_2,cgen ... pop_s,cgen.{Each partition
    consisting of randomly assigned N_V/s variables. Let the V_j denote the
    indices of variables contained in a partition j}
    for i = 1 to s do
        Construct spop_i by combining variable values in partition i with
        best/random variable values from other partitions
        for subgen = 2 to N_SG do
            childpop_i,subgen ← Subevolve (spop_i,subgen−1)
            Evaluate (childpop_i,subgen)
            CP ← Sort (childpop_i,subgen + spop_i,subgen−1)
            spop_i,subgen ← Reduce (CP)
        end for
        Set pop_cgen(V_i) = spop_i
        Update Soln_best {Update best solution}
    end for
end for
```

and mutation parameters, listed in Table 5.2. The statistics shown are for 30 independent runs. For the CCEA, single best collaboration strategy is used[2]. The number of *subevolve* generations used is 10. Results are obtained using 5 and 10 equal partitions in CCEA. It can be seen that, for both 50D and 100D problems, the CCEA is able to achieve results which are, at times, orders of magnitude better than those obtained by the EA. Figures 5.1 and 5.2 show the convergence plot for the median run. The figures indicate faster convergence rates of CCEA compared with that of the conventional EA. For the presented studies, NSGA-II (equivalent to a real coded GA for single-objective optimization) is used

[2]Although the single best collaboration strategy makes the search greedy, it is used consistently throughout this study, in order to limit the variation in results due to stochastic selection of collaborators.

as the representative EA. This same EA is also used as the base algorithm for implementing CCEA, *i.e.* SBX and polynomial mutation are used for evolving solutions during the search. The ranking of feasible solutions is done based on the sorted objective values, while that of infeasible solutions is done based on constraint violations.

Table 5.1: Description of test functions used for the study

Test problem	Objective function
Rastrigin (Minimize)	$f(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$ $-5.12 \leq x_i \leq 5.12$
Schwefel (Minimize)	$f(x) = 418.9829n - \sum_{i=1}^{n} \left(x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right)$ $-500 \leq x_i \leq 500$
Rosenbrock (Minimize)	$f(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$ $-30 \leq x_i \leq 30$
Ackley (Minimize)	$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$ $-32 \leq x_i \leq 32$
Shifted Rotated Rastrigin [129] (Minimize)	$f(x) = \sum_{i=1}^{n} \left[z_i^2 - 10\cos(2\pi z_i) + 10\right]$ $-5.12 \leq x_i \leq 5.12, z = (x - o) \times M$, where o is the shifted global optimum, M is the linear Transformation Matrix (Rastrigin_M_D50).
G2 [55] (Maximize)	$f(x) = -\left\lvert \frac{\sum_{i=1}^{n}\cos^4(x_i) - 2\prod_{i=1}^{n}\cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right\rvert$ Subject to $g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$ and $g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0$ $0 \leq x_i \leq 10\ (i = 1, \ldots, n)$

Table 5.2: Parameters used for the study

Parameter	Value(s)
Population Size	100
Maximum function Evaluations	100000
Crossover probability	1.0
Mutation probability	0.1
Crossover distribution index	15
Mutation distribution index	20

Apart from obtaining better function values, the CCEA also offers an additional advantage of saving on the cost of evolving the population. This is because for each new solution generated during the run, CCEA has to deal with far fewer variables as only a part of the population is evolved at a time; whereas conventional EA has to perform recombination and mutation operators on all

Table 5.3: Results using CCEA (fixed partitions) and EA for 50D Rastrigin, Schwefel, Rosenbrock and Ackley

Test Problem	Algorithm	Best	Mean	Worst	S.D.
Rastrigin	Fixed 5	8.731096e-03	1.776637e-02	3.059190e-02	6.328858e-03
	Fixed 10	**5.779232e-03**	**1.154832e-02**	2.345529e-02	4.235983e-03
	EA	1.662080e+01	2.367404e+01	2.875840e+01	3.018061e+00
Schwefel	Fixed 5	4.353910e-02	1.895843e+02	4.738343e+02	1.411670e+02
	Fixed 10	**4.113830e-02**	**1.586701e+01**	1.185799e+02	4.095543e+01
	EA	5.291690e+02	1.331623e+03	2.766410e+03	4.266604e+02
Rosenbrock	Fixed 5	3.826361e+01	**6.667871e+01**	1.513366e+02	2.989787e+01
	Fixed 10	**2.656768e+01**	9.651688e+01	1.540250e+02	3.973974e+01
	EA	4.204690e+01	8.385528e+01	1.892910e+02	3.590212e+01
Ackley	Fixed 5	2.178630e-02	3.020990e-02	4.191100e-02	5.403417e-03
	Fixed 10	**1.804900e-02**	**2.684102e-02**	4.221360e-02	5.015803e-03
	EA	5.964560e-01	9.039441e-01	1.185250e+00	1.374061e-01

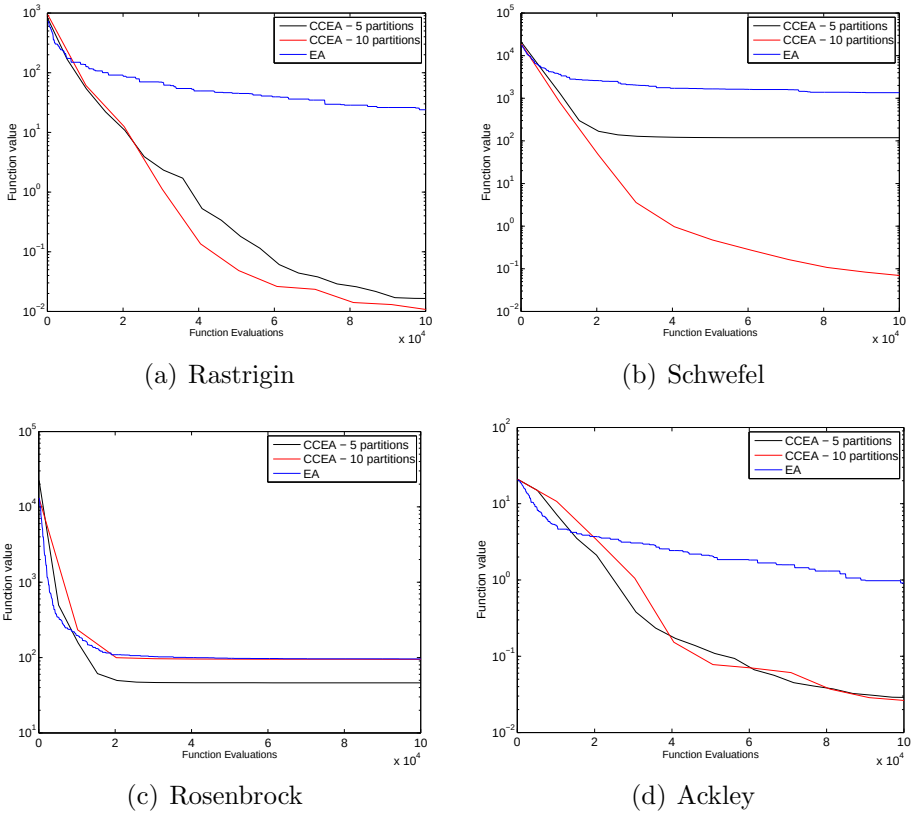
Table 5.4: Results using CCEA (fixed partitions) and EA for 100D Rastrigin, Schwefel, Rosenbrock and Ackley

Test Problem	Algorithm	Best	Mean	Worst	S.D.
Rastrigin	Fixed 5	2.759770e+00	6.940710e+00	1.332520e+01	2.324321e+00
	Fixed 10	**2.646480e-01**	**5.665662e-01**	2.433088e+00	5.157489e-01
	EA	1.649800e+02	1.840125e+02	2.036640e+02	1.209635e+01
Schwefel	Fixed 5	3.653240e+02	9.442860e+02	1.550470e+03	3.218071e+02
	Fixed 10	**1.507234e+00**	**3.610878e+02**	8.309508e+02	2.072720e+02
	EA	4.346710e+03	5.280894e+03	6.328550e+03	5.714214e+02
Rosenbrock	Fixed 5	**8.440033e+01**	1.472800e+02	2.613990e+02	4.064567e+01
	Fixed 10	8.771310e+01	**1.311911e+02**	3.167990e+02	4.927446e+01
	EA	1.259630e+02	2.385880e+02	3.612080e+02	5.010407e+01
Ackley	Fixed 5	2.263960e-01	2.689068e-01	3.276250e-01	2.660889e-02
	Fixed 10	**8.540569e-02**	**1.083423e-01**	1.413097e-01	1.431065e-02
	EA	3.975440e+00	4.172588e+00	4.423220e+00	9.634730e-02

variables simultaneously. For problems with large numbers of variables, this may result in a considerable computational overhead for conventional EAs, as mentioned in [116]. However, this effect may not be a major concern in terms of computational complexity if the cost of evaluating a solution itself is considerably higher than the cost of recombination and mutation operators.

5.2.3 Shortcomings of basic CCEA

Although it is clear from the previous subsection that the basic CCEA is capable of achieving much better results than an EA for certain problems, its performance is likely to deteriorate if the subsystems are not chosen appropriately. In

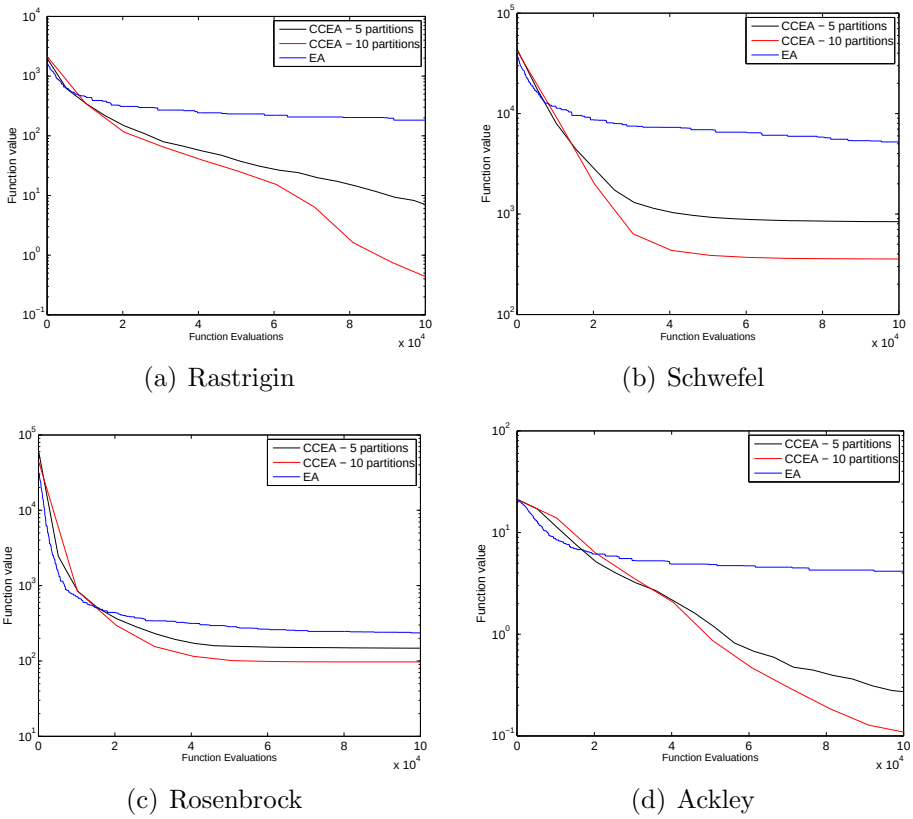


(a) Rastrigin (b) Schwefel

(c) Rosenbrock (d) Ackley

Figure 5.1: Convergence plot of the median runs obtained using EA and CCEA (fixed length partitions) for 50D problems

particular, random partitioning as used in the basic CCEA performs well only if interdependence between the variables are non-existent, which allows each subsystem to be evolved towards the global optimum variable values independently of other subsystems. However, if the problem has a strong interdependence among the variables, it becomes imperative for good performance that the partitioning is intelligent enough to create subsystems such that: (a) the variables within any subsystems are strongly dependent on each other, and (b) interactions among different subsystems are minimal. The problems without substantial inter-variable dependence are commonly referred to as *separable* problems, whereas problems with strong inter-variable dependence are known as *non-separable* problems.



(a) Rastrigin (b) Schwefel

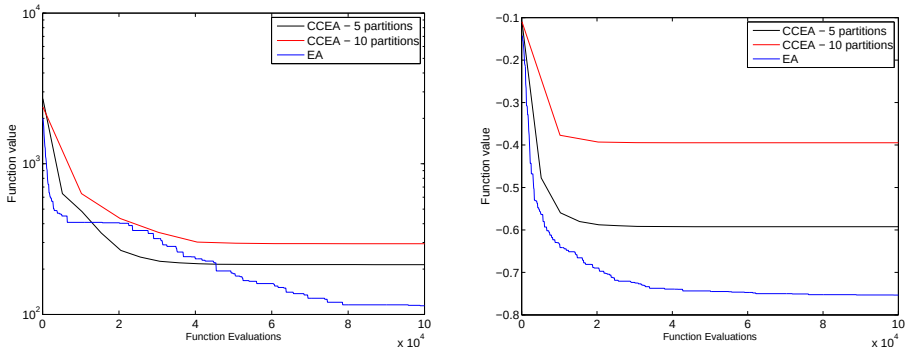
(c) Rosenbrock (d) Ackley

Figure 5.2: Convergence plot of the median run obtained using EA and CCEA (fixed length partitions) for 100D problems

To illustrate this, two non-separable problems are chosen. The first is the Shifted 50D rotated Rastrigin function from the IEEE CEC 2005 benchmark problem suite [129], and the second problem is the 50D G2 [55]. The problem definitions are given in Table 5.1. The performance of CCEA and EA on these two problems, using 100,000 evaluations, is shown in Table 5.5. It is seen that the EA outperforms the CCEA for both these test problems, which is just the opposite of what is observed for the benchmark functions studied earlier. Because of the strong interaction between the variables, random partitioning is ineffective for evolving the solutions individually. The median convergence plots for the problems are shown in Figure 5.3.

Table 5.5: Results using CCEA (fixed partitions) and EA for 50D Shifted Rotated Rastrigin and G2

Test Problem	Algorithm	Best	Mean	Worst	S.D.
Rastrigin-sr	Fixed 5	1.413440e+02	2.198485e+02	3.333840e+02	5.209684e+01
	Fixed 10	1.642592e+02	3.094703e+02	6.378410e+02	9.178733e+01
	EA	**7.684920e+01**	**1.168063e+02**	1.635920e+02	1.956704e+01
(-)G2	Fixed 5	-7.603430e-01	-6.009613e-01	-4.526570e-01	7.011863e-02
	Fixed 10	-5.199560e-01	-3.890296e-01	-2.560880e-01	6.062659e-02
	EA	**-7.847690e-01**	**-7.517443e-01**	-6.991770e-01	2.251734e-02



(a) Rastrigin-sr (b) (-)G2

Figure 5.3: Convergence of the median runs obtained using EA and CCEA for 50D problems (fixed length partitions)

The above examples make it clear that for some problems partitioning variables into non-overlapping populations and coevolving may work well while for others keeping all variables together as a single population EA works better. In the next section, CCEA-AVP is proposed, which exploits the dependence between variables to adaptively create partitions, resulting in good performance of the algorithm across a wide variety of test problems.

5.3 CCEA with Adaptive Variable Partitioning (CCEA-AVP)

For problems with significant variable interactions, the design vectors with high global function fitness are likely to exhibit certain mathematical relationships among the variables, as they tend to lie on objective/constraint boundaries. The proposed CCEA tries to derive benefit from this characteristic by capturing and employing correlations between the variables for their effective partitioning during the search. CCEA-AVP algorithm is outlined in Algorithm 5.2.

In the proposed CCEA algorithm, first of all, the solutions are evolved as done in a conventional EA for a few generations; after which the partitioning is invoked. An archive (*arc*) of all the solutions explored during the run is maintained. To partition the design space into smaller subsets, the following strategy is employed. Firstly, the solutions in *arc* are sorted based on their fitness function value. Thereafter, correlations among the variables are calculated based on the top 50% of the solutions in the sorted archive. For the presented studies, Pearson's correlation coefficient is used, whose magnitude varies between 0 and 1, 0 indicating no correlation, and 1 (or -1) indicating very strong correlation in the data. Once the pairwise correlation values are obtained for each variable ($D \times D$ matrix, where D is the number of variables), the *absolute* correlation values exceeding a cutoff value are identified and collected to form a partition. This process identifies disjoint sets with strong interdependence of variables within the partition and weak interdependence across partitions. Such a scheme for non-separable problems will create large block(s) of interdependent variables, thereby behaving similar to EA. On the other hand, if the problem is separable, the variable correlations will be considerably low, and the results closer to those

Algorithm 5.2 CCEA with correlation based Adaptive Variable Partitioning

Require: N_{EA}) (Number EA generations),N_G (Number of CCEA Generations), N_V (Number of variables for the problem), N_P (Population size), CR (Crossover Rate), MR (Mutation Rate), Distribution index of Crossover and Distribution index of Mutation.

1: Initialize Population (pop), Archive (arc)
2: Set $Soln_{best}$ as the best individual in the population
3: Run EA for a N_{EA} generations:
4: **for** $gen = 2$ to N_{EA} **do**
5: $childpop_{gen} \leftarrow$ Evolve (pop_{gen-1})
6: Evaluate ($childpop_{gen}$)
7: $P \leftarrow$ Sort ($childpop_{gen} + pop_{gen-1}$)
8: $pop_{gen} \leftarrow$ Reduce (P)
9: Update arc, $Soln_{best}$
10: **end for**
11: Invoke CCEA:
12: **for** $cgen = 2$ to N_G **do**
13: Determine number of partitions s based on correlation ($1 \leq s \leq S_{max}$){Refer Algorithm 5.3}
14: Split pop_{cgen} into s partitions $pop_{1,cgen}, pop_{2,cgen} \ldots pop_{s,cgen}$. Let the $\mathbf{V_j}$ denote the indices of design parameters contained in a partition j
15: **for** $i = 1$ to s **do**
16: Construct $spop_i$ by combining variable values in partition i with best/random variable values from other partitions
17: **for** $subgen = 2$ to N_{SG} **do**
18: $childpop_{i,subgen} \leftarrow$ Subevolve ($spop_{i,subgen-1}$)
19: Evaluate ($childpop_{i,subgen}$)
20: $CP \leftarrow$ Sort ($childpop_{i,subgen} + spop_{i,subgen-1}$)
21: $spop_{i,subgen} \leftarrow$ Reduce (CP)
22: **end for**
23: Set $pop_{cgen}(\mathbf{V_i}) = spop_i$
24: Update $Soln_{best}$ {Update best solution}
25: **end for**
26: Update arc
27: **end for**

obtained using uniform partitioning. Hence, the adaptive strategy tries to opt for the better option (whether to evolve the variables together or separately; and how to partition the variables), depending on the problem's behavior. The following cases may arise while partitioning the variables:

1. The total number of correlation based partitions and uncorrelated variables amounts to the exact number of maximum subsystems (S_{max}).

2. The total number of partitions based on the correlation (N_{Sc}) is less than S_{max}. In such a case, if the number of remaining variables (N_{Su}) is greater than the remaining N_r subsystems ($N_r = S_{max} - N_{Sc}$), the remaining variables are distributed in those N_r systems as uniformly as possible. On the other hand, if the number of remaining variables is less than the number of remaining subsets, then each remaining variable is assigned to an individual subset and the left-over subsets are discarded.

3. If the number of correlation based partitions (N_Sc) are greater than S_{max}, the excess partitions are forcibly merged into one partition so that the total number of partitions does not exceed S_{max}. This limit is imposed on partitioning because the presence of too many partitions will result in lower interaction frequency among the partitions, which is likely to result in an inferior performance of CCEA.

The details of the partitioning strategy are given in Algorithm 5.3. For CCEA-AVP, different partitions can contain different numbers of variables. Therefore, the number of *subevolve* generations for each partitions is not kept same. Each partition is allotted $2 \times n_s$ generations for evolution, where n_s is the number of variables in that partition. An upper bound on *subevolve* generations is imposed as 10, so that the interaction frequency (calls to the top-level CCEA) are not significantly reduced.

Algorithm 5.3 Partitioning strategy for CCEA-AVP

Require: arc (Archive of explored solutions), S_{max} (maximum number of subsystems), N_V (number of variables for the problem), Correlation Threshold T

A = Sort (arc) {Sort the solutions in terms of function finesses}
$N_a = |A|$
$C \leftarrow$ Correlation $(A(1 : N_a/2))$
for $i = 1$ to N_V **do**
  $S_i \leftarrow \{j | C_{i,j} \geq T\}$
end for
$Sc \leftarrow \{S_i \quad ; \quad |S_i| > 1\}$ (Sets of variables correlated with at least one other variable)
$Su \leftarrow \{S_i \quad ; \quad |S_i| = 1\}$ (Variables not correlated with any other variable)
while $\{ \ S_i \cap S_j \neq \phi \forall \{S_i, S_j\} \in Sc, i \neq j \ \}$ **do**
  if $S_i \cap S_j \neq \phi$ (where $S_i, S_j \in Sc$) **then**
    $S_{new} = S_i \cup S_j$ (Merge sets with common elements)
    $Sc \leftarrow$ Insert S_{new}
    $Sc \leftarrow$ Delete S_i, S_j
  end if
end while
N_{Sc} = Number of sets in Sc
if $N_{S_c} \geq S_{max}$ **then**
  $Sc_{S_{max-1}} \leftarrow \{Sc_{S_{max-1}} \cup Sc_{S_{max}} \cup \ldots \cup Sc_{N_{Sc}}\} \cup Su$
else
  $N_r = S_{max} - N_{Sc}$ (Number of subsystems left after assigning correlation based partitions)
  $Su \leftarrow$ Distribute S_u into N_r partitions uniformly.
  $S = \{Sc, Su\}$
end if

5.4 Numerical Experiments

Six test functions, listed in Table 5.1, are used to study the proposed algorithm. The first four (Rastrigin, Schwefel, Rosenbrock, Ackley) have relatively low variable interactions, where as the remaining two (Shifted Rotated Rastrigin and G2) are highly non-separable problems.

To study the performance of various algorithms on the aforementioned test functions, multiple (30) independent runs using each algorithm (for each combi-

nation of parameters) on all test functions are conducted. The parameters used for the EA and CCEA are listed in Table 5.6. For Rastrigin, Schwefel, Rosenbrock and Ackley functions, 50D and 100D problems are studied, whereas for the Shifted Rotated Rastrigin and G2 functions, only 50D problems are considered. Experiments are conducted using different numbers of partitions (number of fixed partitions for the basic CCEA and maximum partitions for the CCEA-AVP), and also with different values of the threshold for correlation, as shown in Table 5.6. For the CCEA-AVP, partitioning is invoked after 10000 function evaluations.

Table 5.6: Parameters used for the study

Parameter	Value(s)
Population Size	100
Maximum function evaluations	100000
Crossover probability	1.0
Mutation probability	0.1
Crossover distribution index	15
Mutation distribution index	20
Maximum partitions for CCEA-AVP	5,10
Correlation threshold T	0.3, 0.4, 0.5, 0.6

5.4.1 Results on 50D test problems

Listed in Table 5.7 are the results obtained using the basic CCEA with 10 partitions, CCEA-AVP with a maximum of 10 partitions, and the EA. It is seen that, for Rastrigin, Schwefel, Rosenbrock and Ackley functions, the best performance is achieved using the basic CCEA (fixed 10) partitions, which is expected due to relatively low interactions among the variables for these test functions. The results are closely followed by those obtained using CCEA-AVP. As expected, the results using both these CCEA strategies are much better than those obtained by evolving the solutions as a single population in the EA. On the other hand, for the remaining two functions (Shifted Rotated Rastrigin and G2), the standings of the algorithms are reversed; the EA obtains the best solutions,

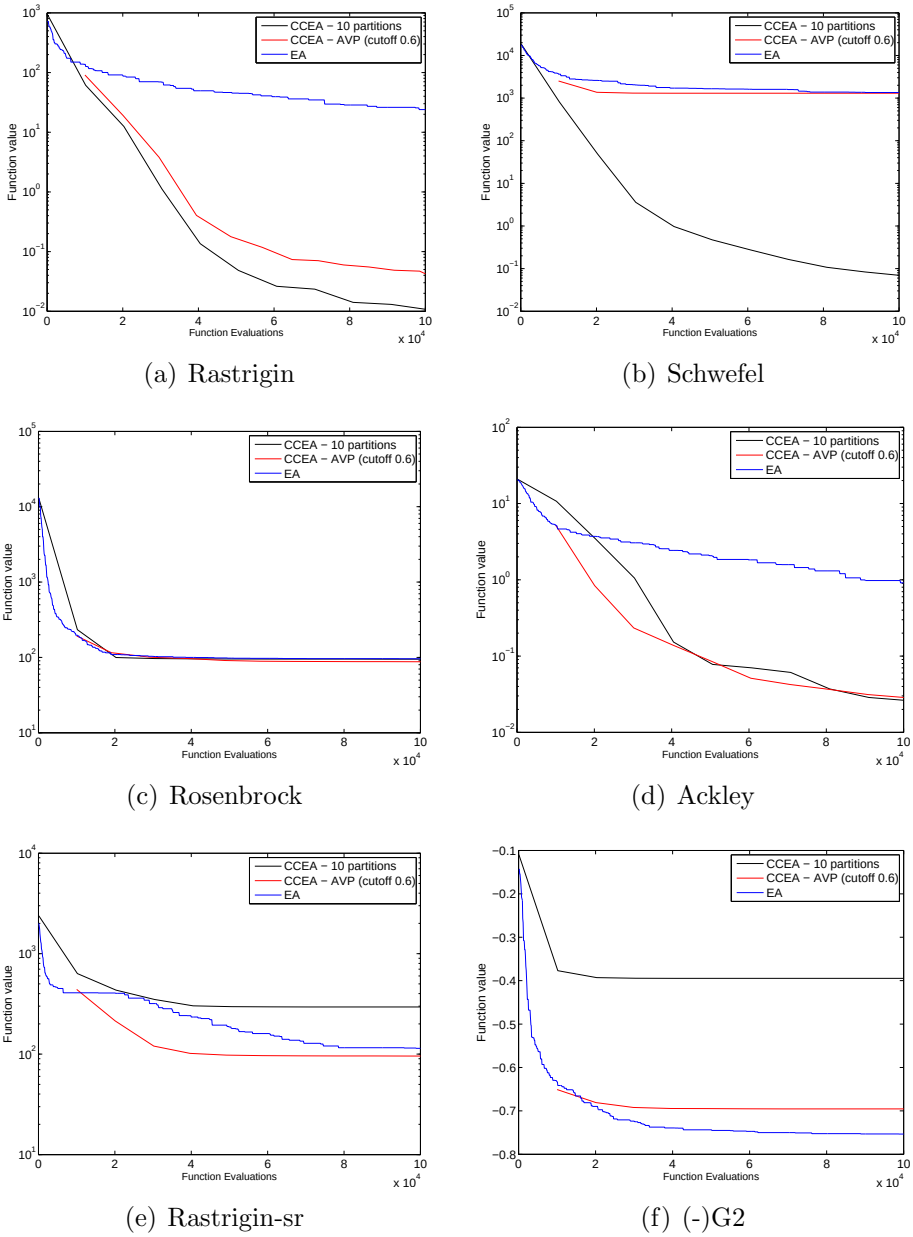
closely followed by CCEA-AVP. The performance of the basic CCEA for these two problems is inferior to the other two algorithms.

Table 5.7: Results using EA and CCEA (maximum 10 partitions) for 50D problems

Test Problem	Algorithm	Best	Mean	Worst	S.D.
