Development of optimization methods to solve computationally expensive problems

Amitay Isaacs

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

31 August 2009

Copyright Statement

Signed .....

Date .....24/11/2009.....

Authenticity Statement

Signed .....

Date .....24/11/2009.....

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 .....24/11/2009.....

Abstract

Evolutionary algorithms (EAs) are population based heuristic optimization methods used to solve single and multi-objective optimization problems. They can simultaneously search multiple regions to find global optimum solutions. As EAs do not require gradient information for the search, they can be applied to optimization problems involving functions of real, integer, or discrete variables. One of the drawbacks of EAs is that they require evaluations of numerous candidate solutions for convergence.

Most real life engineering design optimization problems involve highly nonlinear objective and constraint functions arising out of computationally expensive simulations. For such problems, the computation cost of optimization using EAs can become quite prohibitive. This has stimulated the research into improving the efficiency of EAs reported herein.

In this thesis, two major improvements are suggested for EAs. The first improvement is the use of spatial surrogate models to replace the expensive simulations for the evaluation of candidate solutions, and other is a novel constraint handling technique. These modifications to EAs are tested on a number of numerical benchmarks and engineering examples using a fixed number of evaluations and the results are compared with basic EA. addition, the spatial surrogates are used in the truss design application.

A generic framework for using spatial surrogate modeling, is proposed. Multiple types of surrogate models are used for better approximation performance and a prediction accuracy based validation is used to ensure that the approximations do not misguide the evolutionary search. Two EAs are proposed using spatial surrogate models for evaluation and evolution. For numerical benchmarks, the spatial surrogate assisted EAs obtain significantly better (even orders of magnitude better) results than EA and on an average 5–20% improvements in the

objective value are observed for engineering examples.

Most EAs use constraint handling schemes that prefer feasible solutions over infeasible solutions. In the proposed infeasibility driven evolutionary algorithm (IDEA), a few infeasible solutions are maintained in the population to augment the evolutionary search through the infeasible regions along with the feasible regions to accelerate convergence. The studies on single and multi-objective test problems demonstrate the faster convergence of IDEA over EA. In addition, the infeasible solutions in the population can be used for trade-off studies.

Finally, discrete structures optimization (DSO) algorithm is proposed for sizing and topology optimization of trusses. In DSO, topology optimization and sizing optimization are separated to speed up the search for the optimum design. The optimum topology is identified using strain energy based material removal procedure. The topology optimization process correctly identifies the optimum topology for 2-D and 3-D trusses using less than 200 function evaluations. The sizing optimization is performed later to find the optimum cross-sectional areas of structural elements. In surrogate assisted DSO (SDSO), spatial surrogates are used to accelerate the sizing optimization. The truss designs obtained using SDSO are very close (within 7% of the weight) to the best reported in the literature using only a fraction of the function evaluations (less than 7%).

Acknowledgements

My research work would not have been possible without the help and the support of the family and friends.

I thank Tapabrata Ray for his guidance throughout the course of my research. He has been an excellent supervisor and mentor. Ray's infectious enthusiasm and constant encouragement have kept me motivated. Regular brainstorming sessions with him have provided me with the necessary direction for my research. I am indebted to him for his graciousness and generosity. I also thank his family, Jayati and Pritika, for making me feel at home in a foreign country. I will cherish his friendship forever.

I thank Warren Smith for his valuable suggestions and insights. His meticulous proof-reading has been a tremendous help in improving my writing skills.

Many thanks to all my friends and colleagues at ADFA who made my stay in Canberra and work at ADFA a memorable experience. Special thanks to my officemates Mahendra, Khin, Sudantha, Abhi, Hemant, Ahmad and Vishal who shared countless cups of tea with me and a regular dose of phdcomics. Thanks to Roshan, Vishwas, Laxmikant and Arif for sharing their research and giving me a chance to work with them. A special mention of Hemant for endless discussions that resulted in many joint publications.

Thanks to Russell Boyce and Hideki Ogawa from the University of Queensland for an opportunity to apply my research to a real-life engineering problem.

I dedicate this thesis to my lovely wife, Deanna. I would not have been able to complete my research work without her encouragement. Deanna has accompanied me on this journey every step of the way and helped me emotionally and spiritually during difficult times. I thank our parents for their support and blessings. Thank you Melroy for helping us set up a home away from home in Canberra.

List of Publications

Book Chapters

[1] T. Ray, H. K. Singh, A. Isaacs, and W. Smith. Infeasibility drive evolutionary algorithm for constrained optimization. In Efren Mezura-Montes, editor, *Constraint Handling in Evolutionary Optimization*, Studies in Computational Intelligence, pages 147–167. Springer, 2009.

[2] A. Isaacs, T. Ray, and W. Smith. Set representation and multi-parent learning within an evolutionary algorithm for optimal design of trusses. In Ying ping Chen and Meng-Hiot Lim, editors, *Linkage in Evolutionary Computation*, volume 157 of *Studies in Computational Intelligence*, pages 419–439. Springer, 2008.

[3] T. Ray, A. Isaacs, and W. Smith. A memetic algorithm for dynamic multi-objective optimization. In C. K. Goh, C. K. Tan, and Y. S. Ong, editors, *Multi-objective Memetic Algorithms*, Studies in Computational Intelligence. Springer, 2008.

[4] T. Ray, A. Isaacs, and W. Smith. Multi-objective optimization using surrogate assisted evolutionary algorithm. In G. P. Rangaiah, editor, *Multi-objective Optimization: Techniques and Applications in Chemical Engineering*, pages 131–151. World Scientific, Singapore, 2008.

Journal Papers

[1] A. Isaacs, T. Ray, and W. Smith. Multiobjective design optimization using multiple adaptive spatially distributed surrogates. *International Journal of Product Development*, 9(1-3):188–217, 2009.

Conference Papers

[1] H. Ogawa, Y. Alazet, A. Pudsey, R. R. Boyce, A. Isaacs, and T. Ray. Full-flow path optimization of axisymmetric scramjet engines. In *Proceedings*

of the 48th AIAA Aerospace Sciences Meeting, Florida, January 2010. Accepted.

[2] H. Ogawa, R. R. Boyce, A. Isaacs, and T. Ray. Multi-objective design optimisation of inlet and combustor for axisymmetric scramjets. In *Proceedings of the Australian Combustion Symposium*, December 2009. Accepted.

[3] H. Ogawa, Y. Alazet, R. R. Boyce, A. Isaacs, and T. Ray. Design optimisation of axisymmetric scramjets for access-to-space. In *Proceedings of the 9th Australian Space Science Conference*, Sydney, September 2009.

[4] M. A. Ashraf, A. Isaacs, J. Young, J. C. S. Lai, and T. Ray. Numerical simulation and multi-objective design of flow over oscillating airfoil for power extraction. In *Proceedings of Conference on modelling fluid flow (CMFF)*, pages 221–228, Budapest, Hungary, September 2009.

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

[6] H. K. Singh, A. Isaacs, T. T. Nguyen, T. Ray, and Xin Yao. Performance of infeasibility driven evolutionary algorithm (IDEA) on constrained dynamic single objective optimization problems. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 3127–3134, Norway, May 2009.