Rastrigin	Fixed 10	**5.779232e-03**	**1.154832e-02**	2.345529e-02	4.235983e-03
	AVP(0.3)	1.836837e-01	2.154502e+00	6.299632e+00	1.555445e+00
	AVP(0.4)	3.544890e-02	6.113226e-01	2.716340e+00	6.822502e-01
	AVP(0.5)	1.410410e-02	1.495670e-01	1.102570e+00	2.673272e-01
	AVP(0.6)	1.057469e-02	6.634232e-02	1.015830e+00	1.802082e-01
	EA	1.662080e+01	2.367404e+01	2.875840e+01	3.018061e+00
Schwefel	Fixed 10	**4.113830e-02**	**1.586701e+01**	1.185799e+02	4.095543e+01
	AVP (0.3)	5.925140e+02	1.279976e+03	2.842700e+03	4.391927e+02
	AVP (0.4)	4.737855e+02	1.275286e+03	2.724200e+03	4.261194e+02
	AVP (0.5)	4.737964e+02	1.274598e+03	2.724150e+03	4.294707e+02
	AVP (0.6)	4.738365e+02	1.275259e+03	2.724160e+03	4.261098e+02
	EA	5.291690e+02	1.331623e+03	2.766410e+03	4.266604e+02
Rosenbrock	Fixed 10	**2.656768e+01**	9.651688e+01	1.540250e+02	3.973974e+01
	AVP (0.3)	4.068637e+01	8.381610e+01	1.562630e+02	3.228476e+01
	AVP (0.4)	4.017283e+01	8.319730e+01	1.609554e+02	3.519238e+01
	AVP (0.5)	4.007405e+01	7.699733e+01	1.515280e+02	3.135647e+01
	AVP (0.6)	3.911556e+01	**7.452101e+01**	1.518760e+02	2.923009e+01
	EA	4.204690e+01	8.385528e+01	1.892910e+02	3.590212e+01
Ackley	Fixed 10	1.804900e-02	**2.684102e-02**	4.221360e-02	5.015803e-03
	AVP (0.3)	2.287620e-02	6.374764e-02	1.453340e-01	3.483065e-02
	AVP (0.4)	1.609523e-02	3.154126e-02	4.651380e-02	7.028948e-03
	AVP (0.5)	**1.538815e-02**	2.633498e-02	3.463329e-02	5.177229e-03
	AVP (0.6)	1.590089e-02	2.692727e-02	3.425910e-02	5.114675e-03
	EA	5.964560e-01	9.039441e-01	1.185250e+00	1.374061e-01
Rastrigin-sr	Fixed 10	1.642592e+02	3.094703e+02	6.378410e+02	9.178733e+01
	AVP (0.3)	6.449470e+01	1.025146e+02	1.380228e+02	2.011653e+01
	AVP (0.4)	6.678959e+01	**9.939273e+01**	1.443156e+02	2.119351e+01
	AVP (0.5)	6.683000e+01	1.012824e+02	1.557940e+02	2.379061e+01
	AVP (0.6)	**6.673130e+01**	1.009536e+02	1.531503e+02	2.261265e+01
	EA	7.684920e+01	1.168063e+02	1.635920e+02	1.956704e+01
(-)G2	Fixed 10	-5.199560e-01	-3.890296e-01	-2.560880e-01	6.062659e-02
	AVP (0.3)	-7.492240e-01	-7.049272e-01	-6.323944e-01	3.067198e-02
	AVP (0.4)	-7.469640e-01	-7.005932e-01	-6.329717e-01	3.264801e-02
	AVP (0.5)	-7.648944e-01	-6.920226e-01	-6.269927e-01	3.363706e-02
	AVP (0.6)	-7.506297e-01	-6.901807e-01	-6.237227e-01	3.048877e-02
	EA	**-7.847690e-01**	**-7.517443e-01**	-6.991770e-01	2.251734e-02

The median convergence plots for the 50D test problems obtained using the three algorithms: CCEA-AVP with maximum 10 partitions, CCEA with fixed 10 partitions, and EA, are shown in Figure 5.4. It can be seen that for the Rastrigin, Schwefel, Rosenbrock and Ackley functions, the CCEA-AVP performs better than the EA, and in most cases is quite similar to the performance of the CCEA with fixed partitioning. On the other hand, for the Shifted Rotated Rastrigin and G2 functions, the convergence trend of the CCEA-AVP is similar to EA. Note that

in the figure, only the convergence plots corresponding to CCEA-AVP with a cutoff value of 0.6 are shown. Other cases (cutoff 0.3, 0.4 and 0.5) have similar trends, and are omitted for the sake of brevity.



(a) Rastrigin (b) Schwefel

(c) Rosenbrock (d) Ackley

(e) Rastrigin-sr (f) (-)G2

Figure 5.4: Convergence plots of the median runs for 50D problems; Comparison between CCEA with 10 fixed partitions, CCEA-AVP with maximum 10 partitions (cutoff 0.6), and EA

Next, the same set of results are reproduced using 5 fixed partitions in the basic CCEA, and 5 maximum partitions in the CCEA-AVP to observe the effect of partitioning size. The result are presented in Table 5.8 It is observed that while the relative standings of the algorithms remain unchanged for all the test problems, their converged values show difference from the 10 partition case. It is interesting to note that reducing the number of partitions for the CCEA from 10 to 5 *worsens* its performance for the first four problems (Rastrigin, Schwefel, Rosenbrock, Ackley), but *improves* it for the remaining two cases. This is understandable as, for separable functions, the greater the number of partitions, the easier it is to evolve the subpopulations owing to their smaller sizes. On the other hand, the results using the CCEA-AVP show very marginal variations between the two partitions size limits used. This is because the partitions are created adaptively and the maximum limit is imposed only if the number of adaptively created subsets exceeds the limit on the maximum number of subsets (S_{max}). Thus, the proposed CCEA-AVP is less sensitive to the number of maximum subsets allowed.

The convergence plots for performance on the 50D test problems using 5 partitions for the basic CCEA, a maximum of 5 partitions for the CCEA-AVP, and the EA, are shown in Figure 5.5. The plots reflect the same trends as seen in Table 5.8.

5.4.2 Results for 100D problems

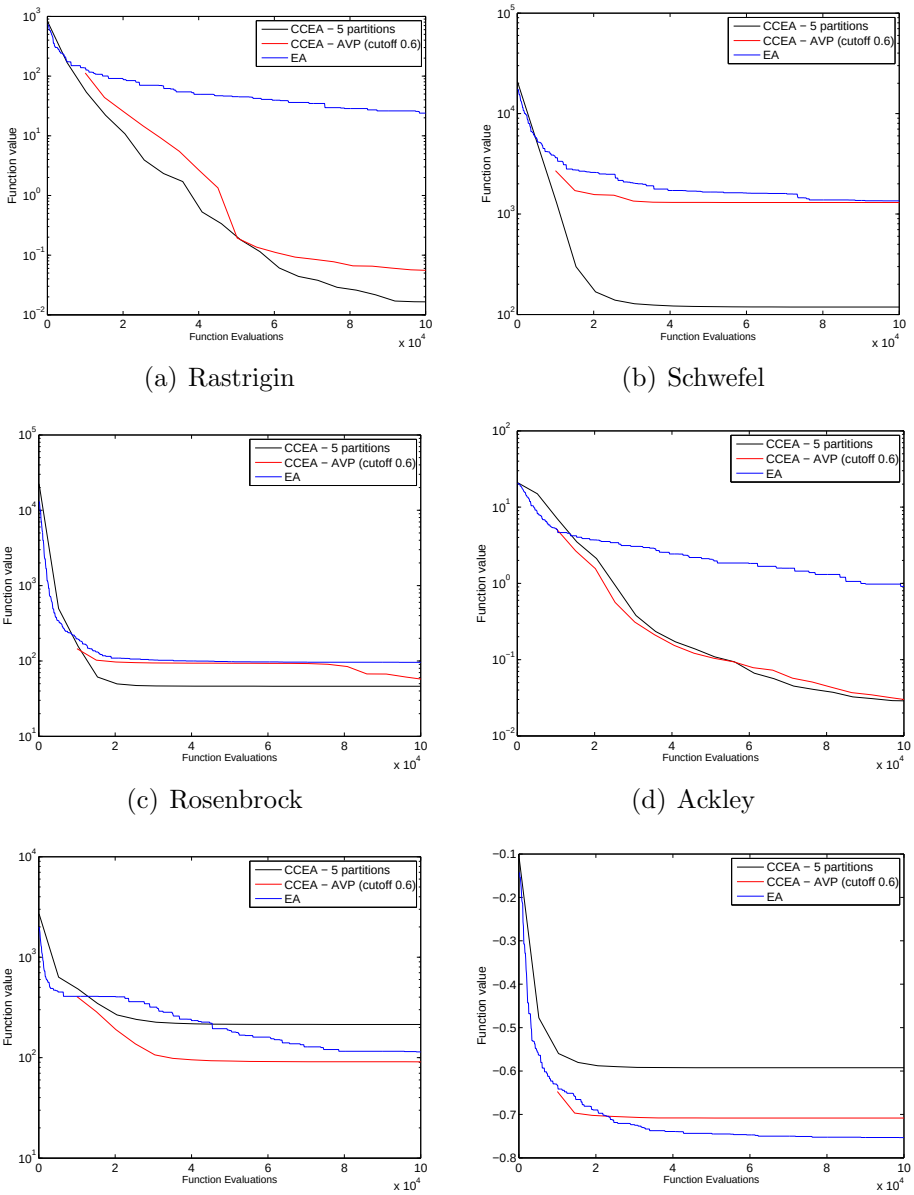
As an extension to the studies on 50D problems described in the previous subsection, additional studies are conducted on 100D test problems using the CCEA-AVP. This is done in order to establish that increasing the problem dimensionality does not adversely effect on the performance of the proposed CCEA-AVP. The

Table 5.8: Results using EA and CCEA (maximum 5 partitions) for 50D problems

Test Problem	Algorithm	Best	Mean	Worst	S.D.
Rastrigin	Fixed 5	**8.731096e-03**	**1.776637e-02**	3.059190e-02	6.328858e-03
	AVP (0.3)	2.783710e-01	1.930074e+00	6.417341e+00	1.522614e+00
	AVP (0.4)	3.544890e-02	8.509676e-01	3.351182e+00	8.333593e-01
	AVP (0.5)	3.077880e-02	2.718896e-01	1.345560e+00	3.599091e-01
	AVP (0.6)	1.421517e-02	7.525285e-02	2.209411e-01	5.821036e-02
	EA	1.662080e+01	2.367404e+01	2.875840e+01	3.018061e+00
Schwefel	Fixed 5	**4.353910e-02**	**1.895843e+02**	4.738343e+02	1.411670e+02
	AVP (0.3)	5.923904e+02	1.282985e+03	2.842700e+03	4.341656e+02
	AVP (0.4)	4.738339e+02	1.275315e+03	2.724200e+03	4.261159e+02
	AVP (0.5)	4.738473e+02	1.271342e+03	2.724230e+03	4.320383e+02
	AVP (0.6)	5.922526e+02	1.279239e+03	2.605790e+03	4.083936e+02
	EA	5.291690e+02	1.331623e+03	2.766410e+03	4.266604e+02
Rosenbrock	Fixed 5	**3.826361e+01**	**6.667871e+01**	1.513366e+02	2.989787e+01
	AVP (0.3)	4.126986e+01	8.421877e+01	1.759625e+02	3.512069e+01
	AVP (0.4)	4.067930e+01	8.041582e+01	1.524260e+02	3.016519e+01
	AVP (0.5)	3.981255e+01	7.538767e+01	1.519500e+02	2.948130e+01
	AVP (0.6)	3.968770e+01	7.478881e+01	1.516210e+02	3.098170e+01
	EA	4.204690e+01	8.385528e+01	1.892910e+02	3.590212e+01
Ackley	Fixed 5	2.178630e-02	3.020990e-02	4.191100e-02	5.403417e-03
	AVP (0.3)	2.287620e-02	5.546456e-02	1.453340e-01	3.171820e-02
	AVP (0.4)	2.418390e-02	3.286762e-02	4.651380e-02	6.782573e-03
	AVP (0.5)	2.298943e-02	3.100161e-02	4.219020e-02	5.266403e-03
	AVP (0.6)	**2.020320e-02**	**3.006383e-02**	3.911600e-02	4.950653e-03
	EA	5.964560e-01	9.039441e-01	1.185250e+00	1.374061e-01
Rastrigin-sr	Fixed 5	1.413440e+02	2.198485e+02	3.333840e+02	5.209684e+01
	AVP (0.3)	6.449470e+01	1.005986e+02	1.379169e+02	1.914945e+01
	AVP (0.4)	6.702810e+01	1.003611e+02	1.406950e+02	2.056791e+01
	AVP (0.5)	6.401050e+01	9.872883e+01	1.440974e+02	2.148299e+01
	AVP (0.6)	**6.384430e+01**	9.813578e+01	1.438673e+02	2.149457e+01
	EA	7.684920e+01	1.168063e+02	1.635920e+02	1.956704e+01
(-)G2	Fixed 5	-7.603430e-01	-6.009613e-01	-4.526570e-01	7.011863e-02
	AVP (0.3)	-7.760420e-01	-7.137156e-01	-6.425938e-01	2.665650e-02
	AVP (0.4)	-7.694752e-01	-7.102947e-01	-6.459791e-01	3.129272e-02
	AVP (0.5)	-7.718898e-01	-7.064999e-01	-6.478198e-01	3.122347e-02
	AVP (0.6)	-7.741146e-01	-7.063887e-01	-6.432024e-01	3.292225e-02
	EA	**-7.847690e-01**	**-7.517443e-01**	-6.991770e-01	2.251734e-02

100-variable Rastrigin, Schwefel, Rosenbrock and Ackley functions are used for this study. As for the 50D case, 30 independent runs are made using partition sizes of 5 and 10 for the basic CCEA and maximum partition sizes of 5 and 10 for CCEA-AVP. The parameters used are the same as those used for 50D studies, listed in Table 5.2.

The results obtained using the basic CCEA (10 partitions), the CCEA-AVP (maximum 10 partitions), and the EA for 100D test problems are shown in Table 5.9. The observations are consistent with those for 50D case – the basic CCEA performs best for these problems, closely followed by CCEA-AVP, whereas



(a) Rastrigin (b) Schwefel

(c) Rosenbrock (d) Ackley

(e) Rastrigin-sr (f) (-)G2

Figure 5.5: Convergence plots of the median runs for 50D problems; Comparison between CCEA with 5 fixed partitions, CCEA-AVP with maximum 5 partitions (cutoff 0.6), and EA

the performance of the EA is significantly inferior to that of these two algorithms. The corresponding convergence plots are shown in Figure 5.6.

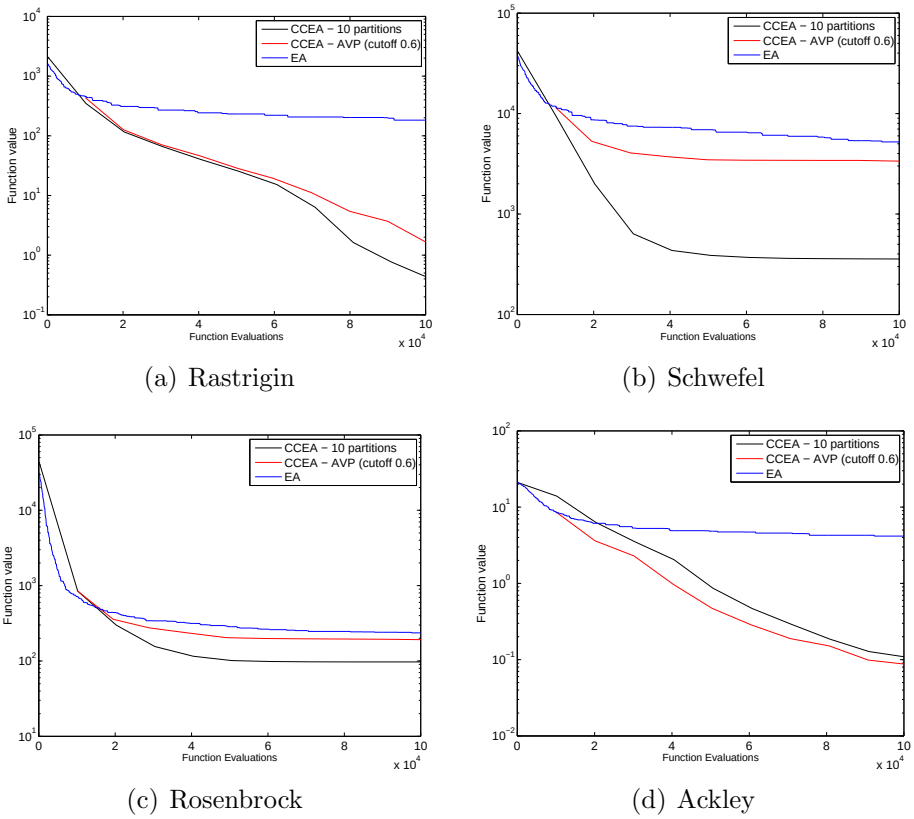
Finally, the same set of studies for 100D problems are repeated with 5 par-

titions for the CCEA and a maximum of 5 partitions for the CCEA-AVP. The average converged values after 100000 evaluations are shown in Table 5.10, and the corresponding convergence plots in Figure 5.7. Once again, it is seen that while the CCEA performance consistently degrades for all four test problems when the number of partitions is reduced, the CCEA-AVP performance is not significantly different from the maximum 10-partition case.

Table 5.9: Results using EA and CCEA (maximum 10 partitions) for 100D problems

Test Problem	Algorithm	Best	Mean	Worst	S.D.
Rastrigin	Fixed 10	**2.646480e-01**	**5.665662e-01**	2.433088e+00	5.157489e-01
	AVP (0.3)	1.681521e+01	3.827224e+01	6.670020e+01	1.335660e+01
	AVP (0.4)	8.706855e-01	1.669110e+01	3.341113e+01	8.153672e+00
	AVP (0.5)	1.483221e+00	4.623035e+00	1.321691e+01	3.104135e+00
	AVP (0.6)	3.742920e-01	2.156513e+00	7.478845e+00	1.860296e+00
	EA	1.649800e+02	1.840125e+02	2.036640e+02	1.209635e+01
Schwefel	Fixed 10	**1.507234e+00**	3.610878e+02	8.309508e+02	2.072720e+02
	AVP (0.3)	2.573690e+03	3.853914e+03	4.802564e+03	5.113526e+02
	AVP (0.4)	2.643870e+03	3.310721e+03	4.094800e+03	3.945394e+02
	AVP (0.5)	2.374810e+03	3.212740e+03	4.030107e+03	4.004432e+02
	AVP (0.6)	2.253450e+03	3.195682e+03	4.028548e+03	4.042763e+02
	EA	4.346710e+03	5.280894e+03	6.328550e+03	5.714214e+02
Rosenbrock	Fixed 10	**8.771310e+01**	**1.311911e+02**	3.167990e+02	4.927446e+01
	AVP (0.3)	1.468420e+02	2.487470e+02	3.439314e+02	5.251006e+01
	AVP (0.4)	1.386270e+02	2.085760e+02	2.792287e+02	4.088388e+01
	AVP (0.5)	9.676000e+01	1.870135e+02	2.897350e+02	4.469083e+01
	AVP (0.6)	9.506310e+01	1.828531e+02	2.689460e+02	4.757594e+01
	EA	1.259630e+02	2.385880e+02	3.612080e+02	5.010407e+01
Ackley	Fixed 10	8.540569e-02	1.083423e-01	1.413097e-01	1.431065e-02
	AVP (0.3)	9.503202e-01	1.545042e+00	2.208593e+00	4.023391e-01
	AVP (0.4)	1.032025e-01	3.802399e-01	1.544246e+00	2.809987e-01
	AVP (0.5)	6.707651e-02	**8.916227e-02**	1.907402e-01	2.303647e-02
	AVP (0.6)	**6.798911e-02**	8.959588e-02	1.220725e-01	1.318414e-02
	EA	3.975440e+00	4.172588e+00	4.423220e+00	9.634730e-02

This study clearly shows that a fixed partitioning scheme as used in the CCEA can lead to significantly poor performance for certain classes of problems while it may be useful for others. The CCEA-AVP is designed to be applicable as a generic optimizer with improved mean performance when compared to those of the EA and CCEA over a wide range of problems.



(a) Rastrigin (b) Schwefel

(c) Rosenbrock (d) Ackley

Figure 5.6: Convergence plots of the median runs for 100D problems; Comparison between CCEA with 10 fixed partitions, CCEA-AVP with maximum 10 partitions (cutoff 0.6), and EA

5.4.3 Variation in performance of CCEA-AVP with different values of the Correlation Threshold

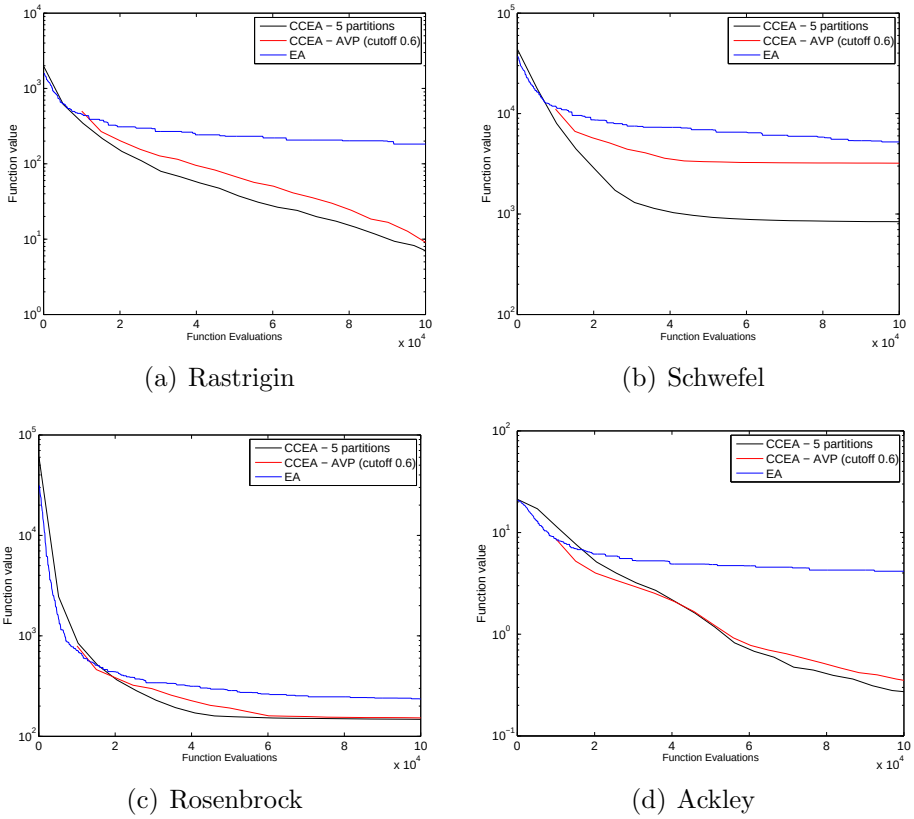
In the preceding subsections, studies using different values of correlation thresholds in order to assign the variables together into one partition are presented. Although all four threshold values (0.3, 0.4, 0.5 and 0.6) have worked well for the given problems, there are slight variations in the performance introduced by the cutoff value. The relative standings of the CCEA-AVP using the above four cutoff values, for the 50D maximum 10 partitions case, are shown in Figure 5.8. It is observed that the 0.6 correlation value works best for most of

Table 5.10: Results using EA and CCEA (maximum 5 partitions) for 100D problems

Test Problem	Algorithm	Best	Mean	Worst	S.D.
Rastrigin	Fixed 5	**2.759770e+00**	**6.940710e+00**	1.332520e+01	2.324321e+00
	AVP (0.3)	1.961420e+01	4.171061e+01	6.578003e+01	1.164476e+01
	AVP (0.4)	1.012750e+01	2.176104e+01	3.822304e+01	6.795122e+00
	AVP (0.5)	5.136025e+00	1.198904e+01	2.377627e+01	4.137389e+00
	AVP (0.6)	3.562711e+00	8.626150e+00	1.334070e+01	2.392076e+00
	EA	1.649800e+02	1.840125e+02	2.036640e+02	1.209635e+01
Schwefel	Fixed 5	**3.653240e+02**	**9.442860e+02**	1.550470e+03	3.218071e+02
	AVP (0.3)	2.573690e+03	3.705760e+03	4.945437e+03	5.973603e+02
	AVP (0.4)	2.568155e+03	3.282633e+03	4.057032e+03	4.499224e+02
	AVP (0.5)	2.507350e+03	3.249588e+03	4.038401e+03	4.626884e+02
	AVP (0.6)	2.384250e+03	3.201359e+03	4.039300e+03	4.548267e+02
	EA	4.346710e+03	5.280894e+03	6.328550e+03	5.714214e+02
Rosenbrock	Fixed 5	**8.440033e+01**	**1.472800e+02**	2.613990e+02	4.064567e+01
	AVP (0.3)	1.217372e+02	2.278973e+02	3.412558e+02	4.463483e+01
	AVP (0.4)	1.107537e+02	1.926592e+02	2.588505e+02	3.599144e+01
	AVP (0.5)	1.004080e+02	1.695221e+02	2.169891e+02	3.484527e+01
	AVP (0.6)	8.913882e+01	1.619662e+02	2.436310e+02	3.954275e+01
	EA	1.259630e+02	2.385880e+02	3.612080e+02	5.010407e+01
Ackley	Fixed 5	2.263960e-01	2.689068e-01	3.276250e-01	2.660889e-02
	AVP (0.3)	5.068778e-01	**1.353864e+00**	2.170572e+00	3.949441e-01
	AVP (0.4)	**1.914989e-01**	3.516999e-01	8.454164e-01	1.451268e-01
	AVP (0.5)	2.387114e-01	2.975644e-01	4.632740e-01	4.476273e-02
	AVP (0.6)	2.489790e-01	3.408935e-01	4.818140e-01	5.677084e-02
	EA	3.975440e+00	4.172588e+00	4.423220e+00	9.634730e-02

the problems (although for Schwefel and G2 its performance is worse compared with other cases, but a look into the mean values reveal that the difference in performance is negligible for these two cases across all cutoff values).

Although it may not be a straightforward choice to determine the best cutoff value for any problem, empirically it is evident that if a too low a cutoff-value ($\approx$ below 0.3) is used, then even the weakly correlated variables will be evolved in the same partition, and the algorithm's performance will be roughly equivalent to that of an EA. At the other extreme, if a too high cutoff-value is used, even strongly correlated variables might end up in different partitions, which will result in the performance of the algorithm being like that of the basic CCEA. Therefore, the cutoff value should be reasonably chosen in order to avoid both extremes and achieve a balance among interacting variable partitions. Based on the experiments conducted in this chapter, a value of 0.4-0.6 should work



(a) Rastrigin (b) Schwefel

(c) Rosenbrock (d) Ackley

Figure 5.7: Convergence plots of the median runs for 100D problems; Comparison between CCEA with 5 fixed partitions, CCEA-AVP with maximum 5 partitions (cutoff 0.6), and EA

reasonably well for a broad variety of problems.

5.5 Summary

This chapter investigates the performance of a single best collaboration strategy CCEA for a set of benchmark test functions The results suggest that while the CCEA shows definite promise for solving large scale optimization problems, it is sensitive to a number of factors, in particular the partitioning scheme. Studies suggest that the basic CCEA with random partitioning may not be suitable for non-separable problems, and in such cases, evolving the variables together as in

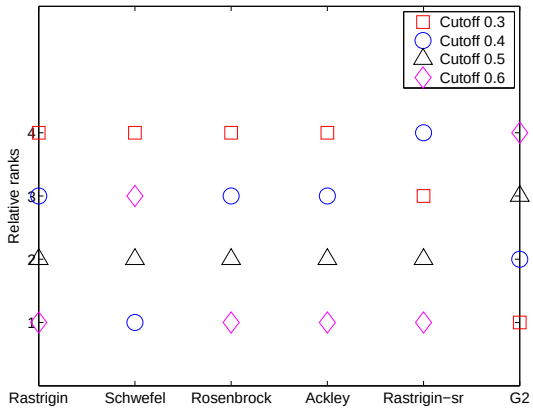


Figure 5.8: The performance of CCEA-AVP with different Correlation Threshold values used in order to assign the variables into the same partition.

an EA might be more beneficial.

Drawing motivation from the aforementioned observations, a novel CCEA algorithm with adaptive variable partitioning (CCEA-AVP), as an extension of Ray and Yao's earlier work [130] is proposed. This algorithm identifies correlations among the variables in each generation and partitions the strongly correlated variables together. Such a scheme preserves the advantages of both, the basic CCEA and an EA. This is because the modularity of the problem at hand is not compromised while creating partitions. If the problem exhibits sufficient separability, the partitioning is similar to that of the basic CCEA, whereas if the variables are highly interdependent, the algorithm behaves similar to an EA. This adaption ensures that the proposed algorithm performs well for both separable and non-separable problems.

Studies are conducted on six benchmark functions to test the efficacy of the proposed algorithm and the results obtained are encouraging. On an average, for separable problems, the algorithm obtains results comparable with those obtained using the basic CCEA, whereas for non-separable problems, it obtains results

comparable with those from the EA. Since the separability of a problem may often not be know a priori, the proposed algorithm demonstrates a wider applicability as a generic optimization tool. Additionally, it is also observed that CCEA-AVP tends to be less sensitive to changes in the number of (maximum) partitions, as it only uses them as an upper bound, and otherwise, the partitions are created based on correlations.

Chapter 6

Trans-dimensional Optimization

Abstract

Trans-dimensional Optimization (TDO) refers to optimization problems in which there exist a number of candidate models that may be used to evaluate the objective(s). In such problems, finding the best model as well as the corresponding variable values is of interest. In this chapter, some exploratory studies are conducted on trans-dimensional optimization. Preliminary experiments are presented using one proposed approach to deal with such problems.

6.1 Introduction

In the problems considered so far in this thesis, the number of variables and the physical quantities they represent have been considered to be fixed. In other words, optimization is carried out on a given *fixed model*. However, there are practical problems for which this may not be the case: there may be multiple possible models for a problem, with none being obviously preferred. In such

a case, the optimization process involves finding the best model as well as the corresponding optimum design variable values. Such problems are referred to as *Trans-dimensional* Optimization (TDO) problems. While studies on optimization for fixed models are abundant in the literature, very few studies have been conducted on TDO problems. In this chapter, exploratory studies are presented on trans-dimensional optimization. The studies, though preliminary, try to emphasize the importance of the field and present one possible approach that can be pursued towards designing efficient algorithms for handling such problems.

A simple example of a TDO problem is finding the optimal clustering of a given data set. While clustering a given data set, finding both the *number* of clusters and the *locations* of cluster centroids are of interest. The number of clusters determine the number of variables in the problem, because the corresponding optimum centroid locations have to be searched for. For example, if the data is two dimensional, for K centroids, there will be a total of $2K$ variables involved ($\{x_i, y_i\}, i = 1, 2, \ldots K$) in the search, where (x_i, y_i) denotes the centroid of the i^{th} cluster. If the locations of the centroids are searched using a conventional optimizer, the number of clusters has to be specified to the algorithm beforehand. The optimization has to be carried out using different possible numbers of clusters (different *models*) in order to finally determine the globally optimum solution across all models.

Similar cases of TDO may arise for a number of real life problems; for example, finding the optimum number and locations of warehouses to minimize the cost of a goods-transport network; identification of the number of layers, their thicknesses and constituents for a composite laminate design (MOC problem [131]); and optimizing the configuration of a vehicle during conceptual design.

Very few attempts have been made so far to deal with such problems efficiently. Since algorithms exist for solving problems with a fixed model, a round-about way is to exhaustively solve the problem for each possible model in order to find out which one is the optimum. However, this may be very inefficient as one may waste a considerable amount of computational effort on evaluating the *sub-optimal* models.

One way to handle TDO problems is to use variable length chromosomes for evolutionary algorithms (EAs). Such an approach necessitates defining operators that can delete or duplicate genes in order to represent the appropriate dimensionality of a solution [132, 133]. These operators need a careful choice of which genes could be recombined or inserted, and if so, what values should be chosen for the inserted genes. Maintaining a variable length chromosome can be particularly difficult if the variables for two different models represent entirely different physical quantities in the variable space.

To overcome the limitations of variable chromosome representation, an alternative approach was proposed in [134]. The idea is to use a representation with a fixed length vector together with a binary string which contains information about reducing the number of variables to the dimensionality of the solution. A mapping is made from the fixed-dimensional representation using the binary string to find variable values for the solution to be evaluated. Although this method was successfully applied to solve two problems (the MOC and ARMA modeling problems) [135], it was observed that such representation leads to variable bias towards medium-dimensional solutions because of the higher probability of having almost equal numbers of zeros and ones in the binary string.

Another approach for model selection, called Reversible Jump Markov Chain Monte Carlo (RJMCMC) was developed by Green [136]. RJMCMC traverses dif-

ferent dimensions by using reversible moves such as *birth, death, split* and *merge*. It has been used predominantly in statistics for various inverse problems, such as variable selection in regression [137], mixture deconvolution [138], multi-point change problems, image segmentation [136], clustering [139] etc. However, similar to the gene insertion/duplication process discussed earlier for evolutionary algorithms, constructive definitions for the birth/death/merge/split moves for the RJMCMC can be difficult for a number of problems.

In this chapter, a new approach, implemented in simulated annealing (SA), is introduced to deal with trans-dimensional problems. As opposed to the neighborhood schemes earlier employed [139], which explored the models using constructive moves, this approach evolves all models simultaneously and, at each iteration, chooses a model to explore based on its *fitness* value. At the same time, the model variables are evolved using exploratory moves within the model. This approach is adopted for three main reasons:

1. The algorithm can be run without designing specific operators for each problem to be solved.

2. Since the models are selected at each iteration from all the possible models based on their performance, there is no need for deciding on "large" or "small" jump from one model to another in order to improve the objective value.

3. The proposed approach can be easily extended to counter various characteristics of engineering problems, such as highly non-linear objectives, constraints, and the presence of multiple objectives.

Remainder of this chapter is organized as follows. The definition of a single-objective TDO problem is given in Section 6.2. The proposed algorithm is

introduced in Section 6.3, which is followed by description of the numerical experiments in Section 6.4. The findings of the presented studies are summarized in Section 6.5.

6.2 Trans-dimensional Optimization Problems

A generic single-objective TDO problem can be defined as follows. Consider an objective function $f(\mathbf{x}_{\mu_i})$, which can be evaluated using candidate solutions which have different numbers of variables (referred to as different models here). A set of all possible models for the problem is represented by $\mu = \{\mu_1, \mu_2, \ldots \mu_{n_{models}}\}$. Each model μ_i has a set of variables $\mathbf{x}_{\mu_i}$ associated with it. For example, to design a cantilever beam for a given application, two possible models could be: a circular cross section (with radius as a variable), and rectangular cross section (with height and breadth as variables). At the same time, there may be some common variables, such as length of the beam and thickness of the wall. If the variable space of a model μ_i is denoted by S_i, the aim is to find the model μ_i^* and the corresponding variable values $\mathbf{x^*}_{\mu_i} \in S_i$ that optimize the objective function $f(\mathbf{x}_{\mu_i})$.

6.3 SA based Trans-dimensional Optimization (SA-TDO)

As discussed in Chapter 2, SA [12] is an optimization algorithm which emulates the behavior of hot metal atoms subjected to slow cooling. In the work presented here, the traditional SA is extended to enable it to traverse the model space along with the variable space. The resulting algorithm is referred to as Simu-

lated Annealing for Trans-dimensional Optimization (SA-TDO). In this study, it is assumed that the number of possible candidate models are enumerable and known. During the search, at each iteration, the algorithm first chooses a model to explore based on the relative performance of the models until that point in the search. The models that show good objective values are selected more often in subsequent iterations whereas those with inferior performances are explored less often. This way, *elite* models are preferred during the search in the same way as elite solutions are preferred in evolutionary search algorithms. In addition, a provision is included in the proposed algorithm whereby a random model is evaluated occasionally , in order to prevent the algorithm from prematurely converging to a model which might eventually turn out to be sub-optimal. Once the model to be explored is chosen, its variable space is searched for a certain number of trials (epoch length). Epoch length in this study is decided based on the number of variables in the chosen model. The proposed approach is outlined in Algorithm 6.1, while a flowchart of the algorithm is given in Figure 6.1 for easier visualization. The details of the various steps involved in the proposed algorithm are discussed in the following subsections.

6.3.1 Calculation of initial and final temperatures

In SA-TDO, the initial and final temperatures are calculated using the method suggested in [69]. To calculate the initial temperature (T_{max}), a certain number (100 in presented studies) of random solutions are generated (with all models considered), and the maximum (f_{max}) and minimum (f_{min}) function values are identified. T_{max} is then calculated using the equation

$$P_i = e^{-((f_{max}-f_{min})/T_{max})},$$

Algorithm 6.1 SA-TDO algorithm

Require: P_i, P_f, α, μ
Calculate T_{max}, T_{min}, N
Initialize *Archive*
Set $T = T_{max}$
for $i = 1$ to N **do**
  if rand[0 1] $\leq p$ **then**
    Choose model $\mu_i \in \mu$ based on fitness
  else
    Choose a random model $\mu_i \in \mu$
  end if
  Calculate epoch length M
  for $j = 1$ to M **do**
    Set $\mathbf{x_{old}}$ = Best solution for μ_i found so far
    $\mathbf{x_{new}}$ = perturb($\mathbf{x_{old}}$)
    if $f_{\text{new}} \leq f_{\text{old}}$ **then**
      Set $\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}}$
    else
      $prob = \exp(-(f_{new} - f_{old})/T)$
      Set $\mathbf{x}_{\text{new}} = \mathbf{x}_{\text{old}}$ with a probability *prob*
    end if
    Update *Archive*
  end for
  Set $T = T \times \alpha$
end for

where P_i denotes the initial probability of jump, and is given as an input parameter. Usually a high value of P_i is chosen so as to allow for sufficient exploration of the search space during the initial stages.

The final temperature (T_{min}) is calculated using the equation

$$P_f = e^{-(f_{tol}/T_{min})},$$

where P_f is the probability of a jump in the final iteration, and f_{tol} the desired accuracy of the objective value being minimized. Generally, a low value of probability P_f is chosen so that towards the end of the run, when the algorithm

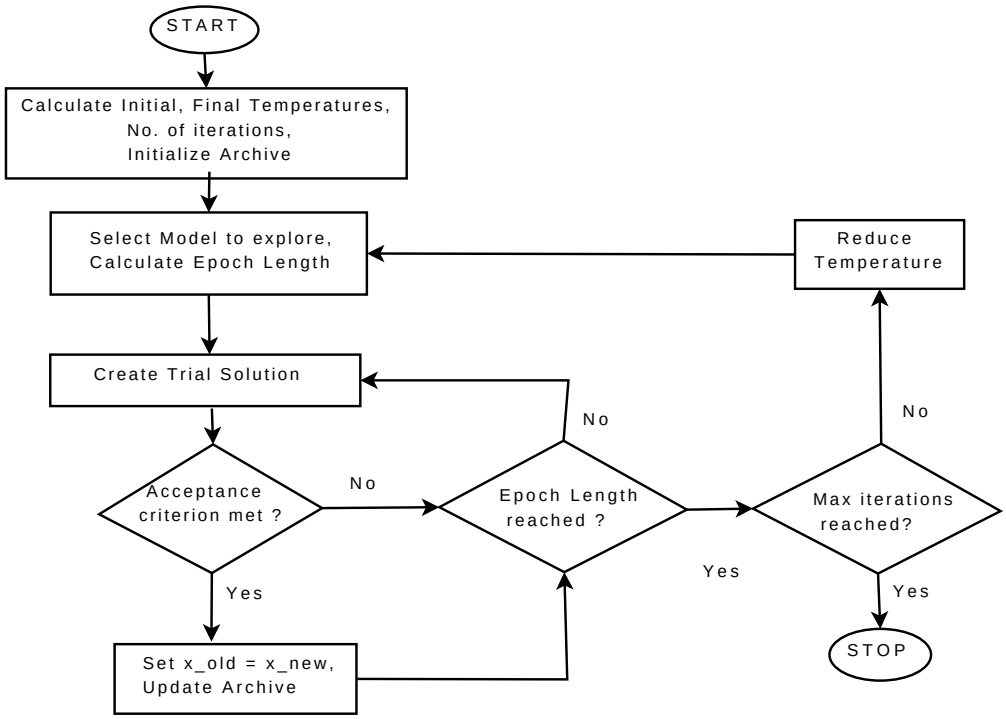


Figure 6.1: Simulated annealing for trans-dimensional optimization (SA-TDO)

is expected to have located an optimal region, it can perform a fine local search and convergence is not delayed due to the acceptance of poor solutions.

6.3.2 Calculation of number of iterations

Once the initial and final temperatures are determined, the number of iterations can be calculated based on the annealing schedule. A very simple and widely used annealing scheme is the exponential schedule, in which the temperature is reduced as

$$T_{i+1} = \alpha \times T_i,$$

where T_{i+1} and T_i represent temperatures at the $(i+1)^{th}$ and i^{th} iteration respectively, and α (< 1) is the annealing ratio which determines the rate of reduction of temperature; the higher the ratio, the faster the cooling.

With an exponential schedule, the number of iterations N can be calculated using

$$T_{min} = T_{max} \times \alpha^{N-1}.$$

6.3.3 Archive

All the trial solutions accepted during the search are stored in an archive; which consists of "bins" corresponding to each model. Whenever a solution is accepted, the archive is updated by storing the accepted solution into its respective bin. The archive is initialized at the beginning of the algorithm with one random solution belonging to each model.

6.3.4 Model selection

At each iteration, the *fitness* (performance based on the search so far) of each model is calculated by averaging the best 20% of the solutions in its bin in the archive. The fitness value of a model i (μ_i) is denoted here by F_i, where $i \in \{1, 2, \ldots n_{models}\}$. The fitness values are then normalized using

$$F_{i,norm} = \frac{max(F) - F_i}{max(F) - min(F)}$$

Hence, the model with better (*lower*) function value so far in the search gets a *higher* normalized fitness value (a minimization problem is assumed).

With a (predefined) probability p, a model to be explored is selected based on the roulette wheel of the normalized fitness values. Otherwise (with a probability of (1-p)), a random model is selected. To prefer the fitness-based selection, a high value of p ($\approx 0.7 - 0.9$) is recommended. This scheme tries to achieve model selection such that (a) the model(s) with better function values, (which

are better contenders for the global optimum), are preferred during the search; and (b) even the inferior models are explored with a low probability.

6.3.5 Model exploration

Once the model to be explored is selected, the epoch length M is calculated. Epoch length represents the number of trials made at each iteration (within an epoch length, the temperature is kept constant). For trans-dimensional problems, as different models can have a different number of variables, epoch length is calculated here as

$$M = L \times n_{\text{var}}$$

where L is a user-defined proportionality constant and n_{var} represents the number of variables in the chosen model. The epoch length is calculated this way in order to deal with the complexity of the models. Those with higher numbers of decision variables are allowed proportionately higher numbers of evaluations during an iteration.

In each iteration, the search starts with the *best solution of the chosen model* found so far. During the search, a trial solution is generated from the current solution using Laplacian perturbation, as suggested in [64]. One decision variable is chosen at random and perturbed by a random variable ϵ drawn from the Laplacian distribution $p(\epsilon) \propto e^{-\|\sigma\epsilon\|}$, where σ represents the spread of the perturbation. For the presented studies, the value of σ is set to 0.1.

6.3.6 Acceptance criteria

The acceptance criterion used here is the same as those in conventional SA for a minimization problem, which is as follows:

- if $f_{new} \leq f_{old}$, then accept the trial solution,
- else, accept the trial solution with a probability $exp\big(-\frac{f_{new}-f_{old}}{T}\big)$.

6.4 Numerical Experiments

Two simple problems are formulated to demonstrate the working of the proposed algorithm. The first one is a clustering problem, in which a given data set is to be grouped into an optimum number of clusters, and the second is a transport network cost minimization problem with a set of possible warehouse locations. These test problems, along with numerical experiments are discussed in the following sub-sections. To assess the performance of the SA-TDO algorithm, the proposed problems are also solved using the conventional optimization algorithm NSGA-II [9] for each model individually, and the results obtained from both algorithms are compared. The method of comparison is as follows. Firstly, the problem is solved using the SA-TDO algorithm. The total number of function evaluations used by SA-TDO are noted. Then, given the same number of evaluations (for a fair comparison), if the same problem is to be solved with a conventional algorithm, one is likely to assign the budget of function evaluations evenly among all the candidate models. The same is done here, *i.e.*, for each model, NSGA-II is allowed to run for the average total number of evaluations used divided by the total number of candidate models. Thereafter, the quality of solutions obtained using both the algorithms is compared using the obtained objective values.

6.4.1 Clustering problem

Clustering is a process of grouping a given data set based on a similarity measure, for example the Euclidean distance between them. Various methods have been suggested in the literature for clustering [140, 141]. The *goodness* of clustering can be measured using various cluster validity indices available in the literature, such as the Davis Bouldin (DB) [142], Xie-Beni [143] Dunn [144], Calinski-Harabasz [145], PBM [146], etc.

The problem considered here is as follows: A data set of 100 points is sampled using a Gaussian distribution around 5 selected points as centroids. The coordinates of the centroids are (2,2), (2,8), (5,5), (8,2) and (8,8). Around each centroid, 20 points are created by perturbing the center's coordinates using a random number from a normal distribution with a mean of zero and a standard deviation of 0.8. All the points are confined within the space $(x, y) \in [0, 10]^2$. A plot of the data set is shown in Figure 6.2. The Xie Beni (XB) [143] index is used as the cluster validity index to be optimized (minimized). The lower the value of the XB index, the better the clustering.

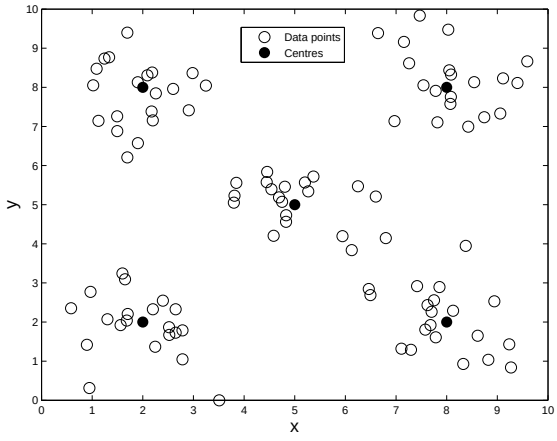


Figure 6.2: Sample data for clustering

In this study, the number of clusters is chosen between 3 and 7, resulting in 5 models: $\mu_1 \equiv 3$ clusters, $\mu_2 \equiv 4$ clusters, $\mu_3 \equiv 5$ clusters, $\mu_4 \equiv 6$ clusters and $\mu_5 \equiv 7$ clusters. A model with K clusters has $2K$ variables, *i.e.*, the cluster centroid coordinates (x_i,y_i), $i = 1 \ldots K$. "Hard clustering" is assumed, which means a point can belong to only one cluster. Assuming a perfect random number generator, since the points are generated using 5 centers within a small radius, the optimum number of clusters should be 5, with the centroids very close to the points used for generating the data.

The above clustering problem is first solved using the proposed SA-TDO algorithm. The parameters used for the algorithm are: $P_i = 0.9$, $P_f = 0.01$, $L = 15$ and $\alpha = 0.92$. The probability of choosing a model based on fitness (p) is set to 0.8. Multiple (30) independent runs are made varying the random seed. The average total number of evaluations used by the SA-TDO algorithm across all runs is 22,964. Hence, NSGA-II is run for $22964/5 \approx 4600$ evaluations for each model so that the total number of evaluations used by both algorithms is approximately the same. For NSGA-II, the population size used is 100 (evolved over 46 generations). The probabilities of crossover and mutation are set to 0.9 and 0.1 respectively. A crossover index of 15 and mutation index of 20 is used.

The results for the clustering problem from the SA-TDO and NSGA-II algorithms are shown in Table 6.1. The best value of the XB index corresponds to the model with 5 centers (μ_3), using both the algorithms. Hence, the optimum model is correctly identified using both algorithms. The search for the optimum model (and corresponding solution) in a typical SA-TDO run is depicted in Figure 6.3(a), which shows the best XB index values obtained during the search for each model, along with the total number of evaluations used. It can be seen how the best value of the objective (XB index) improves over the iterations, with

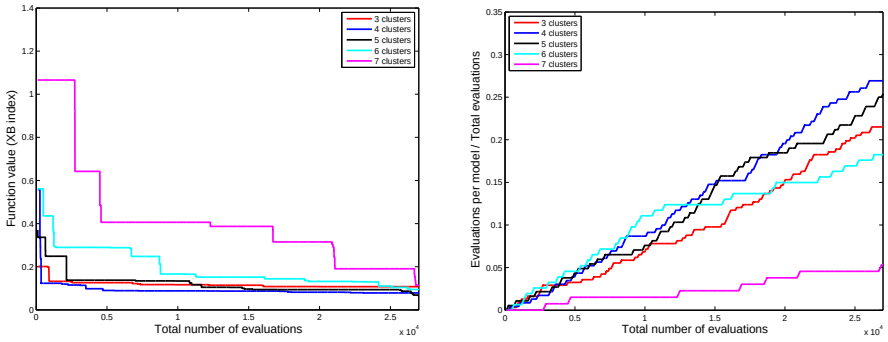
model μ_3 finally outperforming the other models. Also, it can be noted that the objective values corresponding to models with three to six clusters ($\mu_1 - \mu_4$) are fairly close *during* the search. Comparatively, the model with seven clusters (μ_5) shows significantly worse values throughout the run. It follows that most of the exploration during the search is done within the first four models ($\mu_1 - \mu_4$), whereas the model μ_5 is assigned very few evaluations owing to its comparatively poor performance. The number of evaluations assigned to different models during the run are shown in Figure 6.3(b).

It is to be noted here that since the assignment of function evaluations is fitness-based, the SA-TDO may face the problem of insufficient convergence (in terms of objective values) of the optimum model if the candidate models have closely comparable fitness. For example, in this case, the average XB values corresponding to μ_3 obtained using the SA-TDO algorithm are inferior to those obtained using the NSGA-II, as can be seen in Table 6.1. The statistical significance of the results is tested using the Mann-Whitney U test [147, 148]. The U and z values obtained for the results (corresponding to μ_3) using the SA-TDO and NSGA-II algorithms are $U = 134$ and $z = 4.6$, indicating that the difference between the two data sets is statistically significant. The reason for the inferior performance of the SA-TDO algorithm may be that, since the models are closely competing, even the non-optimal models have good probabilities of being selected during the search. Owing to this, in some runs, the non-optimum models may receive undesirably higher numbers of evaluations than the optimum model which could not converge sufficiently due to the inadequate number of evaluations assigned to it. In the worst case, there is a possibility of identifying a wrong model as the optimum, because of the assignment of a disproportionately large number of evaluations to a non-optimal model. However, if the fitness levels

Table 6.1: Results for clustering problem (XB index values averaged over all runs)

	SA-TDO			NSGA-II		
Model	Mean	Best	Worst	Mean	Best	Worst
μ_1 (3 clusters)	0.1078	0.1062	0.1108	0.1080	0.1060	0.1102
μ_2 (4 clusters)	0.0828	0.0779	0.0977	0.0805	0.0775	0.0961
μ_3 (5 clusters)	0.0759	0.0626	0.1885	0.0641	0.0608	0.0882
μ_4 (6 clusters)	0.0940	0.0739	0.1467	0.0782	0.0710	0.1071
μ_5 (7 clusters)	0.1402	0.0886	0.2848	0.0963	0.0792	0.1234

are reasonably distinct, the selection pressure will be more biased towards the fittest (and potentially the best) model, thereby identifying the optimum more accurately. The benefits of the proposed approach will be more pronounced in such cases.



(a) Evolution of function values (XB index) of different models

(b) Proportion of evaluations assigned to the various models

Figure 6.3: Results obtained for the clustering problem using SA-TDO

6.4.2 Warehouse problem

A simple warehouse problem is formulated to demonstrate another application of the SA-TDO algorithm, as follows. There are K locations with specific demands, which have to be catered for through N warehouses, to be set up at any of these K locations. The number of warehouses can vary from 1 to K. The warehouse at each location has a limit on the maximum capacity of the goods it can produce.

The overall cost of the operation of this setup includes the following two key components:

1. The installation cost (IC), which is assumed to depend upon the number of warehouses and the total capacity of all warehouses. It is calculated using Equation 6.1.

$$IC = 5 \times 10^6 \times N + 500 \times T \tag{6.1}$$

   where N is the number of warehouses and T is the sum of their capacities.

2. The transportation cost (TC), which depends on the units transported, and the distance between the supplying and receiving ends. For the presented study, four locations are chosen. A distance matrix D between the various locations is shown in Table 6.2. The diagonal elements in the matrix are set to a small value (assuming that in general the distance between the supplying and receiving ends will not be zero).

Table 6.2: Distance chart for the warehouse problem in km ($C \equiv$ Canberra, $S \equiv$ Sydney, $M \equiv$ Melbourne, $B \equiv$ Brisbane)

	C	S	M	B
C	5	246.99	466.35	940.61
S	246.99	5	713.33	728.83
M	466.35	713.33	5	1371.06
B	940.61	728.83	1371.06	5

The TC is then calculated as given in Equation 6.2.

$$TC = \sum_{i=1}^{N} \sum_{j=1}^{K} D_{ij} W_{ji} \times C_u \tag{6.2}$$

where D_{ij} denotes the distance between the i^{th} and j^{th} locations, W_{ij} denotes the supply received by a location i from a warehouse at location j, and C_u is the cost of transportation of a unit mass for a unit distance. C_u is set to 1 for the problem considered.

The aim is to minimize the total cost $= TC + IC$, subject to the demand and supply constraints :

$$\begin{aligned} \sum_{j=1}^{N} W_{ij} \geq Dem_i \quad \text{for } i = 1 \ldots K, \\ \text{and} \quad \sum_{i=1}^{K} W_{ij} \leq Cap_j \quad \text{for } j = 1 \ldots N \end{aligned} \tag{6.3}$$

where Dem_i denotes the demand at the i^{th} location and Cap_j the maximum capacity of the j^{th} warehouse. The demand at various sites and the maximum capacity of the warehouse at each site are listed in Table 6.3. In this study, the number of warehouses is allowed to vary from 1 to 4 and, hence, there are ${}^4C_1 + {}^4C_2 + {}^4C_3 + {}^4C_4 = 15$ competing models, which correspond to the possible placements of warehouses at different locations. Denoting the cities Canberra, Sydney, Melbourne and Brisbane by C, S, M and B respectively, the possible models corresponding to the location(s) of warehouses are: $\mu_1 \equiv \{C\}$, $\mu_2 \equiv \{S\}$, $\mu_3 \equiv \{M\}$, $\mu_4 \equiv \{B\}$, $\mu_5 \equiv \{C, S\}$, $\mu_6 \equiv \{S, M\}$, $\mu_7 \equiv \{M, B\}$, $\mu_8 \equiv \{C, M\}$, $\mu_9 \equiv \{C, B\}$, $\mu_{10} \equiv \{S, B\}$, $\mu_{11} \equiv \{C, S, M\}$, $\mu_{12} \equiv \{S, M, B\}$, $\mu_{13} \equiv \{C, M, B\}$, $\mu_{14} \equiv \{C, S, B\}$, $\mu_{15} \equiv \{C, S, M, B\}$.

The variables for each model are the outflows (supplies) from the warehouse locations to all the receiving ends. The constraints on demand at each location and total supply from each warehouse are handled using the penalty function approach in which the objective value infeasible solutions are deteriorated by

Table 6.3: Demand and capacity at each site for the warehouse problem ($C \equiv$ Canberra, $S \equiv$ Sydney, $M \equiv$ Melbourne, $B \equiv$ Brisbane)

	C	S	M	B
Demand	10000	15000	12000	20000
Capacity	7000	26000	10000	33000

adding a penalty proportional to the violation of the constraints. The penalty for a feasible solution is zero. The penalty function (PF) used here is formulated as in Equation 6.4.

$$\begin{aligned} PF = 10^6 \times \sum_{i=1}^{K} (max(0, Dem_i - \sum_{i=1}^{N} W_{ij})) \\ + 10^6 \times \sum_{i=1}^{N} (max(0, \sum_{j=1}^{K} W_{ij} - Cap_i)) \end{aligned} \tag{6.4}$$

Hence, the (penalized) objective to be minimized, takes the form given in Equation 6.5.

$$\begin{aligned} F = \sum_{i=1}^{N} \sum_{j=1}^{K} D_{ij} W_{ji} \times C_u \\ + 10^6 \times \sum_{i=1}^{K} (max(0, Dem_i - \sum_{i=1}^{N} W_{ij})) \\ + 10^6 \times \sum_{i=1}^{N} (max(0, \sum_{j=1}^{K} W_{ij} - Cap_i)) \end{aligned} \tag{6.5}$$

This is a simple scenario for illustrating the working of the proposed algorithm. However, better estimates can be made based on real-life data. It is also noteworthy here that, of the fifteen models listed, only four $(\mu_{10}, \mu_{12}, \mu_{14}, \mu_{15})$

can have feasible solutions. This is because for the rest of the models, the total demand of all the locations exceeds the maximum total capacity of the supplying warehouses. However, these models are still considered while solving the problem, in order to pose an additional challenge to the SA-TDO algorithm.

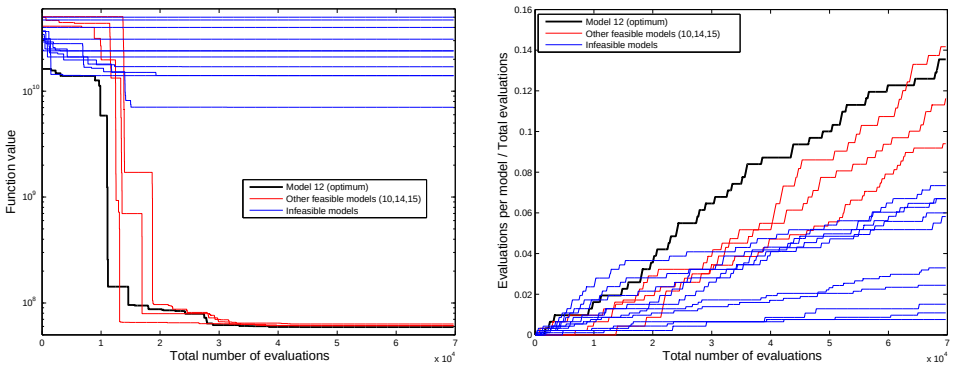
The warehouse problem is first solved using the SA-TDO algorithm, and then using NSGA-II. The former is run independently thirty times with different random seeds, with the same parameters as used for solving the clustering problem. The average number of evaluations utilized by the SA-TDO algorithm is approximately $73,000$. Since 15 models are being evaluated, NSGA-II is run for $73000/15 \approx 5000$ evaluations for each model. Thirty independent NSGA-II runs are done for each model. The crossover and mutation parameters used for NSGA-II are the same as those used for the clustering problem. The population size used is 100. A comparison of solutions obtained using SA-TDO and NSGA-II for the warehouse problem is shown in Table 6.4. Only the feasible models (for which total capacity of warehouses is greater than or equal to the total demand) are shown.

Table 6.4: Results (cost) for warehouse problem (all values in millions)

				NSGA-II								
	SA-TDO			5000 eval/model			10000 eval/model			15000 eval/model		
Model	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst	Mean	Best	Worst
μ_{10}	66.61	58.28	78.14	70.55	58.44	82.24	69.15	58.39	81.77	67.65	58.39	80.84
μ_{12}	59.62	54.48	73.69	69.21	60.40	79.54	60.86	57.41	67.82	57.93	55.31	60.10
μ_{14}	63.58	60.56	78.97	71.9	62.87	88.08	65.58	61.26	79.74	62.75	60.43	71.84
μ_{15}	66.58	60.45	108.11	77.06	69.15	87.36	67.00	62.96	75.52	62.90	60.71	69.02

Based on the average values obtained from multiple runs, model 12 (μ_{12}) is identified as the optimum model by both the SA-TDO and NSGA-II algorithms. However, unlike for the clustering problem, the average function values obtained for the optimum model obtained using SA-TDO are better than those using

NSGA-II. This is because, unlike the case of clustering, the fitness values for the different models are reasonably distinct, which makes it easier for the SA-TDO algorithm to converge to the optimum model. Once again, the Mann-Whitney U test is used to establish the statistical significance of the difference between the results obtained by the two algorithms for μ_{12} across multiple runs. The U and z values corresponding to the objective values obtained using SA-TDO and NSGA-II are $U = 835$ and $z = -5.68$ respectively, which indicates that the difference in the results obtained using the two algorithms is statistically significant. This confirms that the results from SA-TDO are better than those from NSGA-II.