[7] A. Isaacs, T. Ray, and W. Smith. Memetic algorithm for dynamic bi-objective optimization problems. In *Proceedings of the IEEE Congress on Evolutionary Computation 2009 (CEC'09)*, pages 1707–1713, Norway, May 2009.

[8] 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 *Simulated Evolution and Learning, 7th International Conference SEAL 2008, Proceedings*, volume 5361 of *Lectures Notes in Computer Science*, pages 401–410, Melbourne, Australia, 2008. Springer.

[9] A. Isaacs, T. Ray, and W. Smith. An efficient hybrid algorithm for optimization of discrete structures. In *Simulated Evolution and Learning, 7th International Conference SEAL 2008, Proceedings*, volume 5361 of *Lectures Notes in Computer Science*, pages 625–634, Melbourne, Australia, 2008. Springer.

[10] H. K. Singh, A. Isaacs, T. Ray, and W. Smith. Infeasibility driven evolutionary algorithm (IDEA) for engineering design optimization. In *AI2008: Advances in Artificial Intelligence, 21st Australasian Joint*

Conference on Artificial Intelligence, Proceedings, volume 5360 of *Lecture Notes in Artificial Intelligence*, pages 104–115, Auckland, New Zealand, 2008. Springer.

[11] H. K. Singh, A. Isaacs, T. Ray, and W. Smith. A simulated annealing algorithm for single objective trans-dimensional optimization problems. In *Proceedings of Eighth International Conference on Hybrid Intelligent Systems (HIS-08)*, pages 19–24, Barcelona, Spain, 2008.

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

[13] A. Isaacs, T. Ray, and W. Smith. Blessings of maintaining infeasible solutions for constrained multi-objective optimization problems. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 2780–2787, Hong Kong, 2008.

[14] A. Isaacs, V. Puttige, T. Ray, W. Smith, and A. Sreenatha. A development of memetic algorithm for dynamic multi-objective optimization and its application to online system identification. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 548–554, Hong Kong, 2008.

[15] A. Isaacs, T. Ray, and W. Smith. An evolutionary algorithm with spatially distributed surrogates for multi-objective optimization. In *Progress in Artificial Life, Third Australasian Conference on Artificial Life (ACAL)*, volume 4828 of *Lecture Notes in Computer Science*, pages 257–268, Gold Coast, Australia, 2007. Springer.

[16] A. Isaacs, T. Ray, and W. Smith. A hybrid evolutionary algorithm with simplex local search. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 1701–1708, Singapore, 2007.

[17] A. Isaacs, T. Ray, and W. Smith. Novel evolutionary algorithm with set representation scheme for truss design. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, pages 3902–3908, Singapore, 2007.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ACO	Ant Colony Optimization
ANN	Artificial Neural Network
ASCHEA	Adaptive Segregational Constraint Handling Evolutionary Algorithm
BLUP	Best Linear Unbiased Prediction
CEM	Computational Electro-Magnetics
CFD	Computational Fluid Dynamics
COMOGA	Constrained Optimization by Multi-Objective Genetic Algorithms
CONGA	COnstraint based Numeric Genetic Algorithm
CRR	Constraint Relative Rank
DACE	Design and Analysis of Computer Experiments
DE	Differential Evolution
DOE	Design Of Experiments
DSO	Discrete Structures Optimization
EA	Evolutionary Algorithm
EGO	Efficient Global Optimization
EP	Evolutionary Programming
ES	Evolution Strategy
ESO	Evolutionary Structural Optimization
FEA	Finite Element Analysis
FEM	Finite Element Method
GA	Genetic Algorithm
GD	Generational Distance
IDEA	Infeasibility Driven Evolutionary Algorithm

LOOCV	Leave-One-Out Cross Validation
MA	Memetic Algorithm
MAS	Multiple Adaptive Surrogates
MDO	Multidisciplinary Design Optimization
MLE	Maximum Likelihood Estimate
MLP	Multi-Layer Perceptron
MS	Multiple Surrogates
MSE	Mean Squared Error
NN	Neural Network
NPGA	Niched Pareto Genetic Algorithm
NSGA-II	Non-dominated Sorting Genetic Algorithm II
POF	Pareto Optimal Front
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
RMSE	Root Mean Squared Error
RSM	Response Surface Method
SA	Simulated Annealing
SAE	Surrogate Assisted Evaluation
SAR	Surrogate Assisted Recombination
SBX	Simulated Binary Crossover
SDSO	Surrogate-assisted Discrete Structures Optimization
SMES	Simple Multi-membered Evolution Strategy
SS	Single Surrogate
TS	Tabu Search
VGA	Variable-string-length Genetic Algorithm

Chapter 1

Introduction

1.1 Background

Engineering involves the design and manufacture of complex products through the structured and rational application of science and mathematics. An engineered product aims to provide a benefit to mankind and where development is undertaken, there is generally an endeavour to improve the product quality through design improvement. A good design has to deliver the desired functionality (or performance) at competitive cost and resource consumption. The best design will win these comparisons. There is an increasing trend to use simulations to predict a product's performance even before it is manufactured and verify that the design meets all the performance requirements. The simulations (also referred to as numerical experiments) involve solving the mathematical model describing the governing physics. The mathematical model can be represented as a set of algebraic, ordinary or partial differential equations. These equations are solved numerically using computers and the performance of a design is evaluated. For various engineering disciplines sophisticated computer codes have been developed.

Examples of such codes include finite element methods (FEM), computational fluid dynamics (CFD) and computational electro-magnetics (CEM). These simulations can take just a few seconds to run for simple problems and hours or even days to run for complex problems. Often the performance of engineering products is governed by more than one engineering discipline and it is important to balance the interactions between multiple disciplines. In such situations, evaluating a single design involves iterative runs of different simulations, increasing the computational cost even further.

In the design process several designs are typically evaluated before selecting the better design for manufacture. Optimization methods provide a systematic way of searching for better designs based on the performance criteria. They can be applied to any design process as long as there is a way of comparing the performance of two designs quantitatively. Evolutionary optimization methods are inspired by the natural evolution process and can search throughout the design space to find one or more designs that satisfy the performance requirements. Some of the designs can be quite novel and lead to completely new concepts for the product. Shown in Figure 1.1 is the X-band antenna designed for NASA's ST5 mission using evolutionary algorithms. The designed antenna resulted in 93% efficiency as compared to 38% efficiency of the traditional helix antenna. The designed antenna required lower power and achieved high gain across a wider range of elevation angles [3].

In the optimization process the value of the performance criterion (referred to as the objective function) is optimized by varying the values of the design variables. Some of the examples of performance criteria used in engineering design are to minimize weight, to minimize cost, to maximize range, etc. In addition, there are generally constraints on the design that must be met, e.g.