(a) Evolution of objective values of different models with function evaluations

(b) Proportion of evaluations assigned to various models

Figure 6.4: Results obtained for the warehouse problem using SA-TDO

Evolutions of the function values of various models for a typical SA-TDO run are shown in Figure 6.4(a). The four feasible models ($\mu_{10}, \mu_{12}, \mu_{14}, \mu_{15}$) start off with high values of objective function, *i.e.* penalized infeasible solutions. As the search progresses, feasible solutions are found for these models, and the function values reduce significantly where as the infeasible models continue to exhibit worse function values as expected. Eventually, model 12 outperforms the other

candidate models in terms of the objective value, in this case the total cost.

The assignment of function evaluations to each model during the search for a typical run is shown in Figure 6.4(b). Owing to their better fitness, the feasible models are assigned significantly higher proportion of function evaluations, as can be seen in the Figure 6.4(b). Amongst the feasible models themselves, the relatively fitter model (μ_{12}) is chosen more often for evaluation during the course of evolution. It is also to be noted here that the number of function evaluations depends not only on the fitness of the model, but also the number of variables in that model (recall that epoch length $M = L \times n_{\text{var}}$). Therefore, a model with a larger number of variables (such as μ_{15}) may be selected fewer times than a model with smaller number of variables (such as μ_{12}), but may still have been evaluated a greater number of times. Eventually, towards the end of the run, the optimum model ($\mu*$) is expected to have been evaluated a sufficient number of times to converge to the corresponding optimum variables ($\mathbf{x}_\mu*$).

In two further experiments, NSGA-II is run with 10,000 and 15,000 evaluations respectively for each model, and the results compared with those from the SA-TDO. The results are listed in Table 6.4. It can be seen that the SA-TDO algorithm is able to obtain better results in terms of total cost as compared to NSGA-II for the 10000 evaluations per model case. For the case of 15,000 evaluations per model, NSGA-II outperforms SA-TDO in terms of average function values, but at the expense of much higher total number of function evaluations.

6.5 Summary

The presented work identifies a need to develop optimization methods for TDO problems, in which a number of candidate models may exist for the design. Possible solutions to such problems can vary in terms of number of variables and also the physical quantities they represent, depending upon the models. Very limited studies are available currently in the field of TDO.

A simulated annealing based algorithm is proposed for handling TDO problems. The proposed algorithm (SA-TDO) evaluates all models in parallel and allocates function evaluations based on their relative fitness. By spanning model and variable space simultaneously, the algorithm avoids extensive evaluation of each candidate model, and identifies the best model (and optimum solutions) with much less computational expense. The performance of the algorithm is demonstrated on two different problems – a clustering problem with a variable number of clusters, and a warehouse problem in which possible number and locations of warehouses can vary. Although the empirical examples considered are very simple in nature, the studies clearly identify the benefits the proposed algorithm offers. The approach shows great promise for application to a number of optimization problems, in which function evaluations may be computationally intensive and therefore explicitly optimizing each model may not be viable.

While potentially promising, the proposed algorithm is still rudimentary in its current form. A number of directions for further research can be identified to make it more rigorous and competitive for real life applications. As a preliminary implementation, the constraint handling in the algorithm is done using the penalty function method, the performance of which is usually sensitive to the choice of penalty coefficients. However, the framework allows for the incorpora-

tion of a more explicit constraint handling mechanism, such as that presented earlier in Chapter 3. Similarly, the algorithm can be enhanced to handle multiple objectives by incorporating concepts from the C-PSA algorithm in Chapter 3. In such cases, the *fitness* measure of each model can be calculated based on the non-dominated solutions achieved for each model. Thus, the foundation laid in this chapter allows for the further development of SA-TDO to deal with generic problems with constraints and multiple objectives.

Chapter 7

Further Enhancements and Applications

Abstract

In this chapter, the IDEA and C-PSA algorithms proposed in previous chapters are further enhanced by embedding a local search and surrogate modeling in them, respectively, to enable them to deal with engineering problems which have severe restrictions on function evaluations owing to their computational complexity. Thereafter, a number of engineering design problems are solved using the proposed algorithms. Comparisons with previously published results are undertaken to gauge their performances.

7.1 Overview

In the preceding chapters of this thesis, the challenges encountered in engineering design optimization are highlighted, and proposals to deal with them are dis-

cussed. In this chapter, the performances of the proposed algorithms are studied on a set of engineering design problems.

As engineering design problems are often computationally expensive, the limits on the number of evaluations are severe. The algorithms proposed in the previous chapters result in better convergence as compared to the conventional EAs. However, for computationally expensive problems, additional mechanisms may be required to find near-optimal solutions in an affordable number of evaluations. In this chapter, two such mechanisms are integrated within the algorithms proposed previously in this thesis. The enhancements include:

1. **Surrogate modeling**: When evaluation of objective(s) and/or constraints is expensive, the search can be guided by using *approximate* objective and constraint values instead, predicted from surrogate function(s) built using the already existing solutions. In the literature, surrogate assistance has, to date, been used predominantly in the EA paradigm. In this work, it is extended to SA. Surrogate modeling is integrated in the C-PSA algorithm proposed earlier. The resulting algorithm is termed Surrogate Assisted Simulated Annealing (SASA).

2. **Local search**: When used independently, local search methods have limited applicability. While they can quickly converge to a local optimum solution, the global optimum solution is difficult to achieve, since their performance depends significantly on the starting solution. However, the swiftness of local search can be taken advantage of by integrating it with global search methods. This approach of combining a global search method with local search is known as a *memetic* algorithm. In this work, a local search is embedded in IDEA to further expedite convergence. The resulting

algorithm is referred to as Infeasibility Empowered Memetic Algorithm (IEMA).

Following the description of these final two methods, experiments are conducted on a number of engineering examples using the algorithms proposed in this thesis. In order to highlight the contributions, comparisons with some of the results available in the literature are included. The rest of the chapter is organized as follows. SASA and IEMA are described in Sections 7.2 and 7.3 respectively. Numerical experiments on engineering design problems are detailed in Section 7.4. Section presents a summary of the chapter.

7.2 Surrogate Assisted Simulated Annealing

In surrogate modeling, an appropriate mathematical model that can fit the given problem (objective and constraint functions) is sought. If such a model can be identified, the *approximate* values of the objective functions and constraints determined using the model can be used to guide the search, thereby avoiding a need for time-consuming real evaluations of solutions. The process of building a model (based on truly evaluated data) is referred to as *training*, whereas the determination of approximate function and constraint values using the model is referred to as *prediction*. For problems in which good predictions (close to the actual objective values) can be made, surrogate modeling is a beneficial tool for solving computationally expensive problems [149].

7.2.1 Surrogate modeling

Given a data set comprising input variables $\mathbf{x}$ and an output response y, building a surrogate involves constructing a function $F(\mathbf{x})$, such that $F(\mathbf{x}) \approx y$. A model

is an accurate representation of the given data if the predicted responses using the model are close to the actual responses y. A number of surrogate models are available in the literature. In the presented work, two of them are considered.

Response Surface Method (RSM)

The RSM is a linear or polynomial regression which uses first or second degree polynomials to fit the data [150]. A generic second order quadratic polynomial model, with m input variables $\{x_1, x_2, \dots x_m\}$ can be written as

$$y(\mathbf{x}) = \beta_0 + \sum_{i=1}^{m} \beta_i \, x_i + \sum_{i=1}^{m} \beta_{ii} \, x_i^2 + \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \beta_{ij} \, x_i \, x_j \tag{7.1}$$

where $\beta_0, \beta_i, \text{and} \beta_{ij}$ are the unknown parameters of the model that are determined from the given data. In vector form, this can be written as $y(\mathbf{x}) = \mathbf{f}^T \mathbf{b}$, where the vector $\mathbf{f}$ contains all the terms of $x_1, x_2, \dots, x_m$ and vector $\mathbf{b}$ contains all the unknown coefficients. The values of the unknown coefficients are determined using the least squares method. The least squares estimate of $\mathbf{b}$ is given by,

$$\hat{\mathbf{b}} = (\mathbf{F}^T \mathbf{F})^{-1} \mathbf{F}^T \mathbf{Y}, \tag{7.2}$$

where $\mathbf{F}$ is a matrix containing N rows, with each row being a vector $\mathbf{f}^T$ evaluated at an observation and $\mathbf{Y}$ indicates the observed responses.

Radial Basis Function (RBF)

Another technique for approximating responses is known as the RBF [151]. The function for approximating the response y is constructed as

$$y(\mathbf{x}) = \sum_{i=1}^{k} w_i \, \phi(\|\mathbf{x} - \mathbf{x}_i\|) \tag{7.3}$$

where $\phi(\cdot)$ are the radial basis functions, $\|\cdot\|$ is usually taken as the Euclidean norm and w_i are the unknown weights to be determined for the model. A RBF is symmetric around its associated center $\mathbf{x}_i$. A common RBF is the Gaussian function with the Euclidean norm.

$$\phi(\|\mathbf{x} - \mathbf{x}_i\|) = e^{-r^2/\sigma^2}$$

where r is the Euclidean distance between $\mathbf{x}$ and $\mathbf{x}_i$, and σ is the scale or width parameter. In the generalized RBF network, the number of centers (k) are usually less than the number of observations (N). The unknown weights w_i are determined using least squares estimates.

7.2.2 Surrogate Assisted Simulated Annealing (SASA) algorithm

The proposed SASA is outlined in Algorithm 7.1. The main steps of SASA are the same as those of the C-PSA proposed in Chapter 3, with the addition of the surrogate modeling. In SASA, an archive of all truly evaluated solutions (archive_all) is maintained in addition to the archive of all non-dominated solutions. After every prescribed number of function evaluations t_{train}, a surrogate model is built for the objective and constraint functions. The surrogate framework used for building the surrogate model is the same as that proposed in [149]. During the search, the most recently evaluated 1000 solutions are used for the surrogate modeling. Out of these solutions, 90% are used for building the surrogate, whereas the rest

of them are used for validation of the surrogate. Two different surrogates (RSM and RBF) are used in the presented studies to approximate the objective and constraint function responses. Of these, the model that gives a lower prediction error is used to predict the responses, provided the prediction error is smaller than a threshold (0.05). This is because a model with large prediction error may not be an accurate representation of the function response; and using solutions predicted by such a model can misguide the search. The prediction error (p_{err}) is calculated as shown in Equation 7.4

$$p_{err} = \frac{1}{\max_i(y) - \min_i(y)} \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \tag{7.4}$$

where y_i is the actual response and $\hat{y}_i$ the predicted response. If a valid surrogate model is found (if its prediction error is less than the threshold) then it is used to determine the objective and constraint values for that particular iteration. If a valid surrogate model is not found, then the true evaluation of the solution is done instead. If a surrogate is used for prediction during an iteration, all the predicted solutions for that particular iteration are stored separately in a different archive (archive_pred). At the end of the iteration (when the search has progressed through M predicted solutions, where M is the epoch length), a non-dominated sorting is done on the feasible solutions in archive_pred. The non-dominated solutions so obtained are then truly evaluated and merged with the archive of (truly evaluated) non-dominated solutions. From this merged set (archive + archive_pred), any dominated solutions, if present, are removed and the resulting set becomes the updated archive (of non-dominated solutions). Thereafter, all solutions in the archive_pred are removed.

Algorithm 7.1 Surrogate Assisted Simulated Annealing (SASA)

Require: N, M, T_{max}, T_{min}, T_G, HL, SL, α, P_i, P_f, σ_i, σ_f

Initialize archive, archive_all {archive contains non-dominated solutions, archive_all contains all evaluated solutions}
Set $T = T_{max}$, $\mathbf{x_{old}}$ = random solution in search space
for $i = 1$ to N **do**
  if $N_{evals} > t_{train}$ and modulo$(N_{evals}, t_{train}) = 0$ **then**
    $train\_surrogate = 1$
    sur = Build_Surrogate(archive_all)
    if $p_{err} < 0.05$ **then**
      $surr\_valid = 1$
      Initialize archive_pred
    else
      $surr\_valid = 0$
    end if
  else
    $train\_surrogate = 0$
  end if
  for $j = 1$ to M **do**
    $\mathbf{x_{new}} = perturb(\mathbf{x_{old}})$ {Laplacian perturbation if $\mathbf{x_{old}}$ is feasible, ADD perturbation otherwise}
    if $train\_surrogate = 0$ or $surr\_valid = 0$ **then**
      $\mathbf{f_{new}}, \mathbf{g_{new}} \leftarrow$Evaluate$(\mathbf{x_{new}})$
    else
      $\mathbf{f_{new}}, \mathbf{g_{new}} \leftarrow$Predict$(\mathbf{x_{new}}, sur)$
      Update archive_pred
    end if
    Follow acceptance criteria as described in Algorithms 3.3 and 3.4 (Chapter 3)
  end for
  archive_pred ← non-dominated-sort(archive_pred)
  Evaluate (archive_pred) {Evaluate all non-dominated solutions in archive_pred}
  archive ← non-dominated-sort(archive ∪ archive_pred)
  Remove all solutions from archive_pred
  Update T, T_G, σ
end for

7.2.3 Preliminary experiments - SASA

The proposed SASA algorithm is first tested on the set of CTP test problems. To evaluate its performance, thirty independent runs are conducted on each CTP test problem using a small number of iterations (50). The average number of *actual* evaluations (FES) used by SASA are observed. Thereafter, each of the multi-objective algorithms, NSGA-II and IDEA, are run thirty times for the same number of FES (listed in Table 7.2), and the results obtained using the three algorithms are then compared. For comparison, displacement and hypervolume metrics are used. The parameters used for SASA are listed in Table 7.1, and those used for NSGA-II and IDEA are listed in Table 7.2. The building of surrogates in SASA is performed using the framework suggested in [149]. The parameters corresponding to the surrogate modeling are listed in Table 7.3.

Table 7.1: Parameters used for SASA

Parameter	Value
Initial probability of feasible to infeasible jump (P_i)	0.5
Final probability of feasible to infeasible jump (P_f)	0.01
Probability of acceptance used for calculating initial temperature (P_{T_i})	0.9
Final temperature (T_f)	1e-5
No. of exploring solutions for ADD (N_{add})	1
Exploration radius for ADD (r)	1e-3
Initial scaling factor for Laplacian mutation (σ_i)	1
Final scaling factor for Laplacian mutation (σ_f)	0.1
Epoch length (M)	$20 \times n_{var}$
Iterations (N)	50
Hard limit on no. of solutions in archive (HL)	100
Soft limit on no. of solutions in archive (SL)	150
Number of solutions rejected consecutively for restart ($Kmax$)	10

Comparison of displacement metric values obtained using various algorithms is shown in Table 7.4. It is seen that SASA outperforms NSGA-II and IDEA for all problems except CTP8 in terms of mean values. For CTP8, the performance of IDEA is the best among the three algorithms studied. This may be due

Table 7.2: Parameters used for NSGA-II and IDEA

Parameter	Value
Population size	100
Max. FES	for CTP2 : 6000 for CTP3 : 5900 for CTP4 : 6800 for CTP5 : 5900 for CTP6 : 9900 for CTP7 : 7300 for CTP8 : 11900
Crossover probability	0.9
Crossover index	15
Mutation probability	0.1
Mutation index	20

Table 7.3: Parameters used for surrogate modeling in SASA

Parameter	Value
Training data fraction	0.9
Training samples used	1000
Prediction accuracy threshold (p_{err})	0.05
Periodic training interval (t_{train})	500

to the highly discontinuous objective space, in which IDEA has the advantage of searching using the infeasible solutions, whereas SASA may have problems building a surrogate for the objective functions/constraints.

Hypervolume metric comparison (Table 7.5) also shows similar trend as those by the displacement metric. SASA obtains higher values of hypervolume as compared to NSGA-II and IDEA, except for CTP8 for which IDEA shows the best performance on average.

Apart from obtaining good average metric values, SASA also exhibits low values of the standard deviation (S.D.) in the results (except for CTP8). This suggests that the proposed SASA is able to obtain good quality approximation to the Pareto fronts consistently.

Table 7.4: Comparison of displacement metric obtained using SASA, NSGA-II and IDEA

	SASA		NSGA-II		IDEA	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
CTP2	**0.0016**	0.0021	0.0236	0.0122	0.0204	0.0112
CTP3	**0.0179**	0.0047	0.1256	0.0621	0.0999	0.0567
CTP4	**0.0712**	0.0314	0.1703	0.0577	0.1349	0.0519
CTP5	**0.0021**	0.0006	0.0129	0.0100	0.0113	0.0106
CTP6	**0.0037**	0.0022	0.1195	0.1382	0.0129	0.0122
CTP7	**0.0041**	0.0103	0.0451	0.0307	0.0306	0.0223
CTP8	0.0493	0.0931	0.1905	0.1462	**0.0259**	0.0267

Table 7.5: Comparison of hypervolume metric obtained using SASA, NSGA-II and IDEA

	SASA		NSGA-II		IDEA	
	Mean	S.D.	Mean	S.D.	Mean	S.D.
CTP2	**9.0160**	0.0874	8.2395	0.5006	8.3093	0.5095
CTP3	**8.9366**	0.0295	8.1545	0.4789	8.2709	0.4942
CTP4	**8.4892**	0.1705	7.7931	0.4147	8.0195	0.3826
CTP5	**8.9121**	0.0344	8.1760	0.4858	8.2564	0.4803
CTP6	**36.6061**	0.2096	31.4846	5.5777	36.0543	0.6057
CTP7	**9.4315**	0.4734	7.7532	1.3039	8.2832	0.9949
CTP8	33.8385	4.2748	28.3557	5.6023	**34.9932**	1.1874

7.3 Infeasibility Empowered Memetic Algorithm (IEMA)

In the literature, global search methods (such as EAs) are often used in conjunction with local search methods (such as the gradient search) to search efficiently for the optimum solutions. This *hybrid* approach is referred to as a memetic algorithm [152]. A comprehensive review of memetic algorithms can be found in [153].

The primary purpose of the local search is to inject good quality solutions into the population early in the search. Local search methods are usually quick to converge to an optimum in the neighborhood. Population-based global search

methods can provide a number of promising starting solutions to a local search, which in turn can help in identifying good quality solutions quickly. These solutions then further help to generate better quality solutions through evolutionary recombination and mutation.

The algorithm presented in this section, the IEMA, is a memetic algorithm which uses IDEA as a global search method. Within each generation, a local search is initiated from a promising solution in the population. The IEMA tries to exploit the advantages of two approaches, *i.e.*, a) intensifying the search near the constraint boundary by preserving marginally infeasible solutions and b) using the effectiveness of a local search to expedite the convergence in potentially optimal regions of the search space.

The proposed IEMA is outlined in Algorithm 7.2. In IEMA, during each generation, apart from the evolution of the solutions using IDEA, a local search is performed from a solution in the population for a prescribed number of function evaluations (set to $20\times n_{var}$ in the presented studies, where n_{var} is the number of design variables). Sequential quadratic programming (SQP) [154] is used in the presented studies for the local search. The starting solution for the local search is determined from the solutions in the population in the following way:

1. If the local search in the previous generation was able to improve the best solution, the new best solution is used as the starting solution for the local search.

2. If the local search was unable to improve the best solution in the previous generation, it is evident that the existing best solution (in the previous generation) is either not a good starting solution for the local search, or close enough to optimum (either local or global) so that further improvements

are difficult. In such a case, a random solution is selected from the *high ranked* infeasible solutions *and* the feasible solutions in the population, in an attempt to further improve the objective value. High ranked infeasible solutions consist of the $N_{inf} = \alpha * N$ solutions (refer to Algorithm 7.2)

After performing the local search the worst solution in the population is replaced by the best solution found from the local search. The ranking of solutions is done in the same way as in IDEA. The injection of good quality solutions found using the local search guides the population towards potentially optimal regions of the search space. In turn, the evolved solutions act as good starting solutions for the local search in subsequent generations. In this way, IDEA and local search work together to identify the optimum solution.

Algorithm 7.2 Infeasibility Empowered Memetic Algorithm (IEMA)

Require: N {Population size}
Require: $N_G > 1$ {Number of generations}
Require: $0 < \alpha < 1$ {Proportion of infeasible solutions}
1: $N_{inf} = \alpha * N$
2: $N_f = N - N_{inf}$
3: pop_1 = Initialize()
4: Evaluate(pop_1)
5: **for** $i = 2$ to N_G **do**
6: $childpop_{i-1}$ = Evolve(pop_{i-1})
7: Evaluate($childpop_{i-1}$)
8: (S_f, S_{inf}) = Split($pop_{i-1} + childpop_{i-1}$)
9: Rank(S_f)
10: Rank(S_{inf})
11: $pop_i = S_{inf}(1 : N_{inf}) + S_f(1 : N_f)$
12: $\mathbf{x} \leftarrow$ Choose starting solution in pop_i
13: $\mathbf{x_{best}} \leftarrow$ Local_ search ($\mathbf{x}$) {$\mathbf{x_{best}}$ is the best solution found using local search from $\mathbf{x}$}
14: Replace worst solution in pop_i with $\mathbf{x_{best}}$
15: Rank(pop_i) {Rank the solutions again in pop_i}
16: **end for**

7.3.1 Preliminary experiments - IEMA

The performance of the IEMA is first tested on g-series constrained test problems. A small number of function evaluations is used for this study, as compared to 350,000 used earlier in Chapter 3. Thirty independent runs are conducted for each problem using NSGA-II, IDEA and IEMA. The crossover and mutation parameters used are the same as listed in Table 7.2. A population size of 40 is used, and the maximum number of function evaluations is set to 10000.

The results obtained using NSGA-II, IDEA and IEMA are summarized in Table 7.6. It is seen that IEMA is able to achieve better (or same) median and mean values compared with NSGA-II and IDEA for the problems g01, g04, g07, g08, g09 and g12. For the remaining three problems (g02, g06 and g10), the performance of IDEA is better than those of NSGA-II and IEMA.

Table 7.6: Preliminary studies of IEMA on g-series test problems

		NSGA-II	IDEA	IEMA
g01	Median	-12.5276	-13.4862	**-15**
	Best	-14.9823	-14.9756	**-15**
	Mean	-12.378	-13.307	**-14.6094**
	Worst	-8.97155	-9.62892	-13.8281
	Std.	1.80534	1.53221	0.561871
	Feasible runs	30	30	30
g02	Median	0.694529	**0.747131**	0.535252
	Best	0.78176	**0.787357**	0.670918
	Mean	0.689805	**0.741544**	0.545908
	Worst	0.56026	0.627344	0.391273
	Std.	0.0501933	0.0420695	0.0607081
	Feasible runs	30	30	30
g04	Median	-30438.5	-30642.2	**-30665.5**
	Best	-30659.5	-30664.5	**-30665.5**
	Mean	-30435.3	-30624.8	**-30665.5**
	Worst	-30067.4	-30462.4	-30665.5
	Std.	167.35	49.9505	1.2509e-06
	Feasible runs	30	30	30
g06	Median	-5650.15	**-6641.44**	-6187.59
	Best	-6785.81	-6934.38	**-6958.4**
	Mean	-5677.13	**-6718.05**	-6363.21
	Worst	-1289.75	-6412.88	-5622.84
	Std.	1079.54	148.66	408.332
	Feasible runs	22	22	24
g07	Median	28.528	27.924	**24.8033**
	Best	24.637	24.9373	**24.3066**
	Mean	31.941	30.0609	**25.238**
	Worst	61.1841	44.5458	28.9968
	Std.	8.00498	4.99691	1.0988
	Feasible runs	30	30	30
g08	Median	-0.0958232	-0.095825	-0.095825
	Best	-0.095825	-0.095825	-0.095825
	Mean	-0.0868064	-0.0913778	-0.095825
	Worst	-0.025423	-0.0291436	-0.095825
	Std.	0.0233829	0.0169171	1.945e-09
	Feasible runs	30	30	30
g09	Median	685.131	682.588	**680.65**
	Best	681.878	680.973	**680.63**
	Mean	686.583	683.268	**680.769**
	Worst	696.628	692.745	681.773
	Std.	4.22735	2.27533	0.244093
	Feasible runs	30	30	30
g10	Median	8957.87	**8179.32**	9360.77
	Best	7593.53	7282.61	**7049.25**
	Mean	9310.95	**8814.34**	9784.72
	Worst	13331.9	16859.8	23449.3
	Std.	1448.28	1911.75	3426.46
	Feasible runs	29	29	25
g12	Median	-1	-1	-1
	Best	-1	-1	-1
	Mean	-1	-1	-1
	Worst	-1	-1	-1
	Std.	1.28959e-08	1.28959e-08	1.17494e-14
	Feasible runs	30	30	30

7.4 Engineering Design Problems

In this section, the experiments undertaken on engineering design optimization problems are presented. A number of benchmark problems are selected for this study, including Belleville spring design [3], helical spring design [3], heat exchanger design [155], speed reducer design [4], pressure vessel design [156], welded beam design [57], car side impact problem [99], bulk carrier design [157], and airfoil design [158]. The problem definitions are given in Appendix E.

A summary of the design experiments is provided in Table 7.7. For each experiment, thirty independent runs are done by varying the random seed. The parameters used for NSGA-II, IDEA and IEMA are listed in Table 7.8 and those for C-PSA and SASA in Table 7.9. The parameters for surrogate building in SASA remain the same as listed in Table 7.3 earlier. While NSGA-II and IDEA are run for all problems, IEMA is run only for continuous variable problems since it uses a gradient-based search. C-PSA and SASA, being multi-objective algorithms, are run only for the multi-objective problems. Also, C-PSA and SASA are run for continuous problems only because of the nature of mutation (Laplacian/ADD) currently implemented.

A population size of 40 is used for single-objective problems, while a population size of 100 is used for multi-objective problems. The number of function evaluations is limited to 1000 and 5000 for single and multi-objective problems respectively.

Table 7.7: Numerical experiments. (Note: IEMA is run for (single objective) continuous problems only, since it employs gradient search. C-PSA and SASA are run for (multi-objective) continuous variable problems only due to the nature of mutation currently implemented)

	NSGA-II	IDEA	IEMA	C-PSA	SASA
Belleville spring	✓	✓	✓	-	-
Car side impact	✓	✓	✓	-	-
Airfoil design	✓	✓	✓	-	-
Welded beam design (SO)	✓	✓	✓	-	-
Bulk carrier design (SO)	✓	✓	✓	-	-
Speed reducer	✓	✓	-	-	-
Pressure vessel	✓	✓	-	-	-
Heat exchanger	✓	✓	-	-	-
Helical spring (SO)	✓	✓	-	-	-
Helical spring (MO)	✓	✓	-	-	-
Bulk carrier design (MO)	✓	✓	-	✓	✓
Welded beam design (MO)	✓	✓	-	✓	✓

Table 7.8: Parameters used for IDEA and NSGA-II for studies on engineering design problems

Parameter	Value
Crossover probability	0.9
Mutation probability	0.1
Crossover distribution index	15
Mutation distribution index	20

Table 7.9: Parameters used for the C-PSA and SASA

Parameter	Value
Initial probability of feasible to infeasible jump (P_i)	0.5
Final probability of feasible to infeasible jump (P_f)	0.01
Probability of acceptance used for calculating initial temperature (P_{T_i})	0.9
Final temperature (T_f)	1e-5
No. of exploring solutions for ADD (N_{add})	1
Exploration radius for ADD (r)	1e-3
Initial scaling factor for Laplacian mutation (σ_i)	1
Final scaling factor for Laplacian mutation (σ_f)	0.1
Epoch length (M)	$20 \times n_{var}$
Hard limit on no. of solutions in archive (HL)	100
Soft limit on no. of solutions in archive (SL)	150
Number of solutions rejected consecutively for restart ($Kmax$)	10

7.4.1 Results : single-objective problems

The results obtained using NSGA-II, IDEA and IEMA are summarized in Table 7.10. It is seen that the average and median values obtained using IDEA and IEMA are better than those from NSGA-II for all the problems, except the airfoil problem for which NSGA-II obtains better values than IEMA. The best values obtained for each problem are also better than those from NSGA-II for all problems except the speed reducer problem for which IDEA is marginally worse than NSGA-II. The percentage improvements obtained using IDEA and IEMA over NSGA-II (where applicable), are shown in brackets alongside the objective values in Table 7.10.

The percentage improvement attained in using IEMA and IDEA over NSGA-II varies for different problems, but it can be seen that as high as 26.52 % improvement over the best result was obtained using IEMA (for the case of Belleville spring design). Furthermore, improvements in the median values indicate that IEMA is able to achieve good objective values consistently. Again, for the case of Belleville spring design, 46.07 % improvement is seen in the median value using IEMA compared with that obtained using NSGA-II. Improvements in the other problems are comparatively less in magnitude, but are still significant and consistent. The convergence plots for the median runs of each algorithm are shown in Figure 7.1.

In the summary of results shown in Table 7.10, the function evaluations used in some of the previous studies are also listed in addition to the best values reported. Except for recent studies by Isaacs [149] which also use 1000 evaluations for comparison, the numbers of function evaluations used in most other studies are much higher than those used here. Even so, the objectives values obtained using IEMA are better than (or very close to) the best reported

previously[1]. Also worth mentioning here is that although the best results reported for Belleville spring design and welded beam design in [149] use surrogate assisted algorithms, superior results are obtained in the presented studies without the use of surrogates. This also highlights a further scope of improvement over current studies, *i.e.*, the inclusion of surrogate-modeling techniques in IDEA and IEMA.

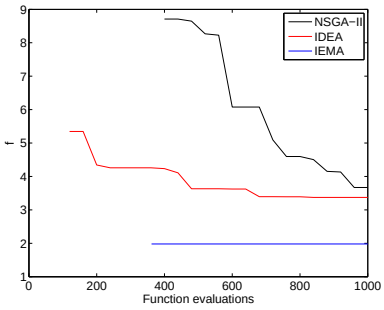
Although the results obtained using the proposed IEMA are very promising, it is not without limitations. The most prominent limitation of IEMA (at least in the current implementation, since a gradient-based local search SQP is used) is its inability to handle discrete variables (during the local search). Therefore, experiments have been reported only on problems with continuous variables. However, it could be resolved with use of more specialized operators. Secondly, the performance is also likely to deteriorate if the number of variables is very high, because the calculation of gradients itself will become computationally expensive in that case.

[1]Please note that slight variations in the results might also result from different precision of the variables or machines used for conducting previously reported experiments. In addition, for the Belleville spring design, the thickness of the spring is considered as a discrete variable in [11], but as a continuous variable in others, including the present studies.

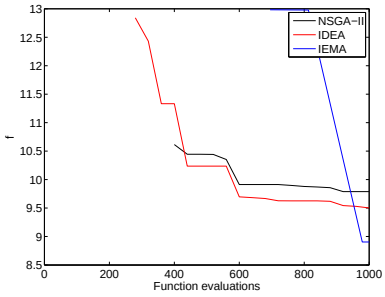
Table 7.10: Results for single objective engineering design problems. The numbers in the brackets indicate percent improvement in the objective values compared to those obtained using NSGA-II. (BS ≡ Belleville spring, BC ≡ bulk carrier, CSI ≡ car side impact, HS ≡ helical spring, AF ≡ airfoil, SR ≡ speed reducer, PV ≡ pressure vessel, WB ≡ welded beam)

.

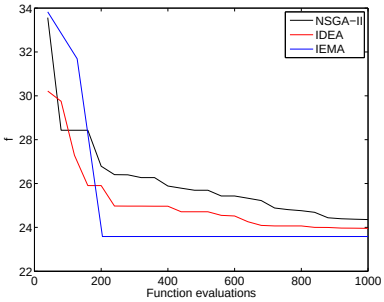
		NSGA-II	IDEA	IEMA	Other best reported [reference] (evals)
BS [3]	Median	3.67072	3.37236 (8.13 %)	**1.97968** (46.07 %)	2.121964 [159] (24K),
	Best	2.694	2.45197 (8.98 %)	**1.97967** (26.52 %)	2.29 [149] (1K),
	Mean	3.67317	3.34078 (9.05 %)	**2.3326** (36.50 %)	2.16256 [11](10K),
	Worst	6.12532	6.08839	7.5429	1.978715[3] (infeas.)
	Std.	0.99401	0.870218	1.17039	
	Feas. runs	23	23	30	
BC [157]	Median	9.78935	9.50139 (2.94 %)	**8.90361** (9.05 %)	
	Best	9.01664	8.79626 (2.44 %)	**8.60617** (4.55 %)	
	Mean	10.0897	9.81623 (2.71 %)	**9.20434** (8.77 %)	
	Worst	12.4753	12.0867	13.3163	8.6083[70](25K)
	Std.	0.857135	0.906221	1.2412	
	Feas.runs	30	30	21	
CSI [99]	Median	24.356	23.9561 (1.64 %)	**23.5857** (3.16 %)	23.585651 [99]
	Best	23.7872	23.6447 (0.60 %)	**23.5857** (0.85 %)	23.59 [160]
	Mean	24.414	24.0859 (1.34 %)	**23.5857** (3.39 %)	
	Worst	25.9416	25.16	23.5857	
	Std.	0.530955	0.357412	1.51e-07	
	Feas. runs	30	30	30	
HS[3]	Median	3.23683	**3.08335** (4.74 %)		
	Best	2.82821	**2.68817** (4.95 %)		
	Mean	3.55707	**3.29658** (7.32 %)		2.665 [161]
	Worst	6.77149	6.74868	-	2.71 [149] (1K)
	Std.	1.09286	0.82978		2.798 [162]
	Feas. runs	24	24		
AF[158]	Median	0.00219	**0.00198** (9.5 %)	0.00243 (-)	
	Best	0.00193	**0.00192** (0.5 %)	0.00198 (-)	
	Mean	0.00330	**0.00204** (35.15 %)	0.00340 (-)	1.9303e-3 [149] (1K)
	Worst	0.01171	0.00238	0.01264	
	Std.	0.00244	0.00013	0.00209	
	Feas. runs	30	30	29	
SR[4]	Median	3028.82	**3019.99** (0.29 %)		
	Best	**3003.1**	3003.77 (-)		
	Mean	3129.38	**3082.8** (1.49 %)		
	Worst	3742.32	3778.73	-	2994.74424[163] (54K)
	Std.	230.015	183.629		
	Feas. runs	30	30		
PV [156]	Median	7253.97	**6953.62** (4.14 %)		
	Best	6270.69	**6190.3** (1.28 %)		
	Mean	7284.16	**7076.27** (2.85 %)		
	Worst	9334.1	9738.97	-	6119.97 [149] (1K)
	Std.	678.625	720.031		
	Feas. runs	30	30		
WB [57]	Median	3.47566	3.03597 (12.65 %)	**2.38096** (31.50 %)	2.3854347 [163] (33K)
	Best	2.55464	2.49649 (2.28 %)	**2.38096** (6.80 %)	2.44 [149](1K)
	Mean	3.67048	3.13758 (14.52 %)	**2.65987** (27.53 %)	2.38119 [49](40K)
	Worst	5.58366	4.95963	6.14381	
	Std.	0.84549	0.548607	0.846031	
	Feas. runs	30	30	30	



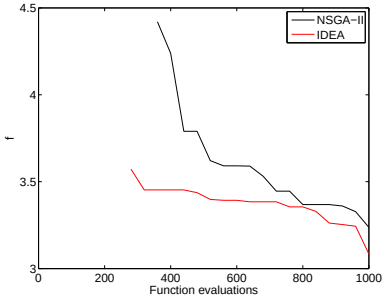
(a) Belleville spring



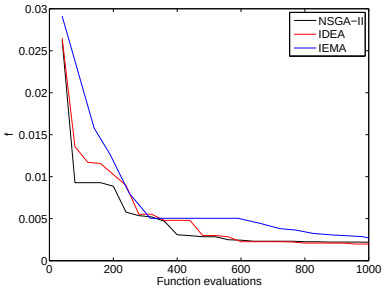
(b) Bulk carrier



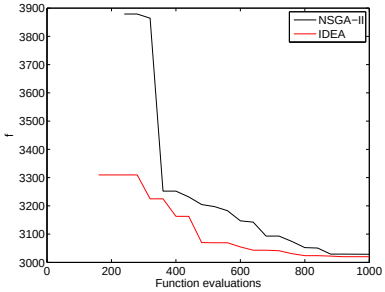
(c) Car side impact



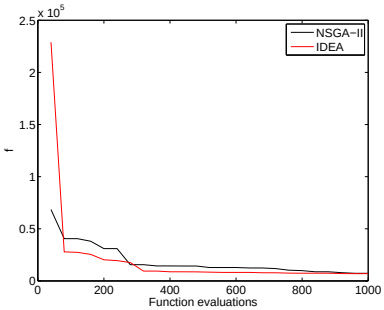
(d) Coil compression spring



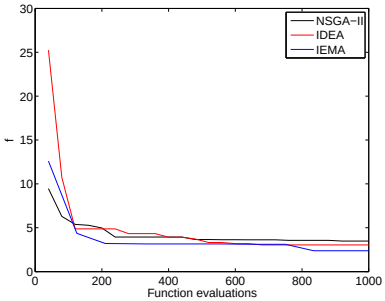
(e) Airfoil



(f) Speed Reducer



(g) Pressure Vessel



(h) Welded Beam

Figure 7.1: Median runs for single objective engineering design problems

7.4.2 Results : multi-objective problems

For multi-objective optimization problems, comparison of performances is done using displacement and hypervolume metrics. For calculating the displacement metric, the Pareto front is required. However, since the actual Pareto fronts are not known for these problems, reference Pareto fronts are constructed by collecting all the final solutions (obtained across all runs), and performing a non-dominated sorting on them. For calculating hypervolume, the reference point is constructed using the maximum value of each individual objective from the collection of all final solutions.

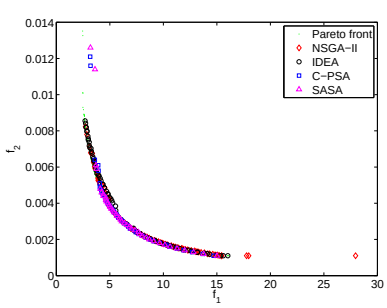
A summary of hypervolume and displacement metrics for multi-objective problems is given in Table 7.11. The mean and median hypervolume values obtained using IDEA are the best among the four algorithms. For the welded beam and bulk carrier problems, the best hypervolume value is obtained using C-PSA. Similarly, The mean displacement values obtained using IDEA outperform those from the other algorithms. Median values are also better than those from other algorithms, except for the case of welded beam where NSGA-II is marginally better than IDEA. For visualization, the median and best runs (based on displacement metric) are shown in Figures 7.2 and 7.3 respectively.

The performance of SASA is found to be poor for these problems, indicating that it could not find good approximations to the objective functions and/or constraints using a single surrogate. Another factor affecting the performance of SASA and C-PSA might be the premature termination of the algorithms due to the fixed number of function evaluations. In the current implementation, each trial solution from an infeasible solution requires two evaluations (using ADD) whereas from a feasible solution it requires only one evaluation (using Laplacian mutation). Therefore, the number of iterations can not be set to a fixed

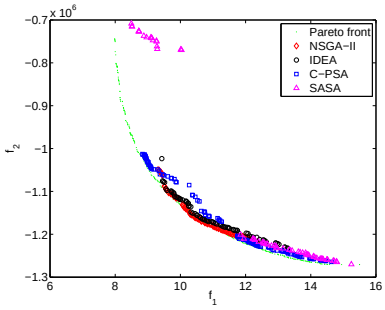
value to obtain a given number of function evaluations. If a complete annealing schedule is not achieved, the results obtained may be pre-converged as the more aggressive (greedy) search happens in SA during later iterations. Thus, possible enhancements to these algorithms in the future include a better way of controlling the number of function evaluations so that a full annealing schedule can be achieved for a given (fixed) number of evaluations. In addition, currently, only one surrogate (with least error) is used to approximate the objectives/constraints in the whole search space. Multiple surrogates could be used to obtain better approximations of objective and constraint values instead.

Table 7.11: Performance metrics for multi-objective engineering design problems (WB ≡ welded beam, BC ≡ bulk carrier, HS ≡ helical spring)

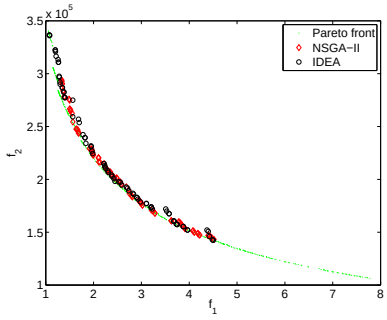
		NSGA-II	IDEA	C-PSA	SASA
		Hypervolume			
WB [57]	Best	0.619668	**0.620004**	0.62048	0.618914
	Mean	0.608964	**0.61107**	0.60001	0.597384
	Worst	0.569534	0.583043	0.563317	0.552339
	Median	0.615523	**0.615665**	0.60627	0.603992
BC [157]	Best	1.82735e+07	1.84188e+07	**1.85733e+07**	1.84227e+07
	Mean	1.70845e+07	**1.71724e+07**	1.48915e+07	1.48888e+07
	Worst	1.47561e+07	1.36718e+07	61542.2	35208.2
	Median	1.73023e+07	**1.73994e+07**	1.73611e+07	1.7085e+07
HS[3]	Best	**1.21144e+06**	1.21e+06	-	-
	Mean	1.12776e+06	**1.15204e+06**		
	Worst	777936	967152		
	Median	1.16512e+06	**1.16765e+06**		
		Displacement			
WB [57]	Best	**0.00275881**	0.00367227	0.00940382	0.00973421
	Mean	0.0082894	**0.00786422**	0.0211862	0.0229821
	Worst	0.0232389	0.0314141	0.0530463	0.0688903
	Median	**0.00412279**	0.00449162	0.0129685	0.0138647
BC [157]	Best	1957.08	1154.46	**611.706**	668.657
	Mean	4691.03	**4421.62**	10514.3	10062.2
	Worst	9502.72	12559.8	53267.1	53605.3
	Median	4611.97	**4156.97**	4372.48	5030.96
HS[3]	Best	150.11	**108.516**	-	-
	Mean	610.232	**425.89**		
	Worst	2245.43	1292.64		
	Median	492.299	**379.889**		



(a) Welded beam

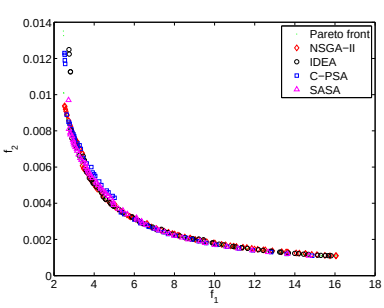


(b) Bulk carrier

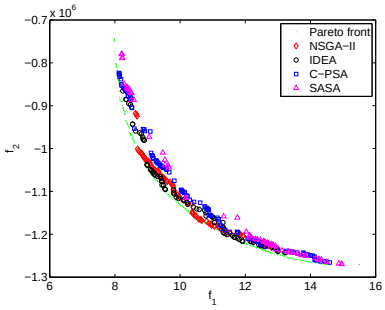


(c) Coil compression spring

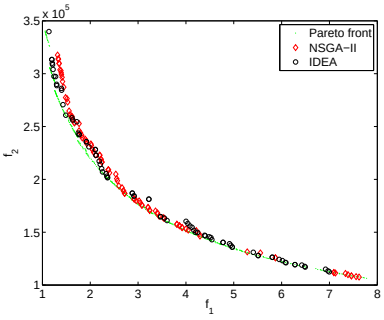
Figure 7.2: Median runs (based on displacement metric) for multi-objective engineering design problems. Reference Pareto fronts shown in the figure are constructed by assembling non-dominated solutions obtained from all runs.



(a) Welded beam



(b) Bulk carrier



(c) Coil compression spring

Figure 7.3: Best runs (based on displacement metric) for multi-objective engineering design problems. Reference Pareto fronts shown in the figure are constructed by assembling non-dominated solutions obtained from all runs.

7.5 Summary

In this chapter, the algorithms proposed in previous chapters are further enhanced by the use of local search and surrogate modeling. A number of engineering design problems are then studied using the proposed algorithms with limited function evaluations. The proposed algorithms are able to achieve competitive and, in some cases, significantly better results than the current state-of-the-art algorithm NSGA-II as well as some others from previously published studies. Overall, the algorithms show a great deal of potential for applications in real-life engineering design optimization problems.

Chapter 8

Conclusions

8.1 Research and Outcomes

Optimization is an integral part of engineering design. In recent times, meta-heuristic techniques have gained popularity as generic optimizers. Their ability to handle complex objective functions commonly occurring in design optimization and to find solutions to multi-objective optimization problems in a single run, along with some other advantages, gives them a clear edge over conventional analytical optimization techniques. However, as these algorithms require a large number of computationally expensive simulations to find the optimum design, they are slow and unsuitable for a number of applications. Since the time taken to evaluate a design is independent of the optimizer itself, reduction in optimization time requires finding the optimum design with as few design evaluations as possible. The work presented in this thesis is directed towards finding potential areas in which function evaluations can be saved, thereby improving convergence and finding the optimum design in minimal time.

As a first step towards achieving the above mentioned goal, the constraint

handling mechanisms used with existing optimization algorithms are examined. Constraints which impose necessary restrictions on cost, strength, geometry or other relevant factors, are inevitably present in most engineering problems. It is also evident that optimum solutions are often bounded by these constraints, or in other words, the optimum solutions often lie on constraint boundaries. Since the final goal of optimization is to find *feasible* optimum design, during the course of a search most of the algorithms prefer feasible solutions over infeasible solutions. In this process, infeasible solutions are weeded out, effectively resulting in the search being conducted through the feasible space only. In this thesis, a new approach which is in contrast with the conventional *feasible-first* ranking techniques is proposed. The main idea proposed here is that since optimum solutions lie on constraint boundaries, an infeasible solution near a constraint boundary is better (in terms of convergence) as compared to a feasible solution away from it. Consequently, preserving infeasible solutions near the constraint boundaries can help achieve faster convergence to the optimum. In addition, these marginally infeasible solutions are also useful for trade-off studies, *i.e.* to find out if significant benefits can be achieved in the objective values by slight compromise of one or more of the constraints. In this work, this concept is implemented in an evolutionary algorithm (Infeasibility Driven Evolutionary Algorithm, IDEA) to demonstrate consistent and significant improvements in convergence for a number of single- and multi-objective mathematical benchmarks and engineering design problems. For the engineering problems studied, up to 9% and 35% improvements in the best and mean objective values, respectively, were achieved using this improved constraint handling method.

The next related study is also in the area of constraint handling, but within the framework of another metaheuristic, Simulated Annealing (SA). SA is a

robust optimization technique with a strong mathematical foundation, and also has a proof of convergence subject to certain conditions. Its ability to escape local minima by probabilistically accepting worse solutions during the search sets it apart from greedy search methods such as hill climbing. However, being a single-point method, SA has an inherent disadvantage for multi-objective optimization problems compared with population-based methods which can capture the whole Pareto front in a single run. Additionally, SA also lacks an explicit constraint handling method. In recent times, some efforts have been made to handle these two issues separately. However, none of them have been directed towards solving difficult constrained multi-objective optimization problems. In this thesis, SA is enhanced to deal with such problems. The resulting SA is termed as Constrained Pareto Simulated Annealing (C-PSA). Its performance is studied on a number of difficult constrained problems; and it is found to be very competitive, and in some cases much better than those of the multi-objective evolutionary algorithms NSGA-II and IDEA.

The two algorithms for constrained optimization described above are further enhanced by the addition of local search (in IDEA) and surrogate modeling (in C-PSA). The local search in IDEA helps to induce good quality solutions during early generations, while the solutions evolved through IDEA provide good starting points for the local search. Together, these two mechanisms help to further improve the convergence rate. On the other hand, the surrogate modeling helps in reducing the number of function evaluations required by guiding the search based on approximate (predicted) objective and constraint values in lieu of actual evaluations (which may be expensive). These predictions are done by periodically building surrogate models during the search using the truly evaluated solutions.

A second aspect which proves prohibitive for the use of optimization algorithms is the case of large scale problems. While most of the existing algorithms are able to solve low dimensional problems effectively, their performance scales poorly with number of objectives and/or variables. This is commonly referred to as the *curse of dimensionality*. Both these situations (a high number of objectives and a high number of variables) are studied in this thesis.

For the case of *many-objective* problems, improvements are made in two different directions. Firstly, convergence is improved by using better ranking techniques. Conventional Pareto-dominance ranking is an inadequate means to drive the solutions towards the optimum since most solutions in the population are non-dominated with respect to each other. Therefore, additional mechanisms are required for better convergence, while also ensuring that the solutions have good diversity. In this thesis, two secondary ranking methods namely Cluster-sort and Modified-ϵ-dominance are proposed. Studies conducted on problems containing up to 30 objectives indicate that the proposed ranking methods are able to achieve improved convergence and diversity. Secondly, in the context of many-objective optimization, dimensionality reduction is studied. For certain many-objective problems, it is possible to reduce the original set of objectives to a subset with a smaller number of objectives, which can be solved more easily using conventional MOEAs. Current dimensionality reduction techniques use a population obtained after evolution through excessive number of generations in order to identify the reduced set of objectives. In this thesis, a novel method of dimensionality reduction in which the *corners* of the Pareto front are sought and used to identify the true dimensionality of the Pareto front is proposed. This approach has a number of advantages over conventional dimensionality reduction techniques, the foremost being that it is able to estimate the dimensionality with much fewer

evaluations. Additionally, the population size required to capture the corners does not, in general, increase exponentially with the number of objectives. Lastly, this method does not involve computationally expensive procedures such as the calculation of hypervolume. Experiments on benchmark problems containing up to 100 objectives and two engineering design examples illustrate the immense potential of this dimensionality reduction technique.

In the field of problems with large numbers of variables, the research conducted in this thesis improves upon the existing Cooperative Coevolutionary Algorithms (CCEAs). Coevolutionary algorithms operate by dividing the variables into several partitions and using different sub-populations to individually optimize variables in each partition. However, the conventional CCEAs do not have a suitable method for partitioning the variables appropriately. Partitioning them into constant or random groups, as practiced currently, may actually cause them to perform worse than typical evolutionary algorithms, especially for the case of *non-separable* problems. In this thesis, this drawback of conventional CCEAs is emphasized by studying two non-separable problems. A novel algorithm, CCEA-AVP, in which the partitioning is done adaptively based on the correlations among the variables, is proposed. In this way, strongly interacting variables are grouped together whereas there is minimal interaction among the different partitions. Separable and non-separable problems containing up to 100 variables are studied and the results indicate consistently competitive performance of the proposed CCEA-AVP.

Lastly, some exploratory studies are done on trans-dimensional optimization. Trans-dimensional optimization refers to problems where the set of variables is not unique, as assumed in all of the other problems studied above. The problem may have a number of candidate models, each of which has a corresponding set of

associated variables. Therefore, the global optimization process involves identifying the best model as well as the corresponding variable values. The conventional approach in such a case is to carry out optimization for each model exhaustively, and then choose the best model. However, this is inefficient. In this thesis, a SA-based Trans-dimensional Optimization algorithm (SA-TDO) is proposed, which searches through the variable and model space simultaneously. During the search, each model is optimized in parallel and the promising models (those which show superior objective values) are promoted by allotting them a relatively higher number of evaluations. The efficacy of this approach is demonstrated using two trans-dimensional optimization problems: the first is the clustering of a given data where the number and location of centroids is unknown; and the second is the minimization of operating cost for a transportation network.

8.2 Achievements

In summary, the contributions of this thesis can be grouped into four broad areas:

1. **Constraint handling**: Four algorithms are proposed to effectively deal with constrained optimization problems. Numerical experiments conducted on a number of constrained benchmark and engineering problems demonstrate significant improvements achieved over conventional EA. These are as follows.

   (a) Infeasibility Driven Evolutionary Algorithm (IDEA): an evolutionary algorithm incorporating new constraint handling mechanism, in which the search is focused near the constraint boundaries by explicitly preserving marginally infeasible solutions during the evolution.

(b) Constrained Pareto Simulated Annealing (C-PSA): a simulated annealing algorithm for solving constrained multi-objective optimization problems.

(c) Infeasibility Empowered Memetic Algorithm (IEMA): a memetic algorithm that uses IDEA as a global search method and SQP to make local improvements in the solutions. In its current form, IEMA can handle single-objective continuous variable problems only.

(d) Surrogate Assisted Simulated Annealing (SASA): similar to C-PSA, with the inclusion of surrogate modeling to occasionally guide the search based on approximate objective values instead of actual function evaluations.

2. **Large scale optimization (many objectives)**: Three improvements are proposed for handling problems with large numbers of objectives, as follows.

(a) Cluster-sort: a secondary ranking method for many-objective problems in which diversity is promoted based on clustering instead of the crowding distance method (implemented in NSGA-II framework). While this method does not improve convergence, it helps to achieve good diversity among solutions; and can be used in conjunction with faster converging schemes with poor diversity for good overall results.

(b) Modified-ϵ-dom: another secondary ranking method, which removes *mutual* ranking from the earlier proposed -ϵ-dom procedure, resulting in better convergence while maintaining good diversity.

(c) Pareto corner based dimensionality reduction: a novel technique for dimensionality reduction in which the true dimensionality of a problem

is identified using a key set of solutions (corners) on the Pareto front. An algorithm to identify the set of corners, namely the Pareto Corner Search Evolutionary Algorithm (PCSEA), is proposed; followed by the dimensionality analysis which involves analysis of the obtained corner solutions by omitting each objective sequentially. The proposed method is able to estimate the dimensionality using a fraction of evaluations required by contemporary techniques.

3. **Large scale optimization (many variables)**: a novel partitioning strategy, based on correlations among the variables, is proposed for Cooperative Coevolutionary Algorithms (CCEAs) to handle problems with large number of variables. The resulting algorithm, CCEA with Adaptive Variable Partitioning (CCEA-AVP) is able to solve a broad class of separable and non-separable problems more efficiently as compared to conventional EA and CCEA.

4. **Trans-dimensional optimization**: a SA-based trans-dimensional optimization (SA-TDO) algorithm is proposed to deal with optimization problems with multiple candidate models. This algorithm searches through the model and variable spaces simultaneously by preferably assigning evaluations to the models that exhibit better objective values during the search. This results in better quality solutions than when the same number of function evaluations is distributed equally among the models for their individual optimization.

All the above algorithms and strategies are coded in MATLAB. The list of the publications based on the research presented in this thesis is given at the beginning of the thesis. The algorithms developed herein are also currently being

used by the Multi-disciplinary Design Optimization (MDO) group at the University of New South Wales at Australian Defence Force Academy (UNSW@ADFA) for various applications. This includes designs of an unmanned underwater vehicle, an optimum hull-form for fast crafts (patrol boats) and an optimum inlet geometry of scram-jets for next generation spacecrafts in the ongoing Australian Space Research Program (ASRP).

8.3 Future Work

While this thesis highlights a number of areas in which existing optimization algorithms can be improved, it is by no means an exhaustive account. With the proliferation of research, particularly in metaheuristic techniques for optimization, we are bound to regularly come across exciting ideas and new insights in this field. No research is complete. However, from the work conducted in this thesis, a few directions (of many) that need further investigation can be identified as follows.

In terms of constraint handling, maintaining *good* infeasible solutions is certainly a beneficial idea, as adequately demonstrated. There may be a number of ways in which this concept may be utilized, one of which is studied in this thesis. While calculating the *Constraint Violation Measure*, effectively the same weight has been given to each constraint. However, since only critical/active constraints are of real interest to us (that's where optimum solutions lie), a better technique could be devised in which such constraints are identified and intrinsically preferred during the evolution, so as to focus the search near only the relevant portions of the constraint boundaries. Additionally, surrogate modeling can be embedded in IDEA for further reduction in the number of function evaluations and improved

computational efficiency.

While the SA-based algorithms (C-PSA and SASA) proposed in this thesis show good results for most problems, their performance is unreliable for studies using a fixed number of function evaluations (especially the engineering design problems studied in Chapter 7). This primarily appears to be due to insufficient time being available for the full annealing schedule to elapse within the given number of evaluations. Therefore, a better annealing scheme has to be devised so that these algorithms can be fairly compared with others for a fixed computational budget. Furthermore, multiple surrogates can be employed for a more accurate prediction of the objective and constraint values.

The study in dimensionality reduction for many-objective algorithms has potential extensions, the foremost of which is identifying the possibility of using appropriate set(s) of solutions rather than the whole Pareto front for dimensionality reduction. This work demonstrates one such possible set, termed *corner solutions* which, while being novel and apt for solving a wide range of problems, has a number of limitations as detailed in Chapter 4. Also, the corner search algorithm PCSEA itself has some limitations, and further studies are required to eliminate them. In addition, studies are required to build a stronger theoretical basis for such dimensionality reduction; which will also provide insights into further possibilities for improving this approach.

Another improvement would be to minimize the number of user-defined parameters in the above algorithms by making them self-adaptive during the search.

Lastly, while all the above contributions are implemented in either the EA or SA paradigm, they can be extended to other metaheuristics. In particular, recent studies have demonstrated superior performance of recombination/mutation operators used by Differential Evolution (DE) methods compared to those used in

conventional EAs. When coupled with some of the techniques presented in this thesis, the benefits of operators and constraint handling can be compounded to give more efficient algorithms.