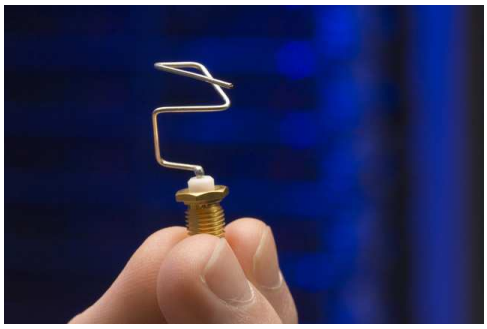


Figure 1.1: Computer evolved X-band Antenna for NASA's ST5 mission (source: http://amesnews.arc.nasa.gov/releases/2004/antenna/ antenna.html)

limits on stress for the structural elements. The objective and the constraint functions are often non-linear and are evaluated using numerical simulations.

1.2 Motivation

Several optimization methods have been developed over the years to solve optimization problems efficiently. The most commonly used methods are the numerical methods that use gradient information to find a search direction and then search along that direction to improve the objective value. Gradient based methods can only be applied to problems where the objective and the constraint functions are smooth (*i.e.* the functions are continuous and the derivatives exist). The gradients are usually calculated using the finite difference formulation and several designs (as many as the number of design variables) need to be evaluated to establish the gradient information at a single point. In cases with a large number of design variables, the gradient calculation can become quite expensive. In addition, the gradient based methods often require a feasible starting point and they are limited to finding a local optimum.

Evolutionary algorithms (EAs) fall in into another class of optimization meth-

ods called heuristic methods. These methods do not require gradient information and can be applied easily to optimization problems with non-smooth functions and disjoint feasible spaces. Heuristic methods are global search methods and can find the globally optimum solutions to optimization problems. Evolutionary algorithms are population based methods that can search different regions of the design space simultaneously. In addition, they can solve single as well as multi-objective optimization problems. The early applications of evolutionary algorithms in engineering design were in the area of structural design [4, 5, 6]. Since then they have been applied in various disciplines, for example, in the design of composite laminates [7, 8] and composite panels [9, 10], aerodynamic design of automobiles [11], compressor blade design [12] and aircraft wing design [13, 14]. Various applications of evolutionary algorithms in the area of structural optimization can be found in the survey by Kicinger, Arciszewski and De Jong [15]. Evolutionary algorithms are also being used in other engineering disciplines including architecture, civil engineering, electrical and control engineering and computer science [16, 17].

In evolutionary algorithms, a population of candidate solutions is evolved over a number of generations to find the optimum solutions. Evolutionary algorithms are known to require evaluations of large number of solutions. For the design optimization problems requiring expensive simulations to evaluate the objective and the constraint functions, the total cost of the optimization can become quite prohibitive. Therefore, an important motivation exists to improve the efficiency and the effectiveness of evolutionary algorithms to reduce the computational cost of the optimization process.

One of the ways to reduce the computational cost of a simulation is the use of approximations. Implied is that the objective and the constraint func-

tions are approximated using simpler functions. These replacement functions are referred to as surrogate models or metamodels. These surrogate models are computationally inexpensive to evaluate when compared to the simulations of the mathematical models and can be used in place of the expensive simulations. Even though surrogate models are being used in evolutionary algorithms to reduce the computation cost, their use in evolutionary algorithms is not straightforward and requires consideration of many issues – type of the surrogate model, selection of the training data, global versus local models, and accuracy of the prediction – to name a few.

Evolutionary algorithms do not have a native mechanism for constraint handling. Different evolutionary algorithms use various schemes to weed out the infeasible solutions and migrate the search to the feasible design space. The performance of EA on constrained optimization problems is greatly dependent on the constraint handling technique used. The common approach is to convert the objectives and the constraints into a single fitness measure using a penalty formulation, but this approach requires the specification of additional parameters (weights) that are very much problem dependent.

1.3 Scope of Research

In this thesis, evolutionary algorithms are used as the underlying optimization method. Although, an EA is well suited for the optimization in engineering design, the use of computationally expensive simulations can offset the advantages of EA as large numbers of evaluations cannot be afforded. The compromise is to fix the number of simulations that can be performed and come up with the best design possible using the fixed number of evaluations. In the engineering

disciplines often the "true" optimum solutions are not sought as it requires significant investment in computational time. However, improvements in the design at reduced computational effort are readily accepted. It follows that the aim of this thesis is to answer the following questions:

- For a fixed number of function evaluations, how can the quality of the best solutions obtained using evolutionary algorithms be improved?
  - How can surrogate models be used to improve convergence?
  - Can better constraint handling methods improve the convergence for constrained optimization problems?

The objective of the thesis is realized through modifications and extensions to EAs to improve their convergence. The extensions should be generic enough to be implemented in any of the EAs, and should not be specific to a particular EA. It is not the aim of the work reported in the thesis to build a black-box implementation of an EA that can solve all the design problems. Rather it is to highlight, through research, the areas in EA where improvements can be made. The benefits of the proposed techniques are demonstrated using numerical benchmarks.

The research is divided in three main topics. The first topic is to augment the EA framework with surrogate models to speed up convergence. As there is no one type of surrogate model that is best suited for every problem, a generic approach is developed using multiple types of surrogate models. The second research topic is constraint handling. The main motivation is to question the assumption – *feasible solutions are better than infeasible solutions*. The final topic is the application of surrogate models in the design of discrete structures like trusses.

1.4 Contributions of Thesis

Five significant contributions made in this thesis are as follows.

1. *Spatial surrogate modeling*, a generic framework for use of surrogate models within evolutionary algorithms is formulated. Spatial surrogate modeling uses multiple types of surrogate models for better approximation of functions unlike the common practice of using a single type of surrogate model. Multiple spatially distributed surrogate models can be built using spatial surrogate models instead of a single global surrogate model for functions of a large number of variables.

2. Two evolutionary algorithms are developed using spatial surrogate models, namely surrogate assisted evaluation (SAE-EA) and surrogate assisted recombination (SAR-EA). Both of these algorithms are tested on a set of benchmark problems and engineering examples. Significantly better objective values are obtained for numerical benchmarks using SAE-EA and SAR-EA as compared to EA for a fixed computational cost. For engineering examples, 5–20% improvement in the objective values are obtained using SAE-EA and SAR-EA.

3. A novel constraint handling method is proposed and implemented in the infeasibility driven evolutionary algorithm (IDEA). Contrary to the common approach of almost blindly preferring feasible solutions over infeasible solutions, a few infeasible solutions are retained and are ranked higher than feasible solutions in IDEA. The performance of IDEA is substantially better than EA for constrained optimization problems owing to simultaneous search through the infeasible and the feasible regions focused near the con-

straint boundaries. The favourable performance of IDEA is demonstrated on a number of single and multi-objective constrained test problems.

4. An evolutionary algorithm, discrete structures optimization (DSO), is proposed and developed for the design of discrete structures. In DSO, topology optimization and sizing optimization are performed in separate steps. Using strain energy based material removal criterion, the optimum topology is identified using very few evaluations. The sizing optimization is accelerated using spatial surrogates in surrogate assisted discrete structures optimization (SDSO). For the truss design problems studied, designs close to the best designs (within 7% of the weight) as reported in the literature are obtained with SDSO using significantly fewer function evaluations (less than 7%).

5. A set of baseline results of surrogate assisted EAs using 1,000 function evaluations are presented across a range of benchmark problems for the thesis. These systematically presented results would be useful to other researchers attempting to further the development of surrogate models assisted optimization algorithms. Similar results are presented for constraint handling method using 4,000 function evaluations.

1.5 Organization of Thesis

The thesis is organized in eight chapters. In this chapter the aim and the motivation of the research is established. Optimization has become an integral part of the design process and performance evaluation using computationally expensive simulations has become a norm. The performance of evolutionary algorithms needs to be improved to make the cost of optimization affordable in

the context of real engineering studies.

An overview of optimization and optimization methods is presented in *Chapter Two*. It is provided for completeness and can be skipped by readers familiar with optimization and evolutionary algorithms. Covered in the chapter are the mathematical definition of an optimization problem, solutions of single- and multi-objective optimization problems, and various optimization methods with focus on evolutionary algorithms. The traditional numerical methods for optimization are compared with the heuristic methods for optimization. A generic framework for evolutionary algorithms is presented and explained with reference to the most popular Non-dominated Sorting Genetic Algorithm II (NSGA-II) [18], which is used as a canonical EA in this thesis. Finally, the performance metric used in this study to compare the results of different algorithms is described.

A literature review on the approximation methods used in evolutionary algorithms is presented in *Chapter Three*. Various techniques used by researchers to embed different surrogate models in EAs are compared and the limitations of the techniques are highlighted. The focus is on surrogate models in evolutionary algorithms, even though they have been applied to other heuristic methods as well. A discussion of memetic algorithms, which are heuristic methods with embedded local search, is presented in the context of evolutionary algorithms, that use surrogate models for local search.

A spatial surrogate modeling framework is proposed in *Chapter Four*. Spatial surrogate models are built using multiple types of surrogate models and they use explicit validation checks to ensure the prediction accuracy. Spatial surrogate models are used in lieu of expensive simulations to evaluate the objectives and the constraints in EAs. Two techniques of using spatial surrogate models in EAs are proposed – surrogate assisted evaluation and surrogate assisted recombination.

The benefits of both the algorithms are highlighted using a number of benchmark test problems.

A novel idea of maintaining infeasible solutions in the population to improve the convergence is presented in *Chapter Five*. First, The existing methods of constraint handling in EA are discussed. Then, the mechanism of retaining infeasible solutions in the population and the proposed constraint handling method in IDEA are presented. In IDEA, the original optimization problem is reformulated with an additional objective, which is the constraint violation measure. Two constraint violation measures are proposed and the benefits are discussed. The performance of IDEA is compared with EA on single and multi-objective test problems to highlight the improvements in convergence achievable.

The advantages of the proposed surrogate assisted EAs (SAE-EA and SAR-EA) and IDEA are highlighted using commonly used engineering design examples in *Chapter Six*. The engineering design examples include spring design, pressure vessel design, welded beam design and airfoil design. For all the engineering examples a fixed number of evaluations is used in the design optimization.

In *Chapter Seven*, the design optimization of discrete structures is studied. Various approaches using evolutionary method of structural design optimization are compared. A new optimization method, discrete structures optimization (DSO), is proposed for the design of truss structures. In DSO, topology optimization is handled separately from sizing optimization. The topology optimization uses strain energy based material removal for identification of optimum topology. The sizing optimization using spatial surrogates is proposed in surrogate assisted DSO (SDSO). The advantages of separating optimum topology identification and sizing optimization are highlighted with 2-D and 3-D truss design examples.

A summary of the research outcomes and contributions are highlighted in the

final chapter. Also, the areas of future research based on the proposed methods are identified. The numerical benchmark optimization problems used in this study are fully described in *Appendix A–D*.

Chapter 2

Optimization and EA

2.1 Overview

Any optimization method requires a formal statement of the optimization problem to be solved. A generic optimization problem statement is presented in Section 2.2. A single objective optimization seeks a single solution – a local or the global optimum. In case of multi-objective optimization there are multiple equally good solutions called the Pareto optimal solutions. These solutions are identified by the non-dominance relationship. The traditional numerical optimization methods are limited to solving single objective optimization problems involving functions having certain properties. The limitations of numerical optimization methods are highlighted in Section 2.3. Heuristic algorithms do not have the same limitations and offer many advantages. A variety of heuristic methods developed over the years are mentioned in Section 2.4. One class of heuristic methods is evolutionary algorithms (EAs) which are used in this study. A generic framework for an EA and the various elements of EA are described in Section 2.5. The performance of two algorithms can be easily

compared for single objective optimization based on the final solution, but it is non-trivial for multi-objective optimization. A performance metric, displacement, used to compare two non-dominated sets of solutions obtained using different algorithms is described in Section 2.6. The summary of the chapter is presented in Section 2.7.

2.2 Optimization Problem

An optimization problem is defined by the design variables, the objectives, and the constraints. The first step in the optimization process is parametrizing the design into a set of design variables. Different designs can then be obtained by assigning different values to the design variables. In the optimization process the values of the design variables are varied to search for the design that has the best performance (the objective) and satisfies all the imposed requirements (the constraints).

A multi-objective or multi-criteria constrained optimization problem (posed as a minimization problem) can be written mathematically as shown in Equation 2.1.

$$\begin{aligned} \text{Find} \quad & \mathbf{x} = (x_1, \ldots, x_n) \\ \text{To minimize} \quad & f_1(\mathbf{x}), \ldots, f_k(\mathbf{x}) \\ \text{Subject to} \quad & g_i(\mathbf{x}) \geq 0, \quad i = 1, \ldots, m \\ & h_j(\mathbf{x}) = 0, \quad j = 1, \ldots, p \end{aligned} \tag{2.1}$$

where $x_1, \ldots, x_n$ are the design variables that can be integer or real, continuous or discrete. The design variables are usually bounded by lower and upper limits (also referred as bounds) to form the design space $\mathcal{S}$. The objective functions $f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})$ are simultaneously minimized while satisfying the inequality con-

straints $g_i(\mathbf{x}) \geq 0$ and the equality constraints $h_j(\mathbf{x}) = 0$. 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(s), where all the constraints are satisfied; and infeasible region(s), where one or more constraints are violated.

2.2.1 Single Objective Optimization

A single objective constrained optimization problem has a single objective function $f(\mathbf{x})$ that is minimized while satisfying the constraints. The solution of a single objective optimization problem is the global minimum. The global minimum solution ($\mathbf{x}^*$) has the least objective value as compared to any other solution in the design domain and is mathematically defined as in Equation 2.2.