References

[1] Collette, Y., Siarry, P.: Multiobjective optimization: Principles and case studies. Springer-Verlag Berlin Heidelberg (2003)

[2] Hedar, A., Fukushima, M.: Derivative-free filter simulated annealing method for constrained continuous global optimization. Journal of Global Optimization **35**(4) (2006) 291–308

[3] Siddall, J.N.: Optimal engineering design - principles and applications. Marcel Dekker, Inc., New York (1982)

[4] Golinski, J.: Optimal synthesis problems solved by means of nonlinear programming and random methods. Journal of Mechanisms **5**(3) (1970) 287–309

[5] Deb, K.: Optimization For Engineering Design: Algorithms And Examples. Prentice-hall of India Pvt. Ltd. (1998)

[6] Reeves, C.R.: Modern Heuristic Techniques for Combinatorial Problems. Orient Longman (1993)

[7] Darwin, C.: On the origin of species by means of natural selection. Murray, London (1859)

[8] Eiben, A., Smith, J.: Introduction to Evolutionary Computing. Natural Computing Series. Spriger-Verlag Berlin Heidelberg (2003)

[9] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation **6** (2002) 182–197

[10] Deb, K., Agrawal, S.: Simulated binary crossover for continuous search space. Complex Systems **9** (1995) 115–148

[11] Deb, K., Goyal, M.: A combined genetic adaptive search (GeneAS) for engineering design. Computer Science and Informatics **26** (1996) 30–45

[12] Kirkpatrick, S., Gelatt, C. D., J., Vecchi, M.P.: Optimization by Simulated Annealing. Science **220**(4598) (1983) 671–680

[13] Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence **6**(6) (1984) 721–741

[14] Bandyopadhyay, S., Saha, S., Maulik, U., Deb, K.: A simulated annealing-based multiobjective optimization algorithm: AMOSA. IEEE Transactions on Evolutionary Computation **12**(3) (2008) 269–283

[15] Ishibuchi, H., Yoshida, T., Murata, T.: Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Transactions on Evolutionary Computing **7**(2) (2003) 204–223

[16] Czyzak, P., Jaszkiewicz, A.: Pareto simulated annealing - a metaheuristic techinique for multiple-objective combinatorial optimization. Journal of Multi-Criteria Decision Analysis **7**(1) (1998) 34–47

[17] Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms - a comparitive case study. In: Proceedings of Parallel Problem Solving from Nature (PPSN-V). (1998) 292–301

[18] Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Shaker Verlag, Germany, ISBN 3-8265-6831-1 (1999)

[19] Coello Coello, C.A.: 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) (2002) 1245–1287

[20] Michalewicz, Z.: A Survey of Constraint Handling Techniques in Evolutionary Computation Methods. In McDonnell, J.R., Reynolds, R.G., Fogel, D.B., eds.: Proceedings of the 4th Annual Conference on Evolutionary Programming. The MIT Press, Cambridge, Massachusetts (1995) 135–155

[21] Mezura-Montes, E., ed.: Constraint-Handling in Evolutionary Optimization. Volume 198 of Studies in Computational Intelligence. Springer-Verlag Berlin Heidelberg (2009)

[22] Kuri-Morales, A., Quezada, C.V.: A Universal Eclectic Genetic Algorithm for Constrained Optimization. In: Proceedings 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT), Aachen, Germany, Verlag Mainz (1998) 518–522

[23] Homaifar, A., Lai, S.H.Y., Qi, X.: Constrained Optimization via Genetic Algorithms. Simulation **62**(4) (1994) 242–254

[24] Joines, J., Houck, C.: On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In Fogel, D., ed.: Proceedings of the first IEEE Conference on Evolutionary Computation, Orlando, Florida, IEEE Press (1994) 579–584

[25] Michalewicz, Z.: Genetic Algorithms, Numerical Optimization, and Constraints. In Eshelman, L.J., ed.: Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA), San Mateo, California, University of Pittsburgh, Morgan Kaufmann Publishers (July 1995) 151–158

[26] Michalewicz, Z., Attia, N.F.: Evolutionary Optimization of Constrained Problems. In: Proceedings of the 3rd Annual Conference on Evolutionary Programming, World Scientific (1994) 98–108

[27] Bean, J.C., Hadj-Alouane, A.B.: A Dual Genetic Algorithm for Bounded Integer Programs. Technical Report TR 92-53, Department of Industrial and Operations Engineering, The University of Michigan (1992)

[28] Hadj-Alouane, A.B., Bean, J.C.: A Genetic Algorithm for the Multiple-Choice Integer Program. Operations Research **45** (1997) 92–101

[29] Hoffmeister, F., Sprave, J.: Problem-independent handling of constraints by use of metric penalty functions. In Fogel, L.J., Angeline, P.J., Bäck, T., eds.: Proceedings of the Fifth Annual Conference on Evolutionary Programming (EP), San Diego, California, The MIT Press (February 1996) 289–294

[30] Ray, T., Tai, K., Seow, K.: Multiobjective design optimization by an evolutionary algorithm. Engineering Optimization **33**(4) (2001) 399–424

[31] Ho, P.Y., Shimizu, K.: Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme. Information Sciences **177** (2007) 2985–3004

[32] Mezura-Montes, E., Coello Coello, C.A.: Constrained Optimization via Multiobjective Evolutionary Algorithms. In Knowles, J., Corne, D., Deb, K., eds.: Multiobjective Problem Solving from Nature: From Concepts to Applications. Natural Computing Series. Springer-Verlag (2008) 53–75

[33] Vieira, D.A.G., Adriano, R.L.S., Krahenbuhl, L., Vasconcelos, J.A.: Handing constraints as objectives in a multiobjective genetic based algorithm. Journal of Microwaves and Optoelectronics **2**(6) (December 2002) 50–58

[34] Vieira, D.A.G., Adriano, R.L.S., Vasconcelos, J.A., Krahenbuhl, L.: Treating constraints as objectives in multiobjective optimization problems using niched pareto genetic algorithm. IEEE Transactions on Magnetics **40**(2) (Mar 2004) 1188 – 1191

[35] Coello Coello, C.A.: Constraint-handling using an evolutionary multiobjective optimization technique. Civil engineering and environmental systems **17**(4) (2000) 319–346

[36] Hamida, S.B., Schoenauer, M.: An adaptive algorithm for constrained optimization problems. In: Proceedings of Parallel Problem Solving from Nature (PPSN-VI), Lecture notes in Computer Science 1917, Springer Berlin/Heidelberg (2000) 529–538

[37] Hamida, S.B., Schoenauer, M.: ASCHEA: new results using adaptive segregational constraint handling. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC). (May 2002) 884–889

[38] Hinterding, R., Michalewicz, Z.: Your brains and my beauty: parent matching for constrained optimisation. Proceedings of IEEE Conference on Evolutionary Computaion (CEC) (May 1998) 810–815

[39] Mezura-Montes, E., Coello Coello, C.: A simple multimembered evolution strategy to solve constrained optimization problems. IEEE Transactions on Evolutionary Computation **9**(1) (Feb. 2005) 1–17

[40] Isaacs, A., Ray, T., Smith, W.: Blessings of maintaining infeasible solutions for constrained multi-objective optimization problems. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Hong Kong (June, 2008) 2785–2792

[41] Davis, L., ed.: Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York, New York (1991)

[42] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Third edn. Springer-Verlag (1996)

[43] Xiao, J., Michalewicz, Z., Zhang, L.: Evolutionary Planner/Navigator: Operator Performance and Self-Tuning. In: Proceedings of the 3rd IEEE International Conference on Evolutionary Computation, Nagoya, Japan, IEEE Press (May 1996)

[44] Xiao, J., Michalewicz, Z., Trojanowski, K.: Adaptive Evolutionary Planner/Navigator for Mobile Robots. IEEE Transactions on Evolutionary Computation **1**(1) (1997) 18–28

[45] Michalewicz, Z., Xiao, J.: Evaluation of Paths in Evolutionary Planner/Navigator. In: Proceedings of the 1995 International Workshop on Biologically Inspired Evolutionary Systems, Tokyo, Japan (May 1995) 45–52

[46] Michalewicz, Z., Nazhiyath, G.: Genocop III: A co-evolutionary algorithm for numerical optimization with nonlinear constraints. In Fogel, D.B., ed.: Proceedings of the Second IEEE International Conference on Evolutionary Computation, Piscataway, New Jersey, IEEE Press (1995) 647–651

[47] Surry, P.D., Radcliffe, N.J.: The COMOGA method: Constrained optimisation by multi-objective genetic algorithms. Control and Cybernetics **26**(3) (1997)

[48] Powell, D., Skolnick, M.M.: Using genetic algorithms in engineering design optimization with non-linear constraints. In Forrest, S., ed.: Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA), San Mateo, California, University of Illinois at Urbana-Champaign, Morgan Kaufmann Publishers (July 1993) 424–431

[49] Deb, K.: An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering **186** (2000) 311–338

[50] Suppapitnarm, A., Seffen, K., Parks, G., Clarkson, P.: A simulated annealing algorithm for multiobjective optimization. Engineering Optimization **33**(1) (2000) 59–85

[51] Nam, D., Park, C.: Multiobjective simulated annealing: A comparative study to evolutionary algorithms. International Journal of Fuzzy Systems **2**(2) (2000) 87–97

[52] Smith, K.I., Everson, R.M., Fieldsland, J.E., Murphy, C., Mishra, R.: Dominance-based multiobjective simulated annealing. IEEE Transactions on Evolutionary Compuration **12**(3) (2008) 323–342

[53] Smith, K.I.: A Study of Simulated Annealing Techniques for Multi-Objective Optimisation. PhD thesis, University of Exeter, Exeter, UK (2006)

[54] Suman, B.: Study of self-stopping PDMOSA and performance measure in multiobjective optimization. Computers and Chemical engineering **29** (2001) 1131–1147

[55] Michalewicz, Z., Schoenauer, M.: Evolutionary Algorithms for Constrained Parameter Optimization Problems. Evolutionary Computation **4**(1) (1996) 1–32

[56] Koziel, S., Michalewicz, Z.: Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. Evolutionary Computation **7**(1) (1999) 19–44

[57] Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley and Sons Pvt. Ltd. (2001)

[58] Wang, Y., Cai, Z., Guo, G., Zhou, Y.: Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics **37**(3) (June 2007) 560–575

[59] Takahama, T., Sakai, S.: Constrained optimization by applying the /spl alpha/ constrained method to the nonlinear simplex method with mutations. IEEE Transactions on Evolutionary Computation **9**(5) (Oct. 2005) 437–451

[60] Fletcher, R., Leyffer, S.: Nonlinear programming without a penalty function. Mathematical Programming **91** (2002) 239–269

[61] Hedar, A., Fukushima, M.: Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. Optimization Methods and Software **19**(3–4) (2004) 291–308

[62] Engrand, P.: A multi-objective approach based on simulated annealing and its application to nuclear fuel management. In: Proceedings of 5th International Conference on Nuclear Engineering, Nice, France (1997) 416–423

[63] Ulungu, E., Teghem, J., Fortemps, P., Tuyttens, D.: MOSA method: A tool for solving multiobjective combinatorial optimization problems. Journal of Multi-Criteria Decision Analysis **8** (1999) 221–236

[64] Smith, K., Everson, R., Fieldsend, J.: Dominance measures for multi-objective simulated annealing. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Portland, Oregon, USA (2004) 23–30

[65] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computing **6**(2) (2002) 182–197

[66] Knowles, J., Corne, D.: Approximating the nondominated front using the Pareto Archived Evolution Strategy. Evolutionary Computation **8**(2) (2000) 149–172

[67] Laguna, M., Marti, R.: Experimental testing of advanced scatter search designs for global optimization of multimodal functions. Journal of Global Optimization **33**(2) (2005) 235–255

[68] Laguna, M., Marti, R.: Scatter Search: Methodology and Implementations in C. Kluver Academic Publishers, Boston (2003)

[69] Ray, T., Gokarn, R., Sha, O.: A global optimization model for ship design. Computers in Industry **26**(2) (1995) 175–192

[70] Singh, H.K., Isaacs, A., Ray, T., Smith, W.: Infeasibility Driven Evolutionary Algorithm (IDEA) for Engineering Design Optimization. In: Proceedings of 21st Australiasian Joint Conference on Artificial Intelligence (AI). (2008) 104–115

[71] Zhang, Q., Zhou, A., Suganthan, P.N., Liu, W., Tiwari, S.: Multiobjective optimization test instances for the CEC 2009 special session and competition. Technical Report CES-487, The School of Computer Science and Electronic Engineering, University of Essex, Colchester, C04, 3SQ, UK (2009)

[72] Zhang, Q., Suganthan, P.N.: Final report on CEC '09 MOEA competition. Technical report, The School of CS and EE, University of Essex,UK and School of EEE, Nangyang Technological University, Singapore (2009)

[73] Singh, H.K., Isaacs, A., Nguyen, T.T., Ray, T., Yao, X.: Performance of Infeasibility Driven Evolutionary Algorithm (IDEA) on constrained dynamic single objective optimization problems. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Trondheim, Norway, IEEE Press (2009) 3127–3134

[74] Khare, V., Yao, X., Deb, K.: Performance scaling of multi-objective evolutionary algorithms. In: Proceedings of Evolutionary Multi-Criterion Optimization (EMO), Lecture notes in Computer Science 2632, Springer Berlin/Heidelberg (2003) 376–390

[75] Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: A short review. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Hong Kong (2008) 2424–2431

[76] Koppen, M., Yoshida, K.: Substitute distance assignments in NSGA-II for handling many-objective optimization problems. In: Proceedings of Evolutionary Multi-Criterion Optimization, Lecture notes in Computer Science 4403, Springer Berlin/Heidelberg (2007) 727–741

[77] Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. Proceedings of IEEE Congress on Evolutionary Computation (CEC) **1** (May 2002) 825–830

[78] Corne, D.W., Knowles, J.D.: Techniques for highly multiobjective optimisation: some nondominated points are better than others. In: Proceedings of the 9th annual conference on Genetic and Evolutionary Computation (GECCO), London, England, ACM, New York, NY, USA (2007) 773–780

[79] Sato, H., Aguirre, H., Tanaka, K.: Controlling dominance area of solutions and its impact on the performance of moeas. In: Proceedings of Evolutionary Multi-Criterion Optimization (EMO), Lecture notes in Computer Science 4403, Springer-Verlag, Berlin (2007) 5–20

[80] Zitzler, E., Kunzli, S.: Indicator-based selection in multiobjective search. In: Proceedings of Parallel Problem Solving from Nature (PPSN VIII), Lecture notes in Computer Science 3242), Springer Berlin/Heidelberg (2004) 832–842

[81] Deb, K., Sundar, J.: Reference point based multi-objective optimization using evolutionary algorithms. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO), New York, NY, USA, ACM (2006) 635–642

[82] Fleming, P., Purshouse, R., Lygoe, R.: Many-objective optimization: An engineering design perspective. In: Proceedings of Evolutionary Multi-Criterion Optimization (EMO), Lecture notes in Computer Science 3410, Springer Berlin/Heidelberg (2005) 14–32

[83] Obayashi, S., Sasaki, D.: Visualization and data mining of pareto solutions using self-organizing map. In: Proceedings of Evolutionary Multi-Criterion Optimization (EMO), Lecture notes in Computer Science 2632, Springer Berlin/Heidelberg (2003) 796–809

[84] Andy Pryke, Sanaz Mostaghim, A.N.: Heatmap visualization of population based multi objective algorithms. In: Proceedings of Evolutionary Multi-Criterion Optimization (EMO), Lecture notes in Computer Science 4403, Springer Berlin / Heidelberg (2007) 361–375

[85] Koppen, M., Yoshida, K.: Many-objective particle swarm optimization by gradual leader selection. In: Lecture notes in Computer Science 4431: Adaptive and Natural Computing Algorithms, Springer Berlin/Heidelberg (2007) 323–331

[86] Wagner, T., Beume, N., Naujoks, B.: Pareto-, aggregation-, and indicator-based methods in many-objective optimization. In: Proceedings of Evolutionary Multi-Criterion Optimization (EMO), Lecture notes in Computer Science 4403, Springer Berlin / Heidelberg (2007) 742–756

[87] Koppen, M., Vincente-Garcia, R., Nickolay, B.: Fuzzy-pareto-dominance and its application in evolutionary multi-objective optimization. In: Proceedings of

Evolutionary Multi-Criterion Optimization (EMO), Lecture notes in Computer Science 2632, Springer Berlin/Heidelberg (2005) 399–412

[88] Deb, K., Jain, S.: Running performance metrics for evolutionary multi-objective optimization. In: Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL). (2002) 13–20

[89] Hernandez-Daz, A.G., Santana-Quintero, L.V., Coello Coello, C.A., Molina, J. Evolutionary Computation **15**(4) 493–517

[90] Sammon, J.: A nonlinear mapping for data structure analysis. IEEE Transactions on Computers **18**(5) (1969) 401–409

[91] Saxena, D.K., Ray, T., Deb, K., Tiwari, A.: Constrained many-objective optimization: A way forward. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Trondheim, Norway (May 2009) 545–552

[92] Maaten, L., Postma, E., Herik, J.: Dimensionality reduction: A comparative review. Technical Report TiCC TR 2009-005, Tilburg centre for Creative Computing, Tillburg University, Netherlands (2009)

[93] Purshouse, R., Fleming, P.: Conflict, harmony and independence: Relationships in evolutionary multi-criterion optimisation. In: Proceedings of Evolutionary Multi-Criterion Optimization (EMO), Lecture Notes in Computer Science 2632, Springer-Berlin (2003) 16–30

[94] Gal, T., Leberling, H.: Redundant objective functions in linear vector maximum problems and theirdetermination. European Journal of Operational Research **1** (1977) 176–184

[95] Agrell, P.: On redundancy in multi criteria decision making. European Journal of Operational Research **98** (1997) 571–586

[96] Deb, K., Saxena, D.K.: Searching for pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Vancouver, Canada (2006) 3353–3360

[97] Saxena, D.K., Deb, K.: Non-linear dimensionality reduction procedures for certain large-dimensional multi-objective optimization problems: Employing correntropy and a novel maximum variance unfolding. In: Proceedings of Evolutionary Multi-Criterion Optimization (EMO), Lecture Notes in Computer Science 4403, Springer-Verlag Berlin Heidelberg (2007) 772–787

[98] Ray, T., Singh, H.K., Isaacs, A., Smith, W.: Infeasibility driven evolutionary algorithm for constrained optimization. In Mezura-Montes, E., ed.: Constraint Handling in Evolutionary Optimization. Studies in Computational Intelligence. Springer (2009) 145–165

[99] Saxena, D.K., Deb, K.: Trading on infeasibility by exploiting constraint's criticality through multi-objectivization: A system design perspective. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Singapore (25-28 Sept. 2007) 919–926

[100] Saxena, D.K., Deb, K.: Dimensionality reduction of objectives and constraints in multi-objective optimization problems: A system design perspective. In: Proceedings of 2008 IEEE Congress on Evolutionary Computation (CEC). (2008) 3204–3211

[101] Brockhoff, D., Zitzler, E.: Dimensionality reduction in multiobjective optimization with (partial) dominance structure preservation: Generalized minimum objective subset problems. Technical Report TIK-Report No. 247, ETH Zurich (2006)

[102] Brockhoff, D., Zitzler, E.: Are all objectives necessary ? On dimensionality reduction in evolutionary multiobjective optimization. In: Proceedings of Parallel Problem Solving fron Nature (PPSN IX), Springer-Verlag Berlin Heidelberg (2006) 533–542

[103] Brockhoff, D., Zitzler, E.: Objective reduction in evolutionary multiobjective optimization: Theory and applications. Evolutionary Computation **17**(2) (2009)

[104] Brockhoff, D., Zitzler, E.: Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Singapore (2007) 2086–2093

[105] Brockhoff, D., Zitzler, E.: Offline and online objective reduction in evolutionary multiobjective optimization based on objective conflicts. Technical Report TIK-Report No. 269, ETH Zurich (2007)

[106] Zitzler, E., Brockhoff, D., Thiele, L.: The hypervolume indicator revisited: On the design of pareto-compliant indicators via weigted integration. In: Proceedings of Evolutionary Multi-Criterion Optimization (EMO), Lecture notes in Computer Science 4403. (2007) 862–876

[107] James, A.L., Coello, C.A.C., Chakraborty, D.: Objective reduction using a feature selection technique. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO). (2008) 673–680

[108] Mitra, P., Murthy, C., Pal, S.: Unsupervised feature selection using feature similarity. IEEE Transactions on Pattern Analysis and Machine Intelligence **24**(3) 301–312

[109] Jaimes, A.L., Coello, C.A.C., Barrientos, J.E.U.: Online objective reduction to deal with many-objective problems. In: Proceedings of Evolutionary Multi-Criterion Optimization (EMO), Lecture Notes in Computer Science 5467, Springer-Verlag Heidelberg (2007) 423–437

[110] Bader, J., Zitzler, E.: HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. Technical Report TIK-Report No. 286, ETH Zurich (2008)

[111] Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation **10**(5) (2006) 477–506

[112] Musselman, K., Talavage, J.: A tradeoff cut approach to multiple objective optimization. Operations Research **28**(6) 1424–1435

[113] Hughes, E.J.: Radar waveform optimsation as a many-objective application benchmark. In: Proceedings of Evolutionary Multi-Criterion Optimization (EMO), Lecture notes in Computer Science 4403, Springer-Verlag Berlin Heidelberg (2007) 700–714

[114] Cao, Y.: Hypervolume indicator. `http://www.mathworks.com/matlabcentral/fileexchange/19651-hypervolume-indicator`

[115] Hughes, E.J.: Many-objective radar design software. `http://code.evanhughes.org`

[116] Potter, M.A., Jong, K.A.D.: A cooperative coevolutionary approach to function optimization. In: Proceedings of Parallel Problem Solving from Nature (PPSN III). (1994) 249–257

[117] Popovici, E., De Jong, K.: Relationships between internal and external metrics in co-evolution. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Edinburgh, UK (2005) 2800–2807

[118] Popovici, E., De Jong, K.: Understanding cooperative co-evolutionary dynamics via simple fitness landscapes. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), New York, NY, USA, ACM (2005) 507–514

[119] Wiegand, R.P., Liles, W.C., Jong, K.A.D.: An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), Morgan Kaufmann (2001) 1235–1242

[120] McNamara, J.M., Barta, Z., Fromhage, L., Houston, A.I.: The coevolution of choosiness and cooperation. Nature **451**(7175) (2008) 189–192

[121] Popovici, E., De Jong, K.: The effects of interaction frequency on the optimization performance of cooperative coevolution. In: Proceedings of Genetic and Evolutionary Computation (GECCO), Seattle, Washington, USA, ACM, New York, NY, USA (2006) 353–360

[122] Popovici, E., De Jong, K.: Sequential versus parallel cooperative coevolutionary algorithms for optimization. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Vancouver, Canada (2006) 1610–1617

[123] Tan, C., Goh, C., Tan, K., Tay, A.: A cooperative coevolutionary algorithm for multiobjective particle swarm optimization. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Singapore (2007) 3180–3186

[124] Li, X., Yao, X.: Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Trondheim, Norway (2009) 1546–1553

[125] Liu, B., Ma, H., Zhang, X.: A coevolutionary differential evolution algorithm for constrained optimization. In: Proceedings of International Conference on Natural Computation (ICNC), Los Alamitos, CA, USA (2007) 51–57

[126] Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Information Sciences **178**(15) (2008) 2985 – 2999

[127] Omidvar, M.N., Li, X.: Cooperative co-evolution for large scale optimization through more frequent random grouping. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Barcelona, Spain (2010) 1754–1761

[128] Omidvar, M.N., Li, X.: Cooperative co-evolution for with delta grouping for large scale non-separable function optimization. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Barcelona, Spain (2010) 1762–1769

[129] Suganthan, P.N., Hansen, N., Liang, J.J., Deb, K., Chen, Y.P., Auger, A., Tiwari, S.: Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical report, Nanyang Technological University, Singapore and Kanpur Genetic Algorithms Laboratory, IIT Kanpur, India (2005)

[130] Ray, T., Yao, X.: A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC), Trondheim, Norway (2009) 983–989

[131] Back, T., Schutz, M.: Evolution strategies for mixed-integer optimization of optical multilayer systems. In: Proceedings of Fourth Annual Conference on Evolutionary Programming (EP). (1995) 33–51

[132] Davidor, Y.: An evolution standing on the design of redundant robot manipulators. In: Proceedings of Parallel Problem Solving from Nature (PPSN I), 1st Workshop. (1991) 60–69

[133] Mandischer, M.: Evolving recurrent neural networks with non-binary encoding. In: Proceedings of IEEE International Conference on Evolutionary Computation. Volume 2., Perth, Australia (1995) 584–589

[134] Schutz, M., Sprave, J.: Application of parallel mixed-integer evolution strategies with mutation rate pooling. In: Proceedings of the Fifth Annual Conference on Evolutionary Programming (EP), The MIT Press (1996) 345–354

[135] Sprave, J., Rolf, S.: Variable-dimensional optimization with evolutionary algorithms using fixed-length representations. In: Proceedings of Seventh Annual Conference on Evolutionary Programming (EP). (1998) 261–269

[136] Green, P.: Reversible jump markov chain monte carlo computation and bayesian model determination. Biometrika **82** (1995) 711–732

[137] Brooks, S., Friel, N., R.King: Classical model selection via simulated annealing. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **65**(2) (May 2003) 503–520

[138] Kang, D., Verotta, D.: Reversible jump markov chain monte carlo for deconvolution. Journal of Pharmacokinetics and Pharmacodynamics **34**(3) (2007) 263–287

[139] Bandyopadhyay, S.: Simulated annealing using a reversible jump markov chain monte carlo algorithm for fuzzy clustering. IEEE Transactions On Knowledge and Data Engineering **17**(4) (2005) 479–490

[140] Hartigan, J.A.: Clustering Algorithms. John Wiley & Sons, Inc., New York, NY, USA (1975)

[141] Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1988)

[142] Davis, D., Bouldin, D.: A cluster separation measure. IEEE Transactions on Pattern Analysis and Machine Intelligence **1** (1979) 224–227

[143] Xie, X., Beni, G.: A validity measure for fuzzy clustering. IEEE Transactions on Pattern Analysis and Machine Intelligence **13**(8) (Aug 1991) 841–847

[144] Dunn, J.: A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. Cybernetics and Systems **3**(3) (1973) 32–57

[145] Calinski, R., Harabasz, J.: A dendrite method for cluster analysis. Communications in Statistics **3**(1) (1974) 1–27

[146] Pakhira, M., Bandyopadhyay, S., Maulik, U.: Validity index for crisp and fuzzy clusters. Pattern recognition **37**(3) (2004) 487–501

[147] Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics Bulletin **1**(6) (1945) 80–83

[148] Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. The Annals of Mathematical Statistics **18**(1) (1947) 50–60

[149] Isaacs, A.: Development of optimization methods to solve computationally expensive problems. PhD thesis, University of New South Wales, Australian Defence Force Academy (UNSW@ADFA), Canberra, Australia (2009)

[150] Myers, R.H., Montgomery, D.C.: Response Surface Methodology: Process and Product in Optimzation using Designed Experiments. John Wiley & Sons, Inc., NY, USA (1995)

[151] Powell, M.: Radial basis functions for multivariate interpolation: a review. In Mason, J., Cox, M., eds.: Algorithms for approximation. Oxford: Clarendon Press (1987) 143–167

[152] Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report C3P report 826, Caltech Concurrent Computation Program, Caltech, California, USA (1989)

[153] Ong, Y.S., Lim, M., Chen, X.: Research frontier: memetic computation - past, present and future. IEEE Computational Intelligence Magazine **5**(2) (May 2010) 24–31

[154] Powell, M.: A fast algorithm for nonlinearly constrained optimization calculations. In Watson, G., ed.: Numerical Analysis. Volume 630 of Lecture Notes in Mathematics. Springer (1978) 144–157

[155] Avriel, M., Williams, A.C.: An extension of geometric programming with applications in engineering optimization. Journal of Engineering Mathematics **5** (1971) 187–194 10.1007/BF01535411.

[156] Coello Coello, C.A.: Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems. Computers in Industry **41**(2) (January 2000) 113–127

[157] Parsons, M., Scott, R.: Formulation of multicriterion design optimization problems for solution with scalar numerical optimization methods. Journal of Ship Research **48**(1) (2004) 61–76

[158] Giannakoglou, K.C.: Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. Progress in Aerospace Sciences **38**(1) (2002) 43–76

[159] Coello Coello, C.A.: Treating constraints as objectives for single-objective evolutionary optimization. Engineering Optimization **32** (2000) 275–308

[160] Gu, L., Yang, R., Tho, C., Makowski, M., O.Faruque, Li, Y.: Optimisation and robustness for crashworthiness of side impact. International Journal of Vehicle Design **26**(4) (2001) 348 – 360

[161] Deb, K., Goyal, M.: A flexible optimization procedure for mechanical component design based on genetic adaptive search. Journal of Mechanical Design **120** (1998) 162–164

[162] Sandgren, E.: Nonlinear integer and discrete programming in mechanical design. In: Proceedings of the ASME Design Technology Conference, Kissimee, FL. (1988) 95–105

[163] Ray, T., Liew, K.: Society and civilization: An optimization algorithm based on the simulation of social behavior. IEEE Transactions on Evolutionary Computation **7**(4) (August 2003) 386 – 396

[164] Deb, K., Pratap, A., Meyarivan, T.: Constrained test problems for multi-objective evolutionary optimization. In: Proceedings of Evolutionary Multi-Criterion Optimization (EMO), Lecture Notes in Computer Science 1993. (2001) 284–298

[165] Sen, P., Yang, J.: Multiple criteria decision support in engineering design. Springer-Verlag, London (1998)

[166] Jameson, A., Schmidt, W., Turkel, E.: Numerical solutions of the euler equations by finite volume methods using Runge-Kutta Time-Stepping schemes. In: Proceedings of the AIAA 14th Fluid and Plasma Dynamic Conference, Palo Alto (1981)

Appendix A

g-series optimization problems

The g-series of problems consists of constrained single objective problems [55, 56]. The g-series has a total of eleven problems out of which nine problems without equality constraints are studied.

A.1 g01

$$\text{Minimize} \quad f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

subject to

$$\begin{aligned}
g_1(x) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \\
g_2(x) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \\
g_3(x) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \\
g_4(x) &= -8x_1 + x_{10} \leq 0 \\
g_5(x) &= -8x_2 + x_{11} \leq 0 \\
g_6(x) &= -8x_3 + x_{12} \leq 0 \\
g_7(x) &= -2x_4 - x_5 + x_{10} \leq 0 \\
g_8(x) &= -2x_6 - x_7 + x_{11} \leq 0 \\
g_9(x) &= -2x_8 - x_9 + x_{12} \leq 0
\end{aligned}$$

where the bounds are $0 \leq x_i \leq 1$ $(i = 1, \cdots, 9)$, $0 \leq x_i \leq 100$ $(i = 10, 11, 12)$, and $0 \leq x_{13} \leq 1$. The global minimum is at $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$

where six constrains are active $(g_1, g_2, g_3, g_7, g_8, g_9)$ and $f(x^*) = -15$.

A.2 g02

$$\text{Maximize} \quad f(x) = \left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

subject to

$$g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$$

$$g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0$$

where $n = 20$ and $0 \leq x_i \leq 10$ $(i = 1, \ldots, n)$. The global maximum is unknown; the best found is $f(x^*) = 0.803619$.