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \quad \forall \mathbf{x} \in \mathcal{S} \tag{2.2}$$

Often another solution is accepted and that is the local minimum solution ($\mathbf{x}^o$) as defined in Equation 2.3.

$$f(\mathbf{x}^o) \leq f(\mathbf{x}) \quad \|\mathbf{x} - \mathbf{x}^o\| < \epsilon \tag{2.3}$$

A local minimum is the best solution in the neighborhood of $\mathbf{x}^o$. The difference between the global minimum and a local minimum is illustrated in Figure 2.1. The solid line represents the objective function $f(x)$ to be minimized subject to the constraint $g(x) \geq 0$ shown as the dashed line. The objective and the constraint are the functions of a single variable x with the limits $0 \leq x \leq 5$. For all the values of $x < 2.06$ the constraint value is less than 0, indicating the infeasible

region. For all values of $x > 2.06$ the constraint value is greater than 0, indicating the feasible region. The solution marked with a square (corresponding to $x = 2.8$) is the local minimum, as there is no better solution in the small neighborhood around the solution. The solution marked with a circle (corresponding to $x = 4.2$) is the global minimum, as it is the smallest possible value for the function in the feasible region.

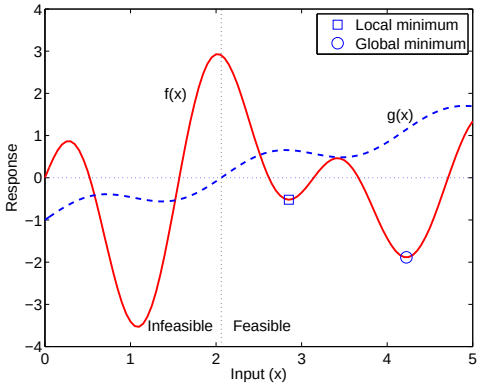


Figure 2.1: Illustration of local/global minimum and infeasible/feasible regions for single objective optimization problem

2.2.2 Multi-objective Optimization

In optimization problems with multiple objectives, the objectives are usually in conflict and reducing the value of one objective results in the increase in the value of one or more of the other objectives. In such cases, there is no single solution that minimizes all the objectives. If there is no conflict among the objectives, there exists a single solution that minimizes all the objectives simultaneously and in such cases the multi-objective optimization problem can be reduced to a single objective optimization problem by considering any one of the objectives. Even with conflict among the objectives, multi-objective optimization problem can be converted to a single objective optimization problem by assigning preferences (or

weights) to the objectives and considering a composite objective function. In absence of any preferences for the objectives, the purpose of multi-objective optimization is to find a set of Pareto optimal solutions (also called non-dominated solutions or trade-off solutions). All of the Pareto optimal solutions are equally good solutions to a multi-objective optimization problem. The Pareto optimal solutions are defined based on the dominance relationship between two solutions.

A solution $\mathbf{x}_1$ is dominated by another solution $\mathbf{x}_2$ if the values of all the objectives corresponding to $\mathbf{x}_2$ are better than or equal to the values of the objectives corresponding to $\mathbf{x}_1$ and at least one of the objective values corresponding to $\mathbf{x}_2$ is strictly better than that of $\mathbf{x}_1$. The two conditions can be written in mathematical form as follows:

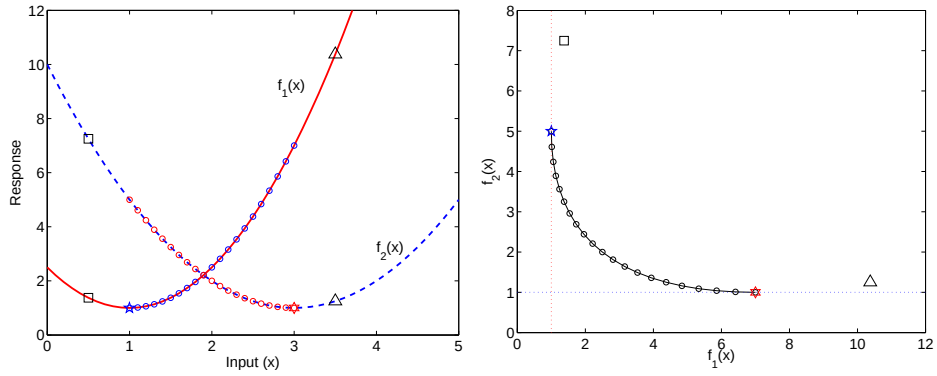
1. The solution $\mathbf{x}_2$ is no worse than $\mathbf{x}_1$ in all objectives,
   $f_i(\mathbf{x}_2) \leq f_i(\mathbf{x}_1), \quad \forall\, i = 1, \ldots, k.$

2. The solution $\mathbf{x}_2$ is strictly better than $\mathbf{x}_1$ in at least one objective,
   $f_i(\mathbf{x}_2) < f_i(\mathbf{x}_1), \quad$ for at least one $i \in 1, 2, \ldots, k.$

If solution $\mathbf{x}_1$ does not dominate solution $\mathbf{x}_2$ and vice versa, then the two solutions are non-dominated with respect to each other. All the Pareto optimal solutions are non-dominated solutions with respect to each other.

The dominance relationship and a set of Pareto optimal solutions for a bi-objective optimization problem are shown in Figure 2.2. The functions $f_1(x)$ and $f_2(x)$ to be minimized are represented by solid line and dashed line respectively in Figure 2.2(a). The minimum value for $f_1(x)$ is 1 at $x = 1$ and for $f_2(x)$ is 1 at $x = 3$. Few of the trade-off solutions (in the range $1 \leq x \leq 3$) are marked with circles. In Figure 2.2(b), the same trade-off solutions are plotted in the f-space or the objective space. For the trade-off points, any decrease in one objective

results in an increase in the other objective value and they are non-dominated with respect to each other. The two solutions marked with a square and a triangle are dominated solutions. The solution marked with a square is dominated by the corner solution (1,5) marked by a pentagon and the solution marked with a triangle is dominated by the other corner solution (7,1) marked with a hexagon in Figure 2.2(b).



(a) Design space (b) Objective space

Figure 2.2: Illustration of the Pareto optimal solutions for bi-objective optimization problem

For bi-objective optimization problems, the minimum values of the two objectives correspond to the corners of the Pareto optimal front. The minimum values of objectives $f_1(x)$ and $f_2(x)$ are marked with a pentagon and a hexagon respectively in Figure 2.2. A Pareto optimal front for a general multi-objective optimization problem is the hyper-surface containing all the Pareto optimal solutions. For bi-objective optimization, the Pareto optimal front is defined by a curve, for three objective optimization problem it is defined by a surface and so on.

2.3 Numerical methods

One of the first optimization methods developed was to find the zeros of a real-valued function by Isaac Newton in 17th century AD. The history of formal optimization methods (using gradient information) can be traced back to the development of calculus. The mathematics for the calculus of variations by Newton and Leibniz paved the way for the development of the gradient-based optimization methods.

The numerical methods using gradient information are applicable to continuous and differentiable real-valued functions. The methods for non-linear optimization include first and second order methods. The first order methods such as trust region method, step length method and quasi-newton method require first order gradient information (*i.e.* derivative, gradient or Jacobian). The second order methods such as newton method and sequential quadratic programming require second order information (*i.e.* second derivative or Hessian) in addition to the first order information. Detailed description of these optimization methods can be found in the textbooks on numerical optimization [19, 20, 21]. There are special methods to handle linear objectives and linear constraints (linear programming), and quadratic objectives and linear constraints (quadratic programming).

2.3.1 Disadvantages

Even though the gradient based methods are very fast and quick to converge to a local optimum, they suffer several limitations.

1. The gradient based methods are based on the assumption of smoothness of the objective and the constraint functions. These assumptions are often not valid in real-life optimization problems. In addition, disconnected design

spaces and/or feasible regions are not handled in these methods.

2. The numerical methods require gradient information. This information is often not available or is very difficult to compute. The simplest way of calculating gradient information is based on finite difference. For a function of n design variables, the gradient vector calculation at a point using finite difference requires $n+1$ function evaluations and this corresponds to evaluating $n+1$ designs in the neighborhood. This cost can be prohibitive for computationally expensive simulations and in addition the accuracy of the gradient is dependent on the accuracy of the simulation.

3. The numerical methods are essentially local search methods and can find only local minima.

4. These methods are developed for single objective optimization methods and do not handle multi-objective optimization problems.

5. None of the gradient methods can be used for integer or discrete design variables.

6. All gradient based methods except exterior penalty function based methods require a feasible starting point.

In addition to the gradient based methods, there are direct search methods (also called derivative-free methods or zero-order methods) that do not use any gradient information. These methods explore the local neighborhood to identify the suitable direction for the search. These include Hooke and Jeeves patten search [22] and Nelder-Mead simplex search [23]. These methods can be used for problems where the gradients are not available, but essentially they are also local search methods and solve only single-objective optimization problems.

2.4 Heuristic methods

Reeves and Beasly [24] defined a heuristic in the context of combinatorial optimization in 1993 as follows –

> *"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."*

The definition is applicable to heuristic methods today, and the methods have garnered enough respect to be applied to real-life problems. The main thrust of heuristic methods is the ability to search for global optima.

The methods that specifically try to escape the local optimum include simulated annealing and tabu search. Simulated annealing (SA) developed by Kirkpatrick, Gelatt and Vecchi [25] was inspired from the heating and the controlled cooling of metals to increase the crystal size and reduce defects. In SA, the neighborhood is explored using random search moves. During the search, the solutions that are worse off than the current best solution are accepted using non-zero probability in the hope to find a better solution. A similar method developed by Glover [26], Tabu Search (TS), tries to overcome the same limitation by using a tabu list of solutions that are visited in the past and prohibits the algorithm from searching near those solutions.

Another class of heuristic methods are the global search methods that can search different parts of the design space simultaneously. Among the earliest global search methods are genetic algorithms (GA) developed by Holland [27] and later by Goldberg [28], evolutionary programming (EP) developed by Fogel [29], and evolution strategy (ES) developed by Rechenberg and Schwefel [30]. Genetic

algorithms mimic the natural evolution process. The candidate solutions are encoded as bit-strings representing the genes. A population of these solutions undergoes selection to choose the parents, and the parents undergo crossover and mutation operations to create new candidate solutions as a offspring population. Better solutions from the parent population and the offspring population are retained to create ideally a superior population for the next generation. This process is repeated over a number of generations resulting in a search through the design space. The other methods, EP and ES, follow the evolution process using only mutation. A method of differential evolution (DE) by Price and Storn [31] uses differential mutation operation for the evolution. All these methods are collectively identified as evolutionary algorithms (EAs).

Swarm intelligence methods on the other hand are modeled after the collection of individuals and their social behaviour. Ant colony optimization (ACO) proposed by Dorigo [32] is based on the behavior of ants seeking a path between the colony and the food source. Particle swarm optimization (PSO) developed by Kennedy and Eberhart [33] tries to simulate social learning by exchange of information between the swarm members or particles. Other swarm intelligence methods include cultural algorithm [34], bacterial foraging [35] and harmony search [36].

2.4.1 Advantages

Since the heuristic methods use only values of the objectives and the constraints and do not need gradient information, they have the following advantages over the numerical methods.

1. Heuristic methods can be used to solve optimization problems of non-smooth

(not continuous and/or not differentiable) functions and also handle problems with disjoint feasible solution spaces.

2. Optimization problems with discrete and integer design variables can be handled using heuristic methods.

3. All the heuristic methods are global search methods and can find the global optimum solutions.

In addition, evolutionary algorithms and swarm intelligence methods have the following advantages.

1. These methods are population based methods and explore different regions of the design space simultaneously with a population of solutions.

2. These methods can generate the Pareto optimal solutions for multi-objective optimization problems in a single run.

3. These methods are inherently parallel and can be easily implemented on parallel computing platforms for faster execution.

4. The starting solutions need not be feasible solutions to the optimization problem. A variety of constraint handling methods are used to steer the search towards the feasible region.

These advantages make heuristic methods ideal candidates for solving single and multi-objective optimization problems. In this thesis evolutionary algorithms are used solve optimization problems.

2.4.2 Disadvantages

Most of the advantages of heuristic methods (like evolutionary algorithms and swarm intelligence) are a direct consequences of using a population of candidate solutions. The use of a population of solutions in evolutionary algorithms gives rise to following issues.

1. In evolutionary algorithms, new candidate solutions are generated through crossover and mutation operations. These stochastic recombination processes do not always produce a better solution and worse solutions get discarded every generation. As a result, the convergence of EAs is very slow and they need to be run for large number of generations to find the optimum solutions.

2. As evolutionary methods are population based and need to be evolved for large number of generations, the total number of function evaluations (to calculate the objective and the constraint values) can be very large. This can be prohibitive for problems that require use of time-consuming simulations.

These issues are further discussed in next chapter. Approximation model assisted evolutionary algorithms are proposed in Chapter 4 and a novel constraint handling method is proposed in Chapter 5 to overcome the limitations of evolutionary algorithms. A general framework for evolutionary algorithms is introduced in next section.

2.5 Evolutionary Algorithm Framework

A generic framework for an evolutionary algorithm is shown in Algorithm 2.1. The algorithm starts with a population (P_1) of N candidate solutions initialized by random sampling from the design space. Each candidate solution of the population is then evaluated to find the corresponding values of the objective and the constraint functions. Then, the solutions in the population are ranked based on the "fitness" value, which is a measure of how good a solution is and is derived from the objective and the constraint values. The next few steps (lines 5-8) are repeated for N_G generations. An offspring population, C_i, is evolved using recombination operation from the current (or parent) population P_{i-1}. The new solutions in the offspring population are evaluated to calculate the objective and the constraint function values. The combined solutions from both the populations are then ranked using the fitness values. Based on the ranks, the *best* solutions from the parent population P_{i-1} and the offspring population C_i are retained to form the population for the next generation P_i. As the processes of evolution, evaluation, ranking and reduction are repeated, the population in successive generations contains better and better solutions.

Algorithm 2.1 EA 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**

Among various evolutionary algorithms available today, Non-dominated Sort-

ing Genetic Algorithm II (NSGA-II) [18] is the most popular algorithm for constrained multi-objective optimization. In this study, NSGA-II with real coding of design variables is used as the canonical EA. The various steps of NSGA-II are explained in the context of the EA framework in the following sections.