A.3 g04

$$\text{Minimize} \quad f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

subject to

$$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$$
$$g_2(x) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$$
$$g_3(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$$
$$g_4(x) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$$
$$g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$$
$$g_6(x) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$$

where $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$ and $27 \leq x_i \leq 45$ $(i = 3, 4, 5)$. The optimum solution is $x^* = (78, 33, 29.99526025682, 45, 36.775812905788)$ where $f(x^*) = -30665.539$. Two constraints are active (g_1 and g_6).

A.4 g06

$$\text{Minimize} \quad f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

subject to

$$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$$
$$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$$

where $13 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$. The optimum solution is $x^* = (14.095, 0.84296)$ where $f(x^*) = -6961.81388$. Both constraints are active.

A.5 g07

$$\begin{aligned}\text{Minimize} \quad f(x) = {} & x_1^2 + x_2^2 + x_1 x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 \\ & + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 \\ & + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45\end{aligned}$$

subject to

$$\begin{aligned}g_1(x) &= -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0 \\ g_2(x) &= 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0 \\ g_3(x) &= -8x_1 - 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0 \\ g_4(x) &= 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0 \\ g_5(x) &= 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0 \\ g_6(x) &= x_1^2 + 2(x_2 - 2)^2 - 2x_1 x_2 + 14x_5 - 6x_6 \leq 0 \\ g_7(x) &= 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0 \\ g_8(x) &= -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0\end{aligned}$$

where $-10 \leq x_i \leq 10$ $(i = 1, \ldots, 10)$. The optimum solution is $x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$ where $f(x^*) = 24.3062091$. Six constraints are active $(g1, g2, g3, g4, g5, g6)$.

A.6 g08

$$\text{Maximize} \quad f(x) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)}$$

subject to

$$g_1(x) = x_1^2 - x_2 + 1 \leq 0$$
$$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0$$

where $0 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 10$. The optimum is located at $x^* = (1.2279713, 4.2453733)$ where $f(x^*) = 0.095825$. The solution lies within the feasible region.

A.7 g09

$$\begin{aligned}\text{Minimize} \quad f(x) = {} & (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 \\ & + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7\end{aligned}$$

subject to

$$g_1(x) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 - 5x_5 \leq 0$$
$$g_2(x) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$$
$$g_3(x) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$$
$$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

where $-10 \leq x_i \leq 10$ $(i = 1, \ldots, 7)$. The optimum solution is $x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$ where $f(x^*) = 680.6300573$. Two constraints are active (g_1 and g_4).

A.8 g10

$$\text{Minimize} \quad f(x) = x_1 + x_2 + x_3$$

subject to

$$\begin{aligned}
g_1(x) &= -1 + 0.0025(x_4 + x_6) \leq 0 \\
g_2(x) &= -1 + 0.0025(x_5 + x_7 - x_4) \leq 0 \\
g_3(x) &= -1 + 0.01(x_8 - x_5) \leq 0 \\
g_4(x) &= -x_1 x_6 + 833.33252 x_4 + 100 x_1 - 83333.333 \leq 0 \\
g_5(x) &= -x_2 x_7 + 1250 x_5 + x_2 x_4 - 1250 x_4 \leq 0 \\
g_6(x) &= -x_3 x_8 + 1250000 + x_3 x_5 - 2500 x_5 \leq 0
\end{aligned}$$

where $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000$ $(i = 2, 3)$, and $10 \leq x_i \leq 1000$ $(i = 4, \ldots, 8)$. The optimum solution is $x^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)$ where $f(x^*) = 7049.3307$. Three constraints are active (g_1, g_2, and g_3).

A.9 g12

$$\text{Maximize} \quad f(x) = (100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$$

subject to

$$g_1(x) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$$

where $0 \leq x_i \leq 10$ $(i = 1, 2, 3)$ and $p, q, r = 1, 2, \ldots, 9$. The feasible region of the search space consists of 9^3 disjointed spheres. The optimum is located at $x^* = (5, 5, 5)$ where $f(x^*) = 1$. The solution lies within the feasible region.

Appendix B

CTP problems

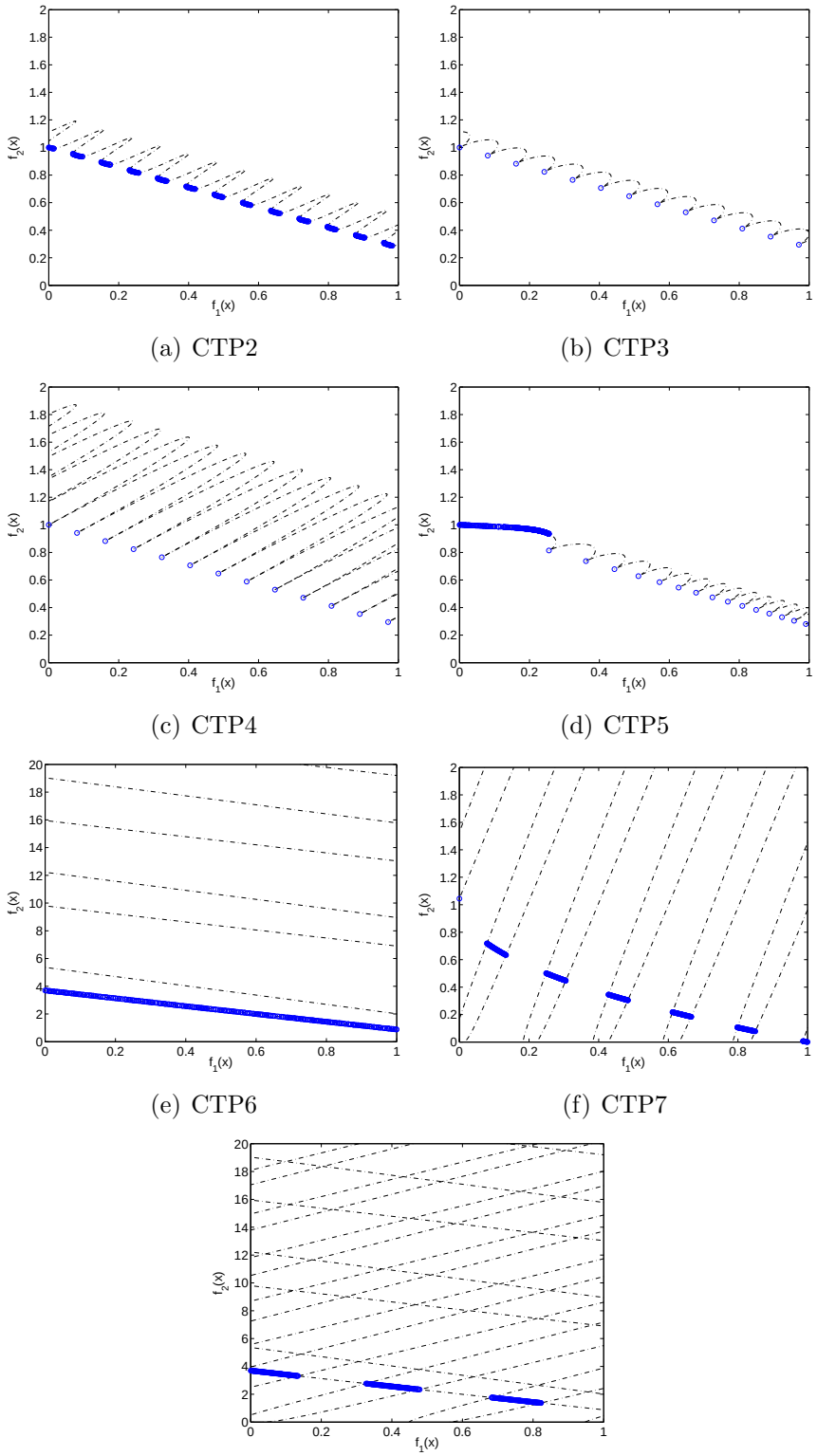
Deb *et al.* [164, 57] proposed constrained bi-objective test problems (CTP). The test functions CTP2-CTP7 have a single constraint and CTP8 has two constraints. The mathematical formulation of CTP2-CTP8 is given in Equation B.1. Both the constraints of CTP8 are of the same form as the single constraint in other CTP problems.

$$\begin{aligned}
&\text{Min. } f_1(\mathbf{x}) = x_1, \\
&\text{Min. } f_2(\mathbf{x}) = C(\mathbf{x}) \left(1 - \sqrt{f_1(\mathbf{x})/C(\mathbf{x})}\right), \\
&\quad g(\mathbf{x}) = \cos(\theta)\left(f_2(\mathbf{x}) - e\right) - \sin(\theta) f_1(\mathbf{x}) \geq \\
&\qquad\qquad a|\sin\left(b\pi\left(\sin(\theta)\left(f_2(\mathbf{x}) - e\right) + \cos(\theta) f_1(\mathbf{x})\right)^c\right)|^d, \\
&\quad C(\mathbf{x}) = 1 + \sum_{i=2}^{10} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right),
\end{aligned} \tag{B.1}$$

where $0 \leq x_i \leq 1$, $i = 1, \ldots, 10$, and $C(\mathbf{x})$ is the generalized Rastrigin function. The constraint parameters θ, a, b, c, d, e for CTP2 to CTP8 are listed in Table B.1.

Table B.1: Parameters for the test problems CTP2 to CTP8

	θ	a	b	c	d	e
CTP2	-0.20π	0.20	10.0	1	6.0	1
CTP3	-0.20π	0.10	10.0	1	0.5	1
CTP4	-0.20π	0.75	10.0	1	0.5	1
CTP5	-0.20π	0.75	10.0	2	0.5	1
CTP6	0.10π	40.00	0.5	1	2.0	-2
CTP7	-0.05π	40.00	5.0	1	6.0	0
CTP8	0.10π	40.00	0.05	1	2.0	-2
	-0.05π	40.00	2.0	1	6.0	0



(a) CTP2 (b) CTP3

(c) CTP4 (d) CTP5

(e) CTP6 (f) CTP7

(g) CTP8

Figure B.1: Pareto fronts for CTP2-CTP8 problems

Appendix C

CF problems

CF series of multi-objective test problems were proposed by Zhang *et al.*[71] for the IEEE Congress on Evolutionary Computation (CEC) 2009 algorithm contest. The mathematical formulation of the problems CF1-CF7, studied in this thesis, are given in the following sections. Their Pareto fronts are shown in Figure C.1.

C.1 CF1

$$
\begin{aligned}
&\text{Min. } f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}\left(x_j - x_1^{0.5(1.0+\frac{3(j-2)}{n-2})}\right)^2 \\
&\text{Min. } f_2(\mathbf{x}) = 1 - x_1 + \frac{2}{|J_2|}\sum_{j\in J_1}\left(x_j - x_1^{0.5(1.0+\frac{3(j-2)}{n-2})}\right)^2 \\
&\quad \text{where } \ J_1 = \{j|j \text{ is odd and } 2 \le j \le n\} \\
&\quad\quad \text{and } J_2 = \{j|j \text{ is even and } 2 \le j \le n\} \\
&\text{Subject to} \\
&\quad g(\mathbf{x}) = f_1 + f_2 - a|sin|\left[N\pi(f_1 - f_2 + 1)\right]| - 1 \ge 0 \\
&\quad \text{where } N \in \mathbb{Z}, a \ge \frac{1}{2N} \\
&\quad\quad 0 \le x_i \le 1, i = 1, \ldots, n \\
&\quad\quad N = 10; a = 1; n = 10
\end{aligned}
\tag{C.1}
$$

C.2 CF2

$$\begin{aligned}
&\text{Min. } f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}\left(x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right)\right)^2 \\
&\text{Min. } f_2(\mathbf{x}) = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}\left(x_j - \cos\left(6\pi x_1 + \frac{j\pi}{n}\right)\right)^2 \\
&\quad \text{where } J_1 = \{j | j \text{ is odd and } 2 \le j \le n\} \\
&\quad\quad \text{and } J_2 = \{j | j \text{ is even and } 2 \le j \le n\}
\end{aligned} \tag{C.2}$$

Subject to

$$\begin{aligned}
g(\mathbf{x}) =& \frac{f_2 + \sqrt{f_1} - a\sin[N\pi(\sqrt{f_1} - f_2 + 1)] - 1}{1 + e^{4\|t\|}} \ge 0 \\
& 0 \le x_1 \le 1; -1 \le x_i \le 1, i = 2, \ldots, n \\
& N = 2; a = 1; n = 10
\end{aligned}$$

C.3 CF3

$$\begin{aligned}
&\text{Min. } f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|}\left(4\sum_{j} \in J_1 y_j^2 - 2\prod_{j\in J_1}\cos(\frac{20y_i\pi}{\sqrt{j}}) + 2\right) \\
&\text{Min. } f_2(\mathbf{x}) = x_1 + \frac{2}{|J_2|}\left(4\sum_{j} \in J_2 y_j^2 - 2\prod_{j\in J_2}\cos(\frac{20y_i\pi}{\sqrt{j}}) + 2\right) \\
&\quad \text{where } J_1 = \{j | j \text{ is odd and } 2 \le j \le n\} \\
&\quad\quad J_2 = \{j | j \text{ is even and } 2 \le j \le n\} \\
&\quad\quad \text{and } y_j = x_j - \sin\left(6\pi x_1 + \frac{j\pi}{n}\right), j = 2, \ldots, n
\end{aligned} \tag{C.3}$$

Subject to

$$\begin{aligned}
g(\mathbf{x}) =& f_2 + f_1^2 - a\sin[N\pi(f_1^2 - f_2 + 1)] - 1 \ge 0 \\
& 0 \le x_1 \le 1; -2 \le x_i \le 2, i = 2, \ldots, n \\
& N = 2; a = 1; n = 10
\end{aligned}$$

C.4 CF4

$$\begin{aligned}
&\text{Min. } f_1(\mathbf{x}) = x_1 + \sum_{j\in J_1} h_j(y_i)\\
&\text{Min. } f_2(\mathbf{x}) = 1 - x_1 + \sum_{j\in J_2} h_j(y_i)\\
&\quad \text{where } \ J_1 = \{j | j \text{ is odd and } 2 \le j \le n\}\\
&\qquad\qquad J_2 = \{j | j \text{ is even and } 2 \le j \le n\}\\
&\qquad\qquad h_2(t) = \begin{cases} |t| & \text{if } t < \frac{3}{2}(1 - 1/\sqrt{2})\\ 0.125 + (t-1)^2 & \text{otherwise}\end{cases}\\
&\qquad \text{and } h_j(t) = t^2 \text{ for } j = 3, \dots n
\end{aligned} \tag{C.4}$$

Subject to

$$\begin{aligned}
g(\mathbf{x}) =& \frac{t}{1 + e^{4|t|}} \ge 0\\
t =& x_2 - \sin(6\pi x_1 + 2\pi/n) - 0.5x_1 + 0.25\\
&0 \le x_1 \le 1; -2 \le x_i \le 2, i = 2, \dots, n; n = 10
\end{aligned}$$

C.5 CF5

$$\begin{aligned}
&\text{Min. } f_1(\mathbf{x}) = x_1 + \sum_{j\in J_1} h_j(y_i)\\
&\text{Min. } f_2(\mathbf{x}) = 1 - x_1 + \sum_{j\in J_2} h_j(y_i)\\
&\quad \text{where } \ J_1 = \{j | j \text{ is odd and } 2 \le j \le n\}\\
&\qquad\qquad J_2 = \{j | j \text{ is even and } 2 \le j \le n\}\\
&\qquad\qquad y_j = \begin{cases} x_j - 0.8x_1\cos(6\pi x_1 + j\pi/n) & \text{if } j \in J_1\\ x_j - 0.8x_1\sin(6\pi x_1 + j\pi/n) & \text{if } j \in J_2\end{cases}\\
&\qquad\qquad h_2(t) = \begin{cases} |t| & \text{if } t < \frac{3}{2}(1 - 1/\sqrt{2})\\ 0.125 + (t-1)^2 & \text{otherwise}\end{cases}\\
&\qquad \text{and } h_j(t) = 2t^2 - \cos(4\pi t) + 1 \text{ for } j = 3, \dots n
\end{aligned} \tag{C.5}$$

Subject to

$$\begin{aligned}
g(\mathbf{x}) =& x_2 - 0.8x_1\sin(6\pi x_1 + 2\pi/n) - 0.5x_1 + 0.25 \ge 0\\
&0 \le x_1 \le 1; -2 \le x_i \le 2, i = 2, \dots, n; n = 10
\end{aligned}$$

C.6 CF6

$$\text{Min. } f_1(\mathbf{x}) = x_1 + \sum_{j \in J_1} (y_j)^2$$

$$\text{Min. } f_2(\mathbf{x}) = x_1 + \sum_{j \in J_2} (y_j)^2$$

$$\text{where } J_1 = \{j | j \text{ is odd and } 2 \leq j \leq n\}$$

$$J_2 = \{j | j \text{ is even and } 2 \leq j \leq n\}$$

$$y_j = \begin{cases} x_j - 0.8x_1 \cos(6\pi x_1 + j\pi/n) & \text{if } j \in J_1 \\ x_j - 0.8x_1 \sin(6\pi x_1 + j\pi/n) & \text{if } j \in J_2 \end{cases}$$

Subject to

$$g_1(\mathbf{x}) = x_2 - 0.8x_1 \sin(6\pi x_1 + 2\pi/n) - \text{sign}(0.5(1-x_1) - (1-x_1)^2) \sqrt{|0.5(1-x_1) - (1-x_1)^2|} \geq 0$$

$$g_2(\mathbf{x}) = x_4 - 0.8x_1 \sin(6\pi x_1 + 4\pi/n) - \text{sign}(0.25\sqrt{1-x_1} - 0.5(1-x_1)) \sqrt{|0.25\sqrt{1-x_1} - 0.5(1-x_1)|} \geq 0$$

$$0 \leq x_1 \leq 1; -2 \leq x_i \leq 2, i = 2, \ldots, n; n = 10 \tag{C.6}$$

C.7 CF7

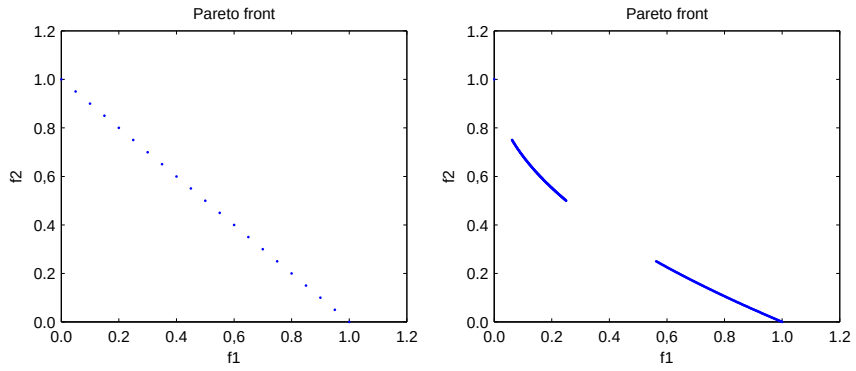
$$\text{Min. } f_1(\mathbf{x}) = x_1 + \sum_{j \in J_1} h_j(y_j)$$

$$\text{Min. } f_2(\mathbf{x}) = x_1 + \sum_{j \in J_2} h_j(y_j)$$

$$\begin{aligned}
\text{where } \; & J_1 = \{j | j \text{ is odd and } 2 \leq j \leq n\} \\
& J_2 = \{j | j \text{ is even and } 2 \leq j \leq n\} \\
& y_j = \begin{cases} x_j - \cos(6\pi x_1 + j\pi/n) & \text{if } j \in J_1 \\ x_j - \sin(6\pi x_1 + j\pi/n) & \text{if } j \in J_2 \end{cases} \\
& h_2(t) = h_4(t) = t^2 \\
& h_j(t) = 2t^2 - \cos(4\pi t) + 1 \text{ for } j = 3, 5, 6, \ldots, n
\end{aligned}$$

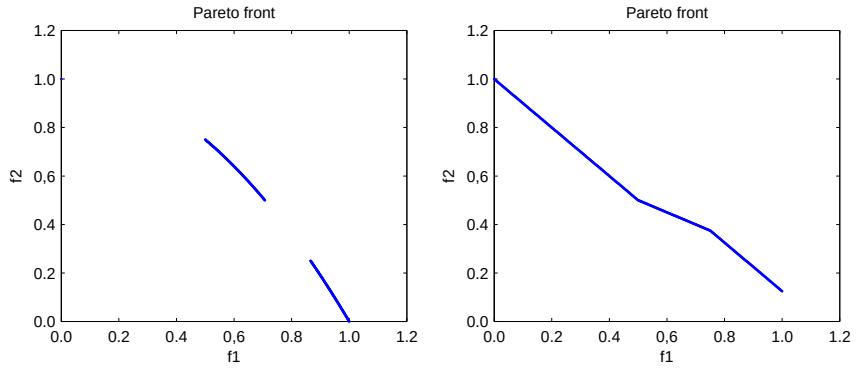
Subject to

$$\begin{aligned}
g_1(\mathbf{x}) = & x_2 - \sin(6\pi x_1 + 2\pi/n) - \\
& \text{sign}(0.5(1 - x_1) - (1 - x_1)^2)\sqrt{|0.5(1 - x_1) - (1 - x_1)^2|} \geq 0 \\
g_2(\mathbf{x}) = & x_4 - \sin(6\pi x_1 + 4\pi/n) - \\
& \text{sign}(0.25\sqrt{1 - x_1} - 0.5(1 - x_1))\sqrt{|0.25\sqrt{1 - x_1} - 0.5(1 - x_1)|} \geq 0 \\
& 0 \leq x_1 \leq 1; -2 \leq x_i \leq 2, i = 2, \ldots, n; n = 10
\end{aligned} \tag{C.7}$$



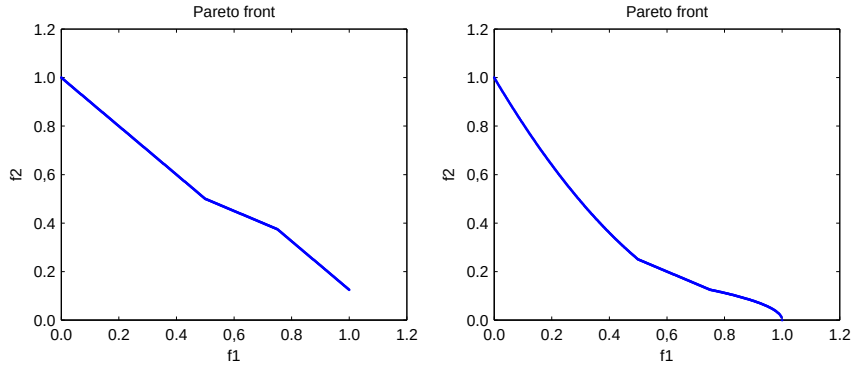
(a) CF1

(b) CF2



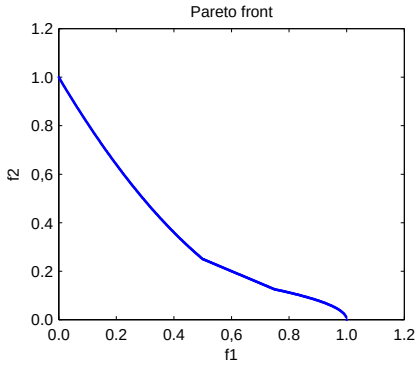
(c) CF3

(d) CF4



(e) CF5

(f) CF6



(g) CF7

Figure C.1: Pareto fronts for CF1-CF7 problems

Appendix D

DTLZ problems

A multi-objective test problem suite, DTLZ, was formulated by Deb *et al.* [77]. The problems of the suite are scalable both in terms of objectives and variables. Two problems from the test suite are studied in this thesis. They are described in next two sections.

D.1 DTLZ2

An M-objective DTLZ2 problem is formulated as follows.

$$
\begin{aligned}
&\text{Minimize} \\
&f_1(\mathbf{x}) = r(\mathbf{x}_M)\cos(\pi x_1/2)\cdots\cos(\pi x_{M-2}/2)\cos(\pi x_{M-1}/2), \\
&f_2(\mathbf{x}) = r(\mathbf{x}_M)\cos(\pi x_1/2)\cdots\cos\pi(x_{M-2}/2)\sin(\pi x_{M-1}/2), \\
&f_3(\mathbf{x}) = r(\mathbf{x}_M)\cos(\pi x_1/2)\cdots\sin(\pi x_{M-2}/2), \\
&\vdots \\
&f_{M-1}(\mathbf{x}) = r(\mathbf{x}_M)\cos(\pi x_1/2)\sin(\pi x_2/2), \\
&f_M(\mathbf{x}) = r(\mathbf{x}_M)\sin(\pi x_1/2), \\
&\text{where} \\
&r(\mathbf{x}_M) = 1 + g(\mathbf{x}_M) = 1 + \textstyle\sum_{x_i \in \mathbf{x}_M}(x_i - 0.5)^2, \\
&0 \le x_i \le 1, \quad \text{for } i = 1, 2, \ldots, n.
\end{aligned}
\tag{D.1}
$$

The total number of variables involved is $n = M + k - 1$. A prescribed value of $k = 10$ is used. The Pareto-optimal solutions correspond to $x_M^* = 0.5$. The Pareto front for an M-objective DTLZ2 problem is M-dimensional, *i.e.* none of the objectives are redundant in the problem. In the objective space, the Pareto

front corresponds to the positive the set of solutions with $\sum_{i=1}^{M} f_i^2 = 1, f_i > 0$ for $i = 1, \ldots, M$, which, for a 3-objective problem, is shown in Figure D.1. The notation DTLZ2(M) is used to refer to an M-dimensional DTLZ2 problem.

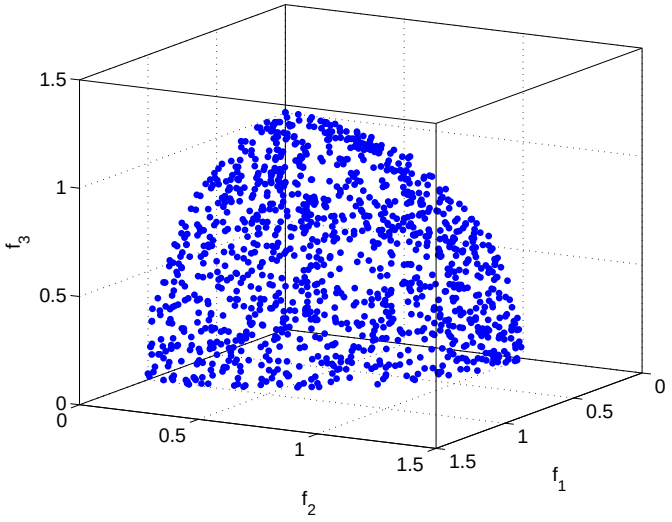


Figure D.1: Pareto front for 3-objective DTLZ2 problem

D.2 DTLZ3

An M-objective DTLZ3 problem is formulated as follows.

Minimize
$f_1(\mathbf{x}) = r(\mathbf{x}_M)\cos(\pi x_1/2)\cdots\cos(\pi x_{M-2}/2)\cos(\pi x_{M-1}/2),$
$f_2(\mathbf{x}) = r(\mathbf{x}_M)\cos(\pi x_1/2)\cdots\cos\pi(x_{M-2}/2)\sin(\pi x_{M-1}/2),$
$f_3(\mathbf{x}) = r(\mathbf{x}_M)\cos(\pi x_1/2)\cdots\sin(\pi x_{M-2}/2),$
$\vdots$
$f_{M-1}(\mathbf{x}) = r(\mathbf{x}_M)\cos(\pi x_1/2)\sin(\pi x_2/2),$
$f_M(\mathbf{x}) = r(\mathbf{x}_M)\sin(\pi x_1/2),$
where
$r(\mathbf{x}_M) = 1 + g(\mathbf{x}_M) = 1 + 100\left[|\mathbf{x}_M| + \sum_{x_i\in\mathbf{x}_M}(x_i - o.5)^2 - \cos(20\pi(x_i - 0.5))\right]$
$0 \le x_i \le 1, \quad \text{for } i = 1, 2, \ldots, n.$
(D.2)

The total number of variables involved is $n = M + k - 1$. A prescribed value of $k = 10$ is used. The g function used in the formulation introduces (3^k-1) local fronts, posing significant challenge to the multi-objective algorithms. The Pareto front correspond to the solutions with $x^*_M = 0.5$, where $g^* = 0$. The Pareto front of a DTLZ3 problem identical to that of a DTLZ2 of same dimension.

D.3 DTLZ5-(I,M)

Another function from the DTLZ test suite, DTLZ5 was later modified to construct a set of test problems in which the dimensionality of the Pareto front is less than the original number of objectives [96]. These test problems are known as DTLZ5-(I, M) problems. Here, I denotes the actual dimensionality of the Pareto front and M the original number of objectives for the problem. The intent for formulating these problems was to critically evaluate dimensionality reduction techniques for many-objective optimization problems. The formulation of DTLZ5-(I, M) is given in Equation D.3.