2.5.1 Initialization

All the individuals in the population are initialized by random sampling from the design space. A value for each design variable is sampled uniformly between the lower and the upper bound for the variable as given in Equation 2.4.

$$x_i = \underline{x_i} + \mathcal{U}[0,1]\ (\overline{x_i} - \underline{x_i}) \qquad 1 \leq i \leq n \tag{2.4}$$

where x_i denotes the initialized variable, $\underline{x_i}$ and $\overline{x_i}$ are lower and upper bounds for the variable, and $\mathcal{U}[0,1]$ is an uniform random number lying between 0 and 1.

2.5.2 Evaluation

For each solution in the population, the values of the objective and the constraint functions are evaluated using appropriate simulation or analysis. The fitness of a solution is calculated based on the objective and the constraints values as follows:

1. For a feasible solution, the fitness corresponds to the objective value(s).

2. For an infeasible solution, the fitness corresponds to the largest constraint violation value for constraints that are violated.

2.5.3 Evolution

In NSGA-II, 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 crossover operator. In mutation, one or more variables of a solution are perturbed. To select a parent, two solutions are picked randomly from the current population and the solution with better fitness is considered. This comparison between two solutions is referred as binary tournament and is described below.

Selection

Binary tournament between two solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ is performed as follows.

1. If $\mathbf{x}_1$ is feasible and $\mathbf{x}_2$ is infeasible, $\mathbf{x}_1$ is selected and vice versa.
2. If both $\mathbf{x}_1$ and $\mathbf{x}_2$ are infeasible, the one for which the value of the maximum constraint violation is smaller is selected.
3. If both $\mathbf{x}_1$ and $\mathbf{x}_2$ are feasible and $\mathbf{x}_1$ dominates $\mathbf{x}_2$, $\mathbf{x}_1$ is selected and vice versa.
4. If both $\mathbf{x}_1$ and $\mathbf{x}_2$ are feasible and neither dominate the other, one of $\mathbf{x}_1$ and $\mathbf{x}_2$ is selected at random.

Crossover

The crossover operation is performed using simulated binary crossover (SBX) [37]. Two offspring solutions $\mathbf{y}_1$ and $\mathbf{y}_2$ are created from parents $\mathbf{x}_1$ and $\mathbf{x}_2$ by operating

on one variable at a time as shown in Equation 2.5.

$$\begin{aligned} y_i^1 &= 0.5\left[(1+\beta_{q_i})\,x_i^1 + (1-\beta_{q_i})\,x_i^2\right] \\ y_i^2 &= 0.5\left[(1-\beta_{q_i})\,x_i^1 + (1+\beta_{q_i})\,x_i^2\right] \end{aligned} \tag{2.5}$$

where β_{q_i} is calculated as,

$$\beta_{q_i} = \begin{cases} (2u_i)^{1/\eta_c+1}, & \text{if } u_i \leq 0.5, \\ \left(\frac{1}{2(1-u_i)}\right)^{1/\eta_c+1} & \text{if } u_i > 0.5. \end{cases} \tag{2.6}$$

and where u_i is the uniform random number in the range $[0, 1)$ and η_c is the user defined parameter, *Distribution Index for Crossover*. Probability of crossover (P_c) determines how often the crossover operation is performed.

Mutation

The polynomial mutation operator [38] is used for mutation. In the mutation operation, the value of one or more variables is randomly perturbed as given in Equation 2.7.

$$y_i = x_i + (\overline{x_i} - \underline{x_i})\,\bar{\delta}_i \tag{2.7}$$

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

$$\bar{\delta}_i = \begin{cases} (2r_i)^{1/(\eta_m+1)} - 1, & \text{if } r_i < 0.5, \\ 1 - [2(1-r_i)]^{1/(\eta_m+1)}, & \text{if } r_i \geq 0.5. \end{cases} \tag{2.8}$$

and 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*. The number of solutions undergoing mutation operation are determined by probability of mutation (P_m).

2.5.4 Ranking

Individual solutions in a population are ranked based on their fitness value. Feasible solutions are considered better than infeasible solutions and are ranked higher. Feasible and infeasible solutions are ranked separately.

For single objective optimization, feasible solutions are sorted based on the objective value as the fitness. For multi-objective optimization the two solutions are compared using dominance relationship as described in Section 2.2.2. NSGA-II uses non-dominated sorting and crowding distance sorting procedure [18] to rank feasible solutions with multiple objectives. In non-dominated sorting the solutions are arranged in multiple non-dominated fronts. In each non-dominated front, the solutions are non-dominated, whereas the solutions in one front dominate the solutions from the other front. Within a non-dominated front, the solutions are ranked based on a diversity measure, crowding distance [18].

For infeasible solutions the fitness corresponds to the maximum constraint violation. If more than one constraints are violated for a solution, the largest constraint violation value is the maximum constraint violation. Infeasible solutions are sorted in the increasing order of maximum constraint violation value.

2.5.5 Reduction

The reduction process is used to retain N best solutions 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.

1. If there are more than N feasible solutions,

    - N feasible solutions are selected in the order of non-dominated fronts and decreasing crowding distance in each front.

2. If the feasible solutions are less than or equal to N,

   - all the feasible solutions are selected in the order of non-dominated fronts and decreasing crowding distance in each front, and
   - the remaining solutions are selected from infeasible solutions in the order of minimum value of maximum constraint violation.

2.6 Performance Measurement

The performance of different optimization algorithms can be compared based on the best solutions obtained by each algorithm. In the case of single objective optimization, a better value of the objective indicates a better performing algorithm. In the case of multi-objective optimization, two sets of non-dominated solutions need to be compared. There are a number of metrics proposed in the literature to judge the goodness of non-dominated sets. There are two types of metrics – absolute and relative. Absolute metrics use the known Pareto optimal set to compare the non-dominated solutions obtained from an optimization algorithm. Relative metrics are used to compare the results from two optimization algorithms in the absence of the Pareto optimal set of solutions. In this study, an absolute metric, *displacement* metric is used to assess the performance of optimization algorithms.

2.6.1 Displacement

Let the Pareto optimal front be denoted by F^* and the non-dominated solutions obtained by an optimization algorithm is denoted by F.

The generational distance is the measure of the distance between the Pareto

optimal front and the non-dominated solution set [39]. The definition of generational distance is given in Equation 2.9.

$$\mathrm{GD}(F, F^*) = \frac{1}{|F|}\left(\sum_{i=1}^{n}(d_i)^p\right)^{\frac{1}{p}}, \tag{2.9}$$

where d_i is the Euclidean distance between the ith non-dominated solution of F and the nearest member of the Pareto front F^*, and $|F|$ is the number of elements in the non-dominated set. Most often $p = 2$. The smaller the generational distance, the closer the solutions are to the Pareto optimal front.