$$
\begin{aligned}
&\text{Minimize} \\
&f_1(\mathbf{x}) = r(\mathbf{x}_M)\cos(\theta_1)\cdots\cos(\theta_{M-2})\cos(\theta_{M-1}), \\
&f_2(\mathbf{x}) = r(\mathbf{x}_M)\cos(\theta_1)\cdots\cos(\theta_{M-2})\sin(\theta_{M-1}), \\
&f_3(\mathbf{x}) = r(\mathbf{x}_M)\cos(\theta_1)\cdots\sin(\theta_{M-2}), \\
&\vdots \\
&f_{M-1}(\mathbf{x}) = r(\mathbf{x}_M)\cos(\theta_1)\sin(\theta_2), \\
&f_M(\mathbf{x}) = r(\mathbf{x}_M)\sin(\theta_1), \\
&\text{where} \\
&r(\mathbf{x}_M) = 1 + g(\mathbf{x}_M) = 1 + \textstyle\sum_{x_i \in \mathbf{x}_M}(x_i - 0.5)^2, \\
&\theta_i = \begin{cases} \frac{\pi}{2}\, x_i, & \text{for } i = 1, \ldots, (I-1), \\ \frac{\pi}{4r(\mathbf{x}_M)}\,(1 + 2g(\mathbf{x}_M)x_i), & \text{for } i = I, \ldots, (M-1) \end{cases} \\
&\text{Subject to:} \\
&\textstyle\sum_{j=0}^{I-2} f_{M-j}^2 + 2^{p_i} f_i^2 \geq 1, \text{ for } i = 1, 2 \ldots (M - I + 1) \\
&\text{where} \\
&p_i = \begin{cases} M - I, \text{ for } i = 1; \\ (M - I + 2) - i, & \text{for } i = 2 \ldots (M - I + 1) \end{cases} \\
&0 \leq x_i \leq 1, \quad \text{for } i = 1, 2, \ldots, n.
\end{aligned}
\tag{D.3}
$$

The total number of variables involved is $n = M+k-1$, where, $k = |\mathbf{x}_M| = 10$ is

prescribed. In addition, there are $M-I+1$ constraints to be satisfied, as shown in Equation D.3. The problem is designed so that the Pareto front corresponds to: (i) a zero value of the g function, in turn implying $x_i = 0.5$ for $i = M, \ldots, (M+k-1)$; (ii) a fixed value of $\pi/4$ for the variables x_I to x_{M-1}; and (iii) independent values for the variables x_1 to x_{I-1}. Hence, by simply setting I to an integer between 2 and M, the dimensionality (I) of the Pareto front can be controlled. Pareto front for DTLZ5-(2,3) is shown in Figure D.2.

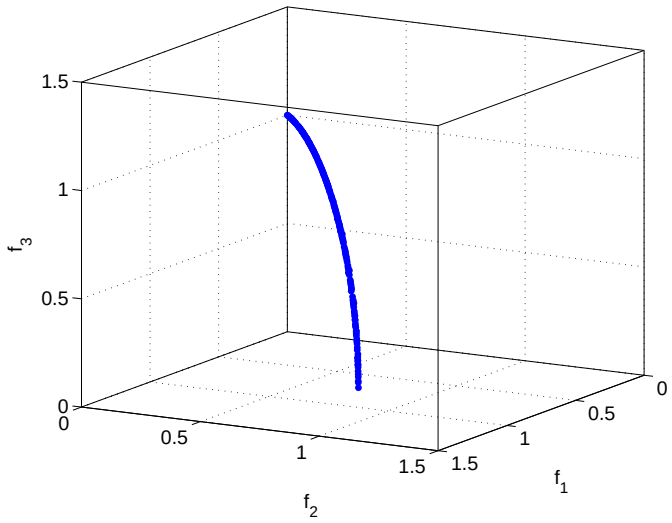


Figure D.2: Pareto front for DTLZ5-(2,3) problem

Appendix E

Engineering problems

E.1 Belleville Spring Design

The Belleville spring design problem is to find the minimum weight spring subject to the stress and displacement constraints [3]. The configuration of the Belleville spring is shown in Figure E.1. The design variables are external diameter (D_e), internal diameter (D_i), thickness t, and free height h. The variable bounds are $0.01 \leq D_e \leq 6.0$, $0.05 \leq D_i \leq 0.50$, $5.0 \leq t \leq 15.0$ and $5.0 \leq h \leq 15.0$. Other parameters used in the design are listed in Table E.1.

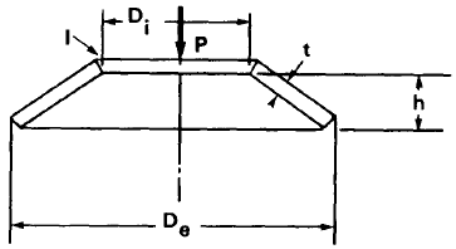


Figure E.1: Belleville spring configuration (Source: [3])

The Belleville spring design optimization problem is defined as follows:

Table E.1: Parameters for Belleville spring design

Parameter	Value
Maximum load (P_{max})	5400 lb
Maximum deflection (δ_{max})	0.2 in
Maximum outside diameter (D_{max})	12.01 in
Maximum allowable stress (σ_D)	200000 psi
Maximum total height (l)	2.0 in

$$
\begin{aligned}
\text{Minimize} \quad & f = \frac{0.283\pi(D_e^2 - D_i^2)t}{4} \\
\text{subject to} \quad & g_1 = \sigma_D - \sigma \geq 0 \\
& g_2 = P - P_{max} \geq 0 \\
& g_3 = \delta_l - \delta_{max} \geq 0 \\
& g_4 = l - h - t \geq 0 \\
& g_5 = D_{max} - D_e \geq 0 \\
& g_6 = D_e - D_i \geq 0 \\
& g_7 = 0.3 - \frac{h}{D_e - D_i} \geq 0
\end{aligned}
$$

where σ is the stress at the upper edge (marked I in Figure E.1), P is the load corresponding to the limiting deflection δ_{max}. The stress σ on the upper edge is given by

$$\sigma = \frac{E\delta_{max}}{(1-\mu^2)\alpha(D_e/2)^2}\left[\beta\left(h - \frac{\delta_{max}}{2}\right) + \gamma t\right].$$

The load P corresponding to the limiting deflection δ_{max} is given by

$$P = \frac{E\delta_{max}}{(1-\mu^2)\alpha a^2}\left[\left(h - \frac{\delta}{2}\right)(h-\delta)t + t^3\right].$$

The geometric parameters α, β and γ are defined in terms of $K = D_e/D_i$ as follows:

$$\alpha = \frac{6}{\pi \ln K}\left(\frac{K-1}{K}\right)^2, \quad \beta = \frac{6}{\pi \ln K}\left(\frac{K-1}{\ln K} - 1\right), \quad \gamma = \frac{6}{\pi \ln K}\left(\frac{K-1}{2}\right)$$

E.2 Design of Coil Compression Spring

A tension/compression spring is shown in Figure E.2. The design of the spring involves minimizing the weight of the spring subject to constraints on minimum deflection, shear stress, surge frequency [3]. The design variables are the mean coil diameter D, the wire diameter d and the number of active coils N.

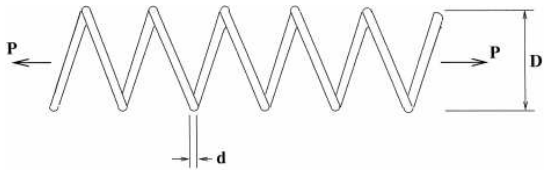


Figure E.2: Tension/Compression spring

The optimization problem is defined as follows:

$$
\begin{aligned}
\text{Minimize} \quad & f = \frac{\pi^2}{4}(N+2)Dd^2 \\
\text{subject to} \quad & g_1 = S - C_f 8F_{max}\frac{D}{\pi d^3} \geq 0 \\
& g_2 = l_{max} - l_f \geq 0 \\
& g_3 = d - d_{min} \geq 0 \\
& g_4 = D_{max} - D - d \geq 0 \\
& g_5 = C - 3 \geq 0 \\
& g_6 = \delta_{pm} - \frac{F_p}{K} \geq 0 \\
& g_7 = l_f - \delta_p - \frac{F_{max} - F_p}{K} - 1.05(N+2)d \geq 0 \\
& g_8 = \frac{F_{max} - F_p}{K} - \delta_w \geq 0 \\
& g_9 = \frac{P_{crit}}{1.25} - F_{max} \geq 0
\end{aligned}
$$

where g_1 is a stress constraint, g_2–g_5 are geometry constraints, g_6–g_7 are deflection constraints and g_9 is a buckling constraint. The variables used in the constraints are defined as follows:

$$
C = \frac{D}{d}, \quad \delta = \frac{F_{max}}{K}, \quad K = \frac{Gd^4}{8ND^3}, \quad C_f = \frac{4C-1}{4C-4} + \frac{0.615}{C},
$$

$$l_f = \delta + 1.05(N+2)d, \quad P_{crit} = \frac{l_f}{K}$$

The other parameters required for the design are specified in Table E.2. The variable bounds are $0.05 \leq D \leq 2.0$, $0.25 \leq d \leq 1.3$, $2 \leq N \leq 20$.

Table E.2: Parameters for coil compression spring design

Parameter	Value
Maximum working load (F_{max})	1000.0 lb
Maximum free length (l_{max})	14.0 in
Minimum wire diameter (d_{min})	0.2 in
Maximum outer diameter (D_{max})	3.0 in
Preload compression force (F_p)	300.0 lb
Maximum allowable deflection under preload (δ_{pm})	6.0 in
Deflection from preload to maximum load (δ_w)	1.25 in
Maximum allowable shear stress (S)	189000 psi
Shear modulus of the material (G)	1.15e7 psi
Modulus of elasticity (E)	3.0e7 psi
End coefficient for spring (C_E)	1.0

A multi-objective formulation of the problem is also considered, where the second objective is to minimize the shear stress.

E.3 Speed Reducer Design

A speed reducer configuration and a typical gear are shown in Figure E.3. The design variables consist of width of gear face (b) and teeth module (m), number of pinion teeth (z), shaft lengths (l_1, l_2) and diameters (d_1, d_2) for each of the two gears. The design problem is to minimize the weight of the speed reducer subject to stress and geometry constraints [4].

The objective function is to minimize the weight of the speed reducer given by

$$f = \sum_{i=1}^{2} \left[(d_{st_{(i)}}^2 - d_{w_{(i)}}^2)\frac{\pi b}{4} + (d_{w_{(i)}}^2 - d_{p_{(i)}}^2\frac{\pi b}{12} + (d_{p_{(i)}}^2 - d_{(i)}^2)2b \right] + \pi \sum_{i=1}^{2} \frac{d_{(i)}}{4} l_{(i)}$$

where,

$$d_{p_{(i)}} = d_{(i)} + 2e_{(i)}, \quad e_{(i)} = 0.7d_{(i)},$$

$$d_{st_{(i)}} = mz_{(i)} - 2.4m, \quad d_{w_{(i)}} = d_{st_{(i)}} - 4m.$$

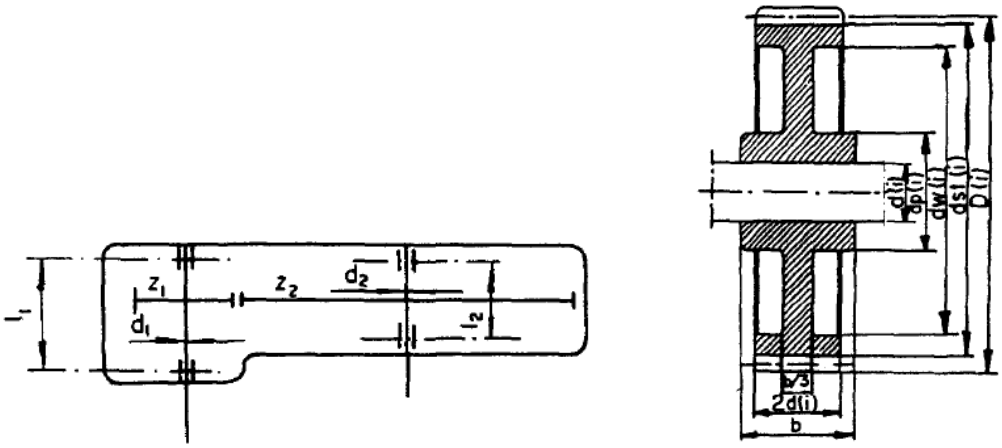


Figure E.3: Speed Reducer and typical gear (Source: [4])

The constraints for the optimization problem are:

$$\sigma_y = \frac{2Mq}{bm^2z} \leq k_g, \quad P_s^2 = \frac{2BM}{m^2z^2b} \leq P_d^2$$

$$5 \leq \frac{b}{m} \leq 12, \quad m(z_1 + z_2) \leq 160$$

$$f_1 = \frac{1}{48}\frac{Pl_1^3}{EI_1} \leq f_{01} = 0.001$$

$$f_1 = \frac{1}{48}\frac{Pl_2^3}{EI_2} \leq f_{02} = 0.001$$

$$\sigma_{g1} = \frac{M_{z1}}{W_{z1}} \leq k_{g1}, \quad \sigma_{g2} = \frac{M_{z2}}{W_{z2}} \leq k_{g2}$$

$$d_2 \geq 5, \quad 1.5d_1 + 1.9 \leq l_1, \quad 1.1d_2 + 1.9 \leq l_2$$

The bounds of the variables are – $2.6 \leq m \leq 3.6$, $0.7 \leq b \leq 0.8$, $17 \leq z \leq 28$, $7.3 \leq l_1, l_2 \leq 8.3$, $2.9 \leq d_1 \leq 3.9$, and $5.0 \leq d_2 \leq 5.5$. The values of other parameters are listed in Table E.3.

E.4 Design of a Welded Beam

The design of a welded beam is to minimize the manufacturing cost of the beam subject to constraints on deflection, shear stress, bending stress and buckling

Table E.3: Parameters for speed reducer design

Parameter	Value
Transmitted Power (N)	100 km
Pinion Speed (n)	1500 1/min
Transmission Ratio (i)	3
Permissible bending stress of gear teeth (k_g)	900 k G cm^{-2}
Permissible surface compressive stress (p_d)	5800 k G cm^{-2}
Permissible bending stress for shaft 1 (k_{g_1})	1100 k G cm^{-2}
Permissible bending stress for shaft 2 (k_{g_2})	850 k G cm^{-2}
Tooth form factor (q)	2.54

load [57]. The optimization problem has four continuous design variables – h and l are the height and the length of the weld; t and b are the thickness and the breadth of the beam as shown in Figure E.4.

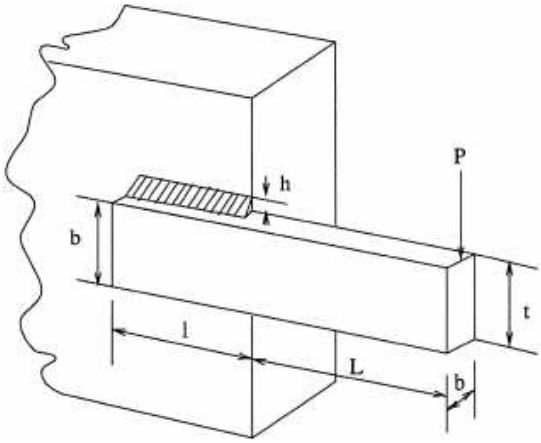


Figure E.4: Welded-Beam Problem Configuration

The mathematical formulation of the optimization problem with five constraints is as follows:

$$\begin{aligned} \text{Minimize} \quad & f = 1.10471h^2l + 0.04811tb(14 + l) \\ \text{Subject to} \quad & g_1 = 0.25 - \delta \geq 0 \\ & g_2 = 13600 - \tau \geq 0 \\ & g_3 = 30000 - \sigma \geq 0 \\ & g_4 = b - h \geq 0 \\ & g_5 = P_c - 6000 \geq 0 \end{aligned}$$

where,

$$\delta = \frac{2.1952}{t^3 b}$$
$$\sigma = \frac{504000}{t^2 b}$$
$$P_c = 64746.022(1 - 0.0282346t)tb^3$$
$$\tau = \sqrt{(\tau')^2 + (\tau'')^2 + \frac{l\tau'\tau''}{\sqrt{0.25(l^2 + (h+t)^2}}}$$
$$\tau' = \frac{6000}{\sqrt{2}hl}$$
$$\tau'' = \frac{6000(14 + 0.5l)\sqrt{0.25[l^2 + (h+t)^2]}}{2[0.707hl(l^2/12 + 0.25(h+t)^2)]}$$

The bounds for the design variables are $0.125 \leq h \leq 5.0$, $0.125 \leq t \leq 5.0$, $0.1 \leq l \leq 10.0$ and $0.1 \leq b \leq 10.0$.

A multi-objective formulation of this problem is also solved, where the additional objective is to minimize the deflection at the end of the beam. This objective is in conflict with the first objective (cost), since beam deflection is likely to be large for a beam with smaller dimensions [57].

E.5 Design of a Pressure Vessel

A cylindrical vessel is capped at both ends by hemispherical heads as shown in Figure E.5. The objective is to minimize the total cost, including the cost of material, forming and welding [156]. There are four design variables: T_s (thickness of the shell), T_h (thickness of the head), R (inner radius) and L (length of the cylindrical section of the vessel, not including the head). T_s and T_h are integer multiples of 0.0625 inch, which are the available thicknesses of rolled steel plates, and R and L are continuous.

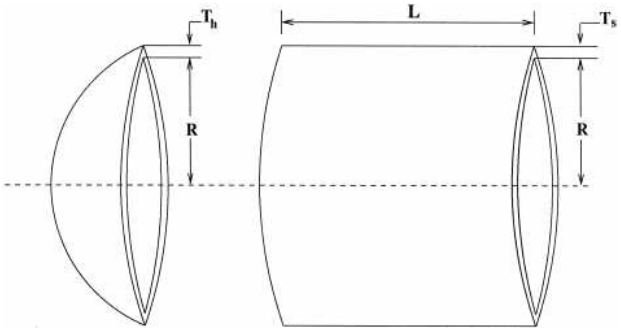


Figure E.5: Center and End section of the pressure vessel

The optimization problem is defined as follows:

$$\begin{aligned}
\text{Minimize} \quad & f = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \\
\text{subject to} \quad & g_1 = -x_1 + 0.0193x_3 \leq 0 \\
& g_2 = -x_2 + 0.00954x_3 \leq 0 \\
& g_3 = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leq 0 \\
& g_4 = x_4 - 240 \leq 0
\end{aligned}$$

The bounds on the design variables are $0.0625 \leq T_s, T_h \leq 6.25$ and $10 \leq L, R \leq 200$.

E.6 Car Side Impact Problem

Car side impact problem is a single objective problem, where the objective is to minimize the weight of the car subject to constraints on safety performance characteristics when subjected to side impact. The problem formulation is taken from [99], as shown below.

$$\begin{aligned}
\text{Minimize} \quad & f = 1.98 + 49x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 \\
& + 0.00001x_6 + 2.73x_7
\end{aligned}$$

Subject to

$$\begin{aligned}
g_1 =& 1.16 - 0.3717x_2x_4 - 0.00931x_2x_{10} - 0.484x_3x_9 + 0.01343x_6x_{10} - 1 \leq 0 \\
g_2 =& 0.261 - 0.0159x_1x_2 - 0.188x_1x_8 - 0.019x_2x_7 + 0.0008757x_5x_{10} \\
& + 0.0144x_3x_5 + 0.08045x_6x_9 + 0.00139x_8x_{11} \\
& + 0.00001575x_{10}x_{11} - 0.32 \leq 0 \\
g_3 =& 0.214 + 0.00817x_5 - 0.131x_1x_8 - 0.0704x_1x_9 + 0.0007715x_5x_{10} \\
& - 0.018x_2x_7 + 0.0208x_3x_8 + 0.121x_3x_9 - 0.00364x_5x_6 - 0.018x_2x_2 \\
& - 0.0005354x_6x_{10} + 0.00121x_8x_{11} + 0.00184x_9x_{10} \\
& + 0.03099x_2x_6 - 0.32 \leq 0 \\
g_4 =& 0.74 - 0.61x_2 - 0.163x_3x_8 + 0.001232x_3x_{10} - 0.166x_7x_9 \\
& + 0.227x_2x_2 - 0.32 \leq 0 \\
g_5 =& 28.98 + 3.818x_3 - 4.2x_1x_2 + 0.0207x_5x_{10} + 6.63x_6x_9 - 7.77x_7x_8 \\
& + 0.32x_9x_{10} - 32 \leq 0 \\
g_6 =& 33.86 + 2.95x_3 + 0.1792x_{10} - 5.057x_1x_2 - 11x_2x_8 - 0.0215x_5x_{10} \\
& - 9.98x_7x_8 + 22x_8x_9 - 32 \leq 0 \\
g_7 =& 46.36 - 9.9x_2 - 12.9x_1x_8 + 0.1107x_3x_{10} - 32 \leq 0 \\
g_8 =& 4.72 - 0.5x_4 - 0.19x_2x_3 - 0.0122x_4x_{10} + 0.009325x_6x_{10} \\
& + 0.000191x_{11}x_{11} - 4 \leq 0 \\
g_9 =& 10.58 - 0.67x_1x_2 - 1.95x_2x_8 + 0.02054x_3x_{10} - 0.0198x_4x_{10} \\
& + 0.028x_6x_{10} - 9.9 \leq 0 \\
g_{10} =& 16.45 - 0.489x_3x_7 - 0.843x_5x_6 + 0.0432x_9x_{10} - 0.0556x_9x_{11} \\
& - 0.000786x_{11}x_{11} - 15.7 \leq 0
\end{aligned}$$

E.7 Bulk Carrier Design Problem

The bulk carrier design problem was originally formulated by [165]. The same formulation with corrections for Froude number has been presented in [157]. The original problem has three objectives: (1) Minimization of transport cost, (2) Minimization of light ship mass and (3) Maximization of annual cargo transport capacity. There are six independent variables in the problem: L = length(m), B = beam(m), D = depth(m), T = draft(m), V_k = speed(knots) and C_B = block coefficient. The variable bounds are given by $0 \leq L \leq 500$, $0 \leq T50$, $0 \leq D \leq 50$, $0.63 \leq C_B \leq 0.75$, $0 \leq B \leq 100$ and $14 \leq V_k \leq 18$. The objective formulation for the problem is given below.

$$\begin{aligned}
&\text{Minimize} \quad \text{transportation cost} = \text{annual costs / annual cargo} \\
&\text{Minimize} \quad \text{light ship weight} = W_s + W_o + W_m(t) \\
&\text{Maximize} \quad \text{annual cargo} = \text{cargo}DWT.RTPA(t/yr)
\end{aligned}$$

The objectives are calculated based on the following model for a family of bulk carriers ranging from 3000 to 500000 DWT and the speed ranging from 14 to 18 knots [157]:

Bulk carrier design model

annual cost = capital costs + running costs + voyage costs

capital costs = $0.2 \times$ ship cost

ship cost = $1.3(2000W_s^{0.85} + 3500W_o + 2400P^{0.8}$

steel weight = $W_s = 0.034L^{1.7}B^{0.7}D^{0.4}C_B^{0.5}$

outfit weight = $W_o = 1.0L^{0.8}B^{0.6}D^{0.3}C_B^{0.1}$

machinery weight = $W_m = 0.17P^{0.9}$

displacement = $1.025LBTC_B$

power = $P =$ displacement$^{2/3}V_k^3/(a + bF_n)$

Froude number = $F_n = V/(gL)^0.5, V = 0.5144m/s; g = 9.8065m/s^2$

$a = 4977.06C_b^2 - 8105.61C_B + 4456.51$

$b = -10847.2C_B^2 + 12817C_B - 6960.32$

running costs = $40000DWT^0.3$

deadweight = DWT = displacement - light ship

voyage costs = fuel cost + port cost.$RTPA$

fuel cost = 1.05daily consumption.sea daysfuel price

daily consumption = $0.19P.24/1000 + 0.2$

sea days = round trip miles/$24V_k$

fuel price = 100

port cost = $6.3DWT^{0.8}$

round trips per year = $RTPA = 350/$(sea days + port days)

port days = 2[(cargo deadweight/handling rate) + 0.5]

cargo deadweight = $DWT -$ fuel carried - miscellaneousDWT

handling rate = $8000(t/day)$

vertical center of buoyancy = $KB = 0.53T$

metacentric radius = $BM_T = (0.085C_B - 0.002)B^2/(TC_B)$

vertical center of gravity = $KG = 1.0 + 0.52D$

Subject to:

$$L/B \leq 6, \quad L/D \leq 15, \quad L/T \leq 19$$
$$T \leq 0.45DWT^{0.31}$$
$$T \leq 0.7D + 0.7$$
$$3000 \leq DWT \leq 500000$$
$$0.63 \leq C_B \leq 0.75$$
$$F_n \leq 0.32$$
$$GM_t = KB + BM_T - KG \geq 0.07B$$

Studies on two different formulations of the problem are presented in this paper, as detailed below:

Single objective problem: For single objective formulation, only the minimization of transport cost is considered. Along with the original constraints, an additional constraint on the minimum cargo transported (10^6 tonnes/year) is imposed.

Multi-objective design problem: For multi-objective formulation, two objectives are considered: minimization of transport cost and maximization of annual cargo transport capacity.

E.8 Airfoil Design

The airfoil shape is represented using a PARSEC formulation [158] as shown in Figure E.6. The PARSEC method uses eleven parameters to represent the airfoil. They are leading edge radius (r_{le}), upper crest location (X_{up}, Z_{up}), upper crest curvature (Z_{XXup}), lower crest location (X_{lo}, Z_{lo}), lower crest curvature (Z_{XXlo}), trailing edge wedge direction (α_{TE}), trailing edge wedge angle (β_{TE}), trailing edge offset (Z_{TE}) and trailing edge thickness (ΔZ_{TE}).

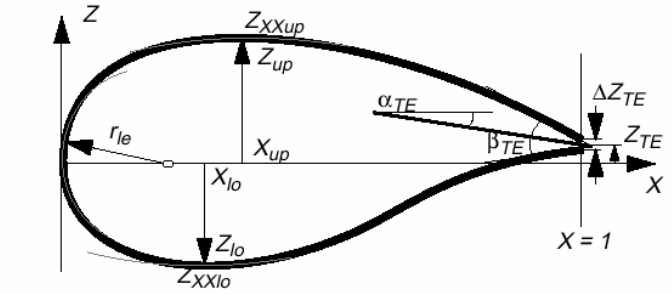


Figure E.6: PARSEC representation for 2-D airfoil

The mathematical formulation of PARSEC is given by,

$$Z_u = \sum_{i=1}^{6} a_i X^{n-1/2}$$

$$Z_l = \sum_{i=1}^{6} b_i X_{n-1/2}$$

where X is the chord wise location, Z_u is the coordinate of the upper surface and Z_l is coordinate of the lower surface of the airfoil. The chord length is assumed to be one. The coefficients a_i and b_i are solved using the eleven parameters.

In this study, a multigrid Euler code is used to compute the flow around the airfoil. The governing Euler equations are solved using a finite volume formulation as proposed in [166]. The field grid is generated algebraically using a conformal mapping method to create an O-grid airfoil of size 161×33 around the airfoil. In the interest of robustness in the multigrid computation, full coarsening up till a minimum of four cells was used with lower values of Courant-Friedrichs-Lewy number set at seven.

Aerodynamic design of the airfoil is carried out to minimize the drag coefficient (C_d) subject to the lift coefficient (C_l) value of 0.824 ($C_l \geq 0.824$). Flow Mach number is set to 0.73 and the angle of attack is set at two degrees. The trailing edge offset (Z_{TE}) and the trailing edge offset (ΔZ_{TE}) are set to zero. The bounds for the remaining variables are listed in Table E.4.

Table E.4: Design variable limits for airfoil design problem

Variable	Lower Bound	Upper Bound
R_{le}	0.0055	0.0085
X_{up}	0.3	0.5
Z_{up}	0.0055	0.0085
Z_{XXup}	-0.6	-0.4
X_{lo}	0.28	0.42
Z_{lo}	-0.075	-0.05
Z_{XXlo}	0.55	0.85
α_{TE}	-12	-8
β_{TE}	-14.5	-9.5

E.9 Water Resource Problem

The water resource problem, proposed by Musselman and Talavage [112] consists of 5 objectives and 7 constraints; and its formulation is given in Equation E.1.

$$
\begin{aligned}
&\text{Minimize}\\
&f_1(\mathbf{x}) = 106780.37(x_2 + x_3) + 61704.67,\\
&f_2(\mathbf{x}) = 3000x_1,\\
&f_3(\mathbf{x}) = (305700/(0.06 \times 2289)^{0.65}) \times 2289x_2,\\
&f_4(\mathbf{x}) = 250 \times 2289 \times exp(-39.75x_2 + 9.9x_3 + 2.74),\\
&f_5(\mathbf{x}) = 25 \times (1.39/(x_1x_2) + 4940x_3 - 80),\\
&\text{Subject to}\\
&g_1(\mathbf{x}) = 0.00139/x_1x_2 + 4.94x_3 - 0.08 \leq 1\\
&g_2(\mathbf{x}) = 0.0000306/x_1x_2 + 0.1082x_3 - 0.00986 \leq 0.10\\
&g_3(\mathbf{x}) = 12.307/x_1x_2 + 49408.24x_3 - 4051.02 \leq 50000\\
&g_4(\mathbf{x}) = 2.098/x_1x_2 + 8.046.33x_3 - 696.71 \leq 16000,\\
&g_5(\mathbf{x}) = 2.138/x_1x_2 + 7883.39x_3 - 705.04 \leq 10000,\\
&g_6(\mathbf{x}) = 0.417/x_1x_2 + 1721.36x_3 - 136.54 \leq 2000,\\
&g_7(\mathbf{x}) = 0.164/x_1x_2 + 631.13x_3 - 54.48 \leq 550\\
&\text{where}\\
&0.01 \leq x_1 \leq 0.45, 0.01 \leq x_2, x_3 \leq 0.10
\end{aligned}
\tag{E.1}
$$