The displacement metric [40] is used as an indication of not only how close the non-dominated solutions are to the Pareto optimal front, but also to measure the distribution or the spread of the non-dominated solutions. The displacement metric is defined as inverse generational distance as shown in Equation 2.10.

$$\mathrm{Displacement}(F) = \mathrm{GD}(F^*, F) \tag{2.10}$$

In the case of two non-dominated solution sets with the same generational distance, the set whose solutions are well distributed with respect to the Pareto optimal front will have smaller Displacement. The differences between generational distance and displacement metric are illustrated in Figure 2.3. The figure shows Pareto optimal front (POF) marked as solid line and three non-dominated sets of solutions (ND1, ND2 and ND3). The solutions in ND1 and ND2 are equidistant from the Pareto optimal front, so generational distance for ND1 and ND2 is the same. The solutions in ND2 have better spread as compared to solutions in ND1 and the displacement value for ND2 is lower (better) than ND1. Solutions in ND3 are further away from the Pareto optimal front than ND1 and ND2, but are

well distributed as compared to ND1 and ND2. Subsequently, the generational distance for ND3 is higher than ND1 and ND2. The displacement value for ND3 is the same as that of ND1. Although the distances between the POF solutions covered by solutions in ND1 and the closest solutions (corresponding to each solution in POF) in ND1 is very small, the distances between the other solutions of POF (far away from ND1) and solutions in ND1 are quite large. The solutions in ND3 are spread evenly along the POF and have the same average distance between the POF solutions and corresponding closest solutions in ND3. The sets ND1 and ND3 are equivalent according to the displacement metric.

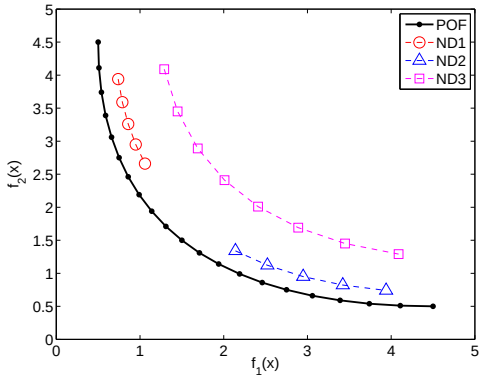


Figure 2.3: Non-dominated solutions with equivalent displacement metric

2.7 Summary

An overview of single and multi-objective optimization is presented. The solution of a single objective optimization problem is the global optimum, whereas a multi-objective optimization problem (with no preference for any of the objectives) can have a large number of equally good solutions that form the Pareto optimal front.

Numerical optimization methods require gradient information to search for the optimum solutions in the design space. The applicability of numerical methods is limited to problems involving smooth functions. The calculation of gradient information is often difficult and requires additional function evaluations (when finite difference is used).

Evolutionary algorithms are heuristic methods that do not need gradient information and can be easily applied to non-smooth functions involving discrete, integer, real or mixed design variables. They can also find solutions to a multi-objective optimization problem in a single run. However, EAs are population based methods and require evaluation of large number of candidate solutions. To improve the performance of EAs, spatial surrogates and constraint handling methods are proposed in following chapters. The proposed algorithms are based on a generic EA framework introduced in this chapter. The most popular evolutionary algorithm, NSGA-II, is used in this study as the canonical EA. The various components of EA are explained with reference to NSGA-II.

Chapter 3

Approximations in EA

3.1 Overview

Evolutionary algorithms are population based heuristic methods that can be applied to solve constrained nonlinear optimization problems. They can handle single and multi-objective optimization problems seamlessly. Since the '80s, EAs have been a common choice for multi-objective optimization problems as they result in a set of non-dominated solutions in a single run. Evolutionary algorithms require evaluations of numerous candidate solutions during the course of the search. Such an approach turns out to be computationally prohibitive for problems requiring computationally expensive analysis and there is a growing interest in the use of approximations to reduce the total number of evaluations using the expensive analysis.

The different ways in which approximations have been used to solve an optimization problem are presented in Section 3.2. One of the ways is function approximation in which surrogate models are used to approximate the response of the expensive functions is described in Section 3.3. This section is organized

in several topics – the sampling methods, surrogate modeling techniques, local/global surrogate models and ensembles and surrogate training. In addition to replacing the function evaluations, surrogate models can be used to improve the performance of various operators in EA. The research done in the area of informed operators is presented in Section 3.4. Researchers have used a local search based on the surrogate models in memetic algorithms, details of which are given in Section 3.5. The chapter concludes with the summary of the limitations of various approaches using surrogate models in Section 3.6.

3.2 Levels of Approximations

Barthelemy [41] identified two ways of using approximations to solve an optimization problem. They are:

1. *Problem approximation*, where the overall optimization problem is replaced with a problem that is easier to solve and the solution to this simpler problem approximates the solution to the original problem.

2. *Function approximation*, where the objective and constraint functions are approximated using simpler and explicit functions.

Problem approximation is commonly used in CFD where Navier-Stokes equations governing the flow are reduced to Euler equations or potential flow equations based on the certain assumptions of flow properties, or 3-D axi-symmetric flow is solved as 2-D flow. For simulations requiring discretization, the problem approximation can be in terms of the level of discretization. For example, FEM model with millions of degrees of freedom can be solved using simpler elements and fewer degrees of freedom or a CFD simulation is differentiated based on the

mesh size.

In function approximation, surrogate models are built for each of the objective and the constraint functions. This is the most common type of approximation used as it does not depend upon the problem domain. In addition, significant developments in the area of surrogate modeling techniques can be directly applied to the problems at hand.

Another type of approximation has been identified in the context of EAs by Jin [42] is *evolutionary approximation*. In this approach the fitness of an offspring is derived from the fitness of the parent(s). In fitness inheritance, the objective and the constraint values for a solution are derived as a weighted sum of the respective objective and constraint values of the two parent solutions [43, 44, 45]. In fitness sharing, the fitness of similar solutions is degraded to improve the diversity of the solutions in the population [46].

Problem approximation requires the specific disciplinary expertise to identify replacement theories or simplifications that can be applied with valid assumptions. Evolutionary approximation, though generic in approach, requires specific implementation of EA. On the other hand, functional approximation is the most generic approximation technique and does not need specific disciplinary knowledge.

3.3 Function Approximation

Function approximation involves building surrogate models that can approximate the response for the functions and are computationally cheaper than the original function evaluations. A generic function can be mathematically represented as $F(\mathbf{x}, y) = 0$, where $\mathbf{x}$ are m independent variables and y is the response. A surro-

gate model is trained using the responses $y_1, y_2, \ldots, y_N$ to solutions $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ sampled in the design space. The surrogate model is an explicit function of the form $y = f(\mathbf{x})$ that mimics the response y. The number of samples required to train the surrogate model is often related to the complexity of the function being approximated. The more complex the function, the more samples are required to adequately represent the response. Since the number of samples dictates the number of actual function evaluations, it is essential to keep the number of samples as low as possible for function evaluations requiring computationally expensive simulations. For a given number of samples, various sampling techniques try to position those samples in the design space to improve the quality of the surrogate model built using those samples.

3.3.1 Surrogate Modeling Techniques

There are many different types of surrogate models including response surface methods, artificial neural networks, Kriging, support vector machines.

Response Surface Method

Response surface method (RSM), also known as polynomial regression or linear regression, uses first or second order polynomial models to fit the data [47]. A second order quadratic polynomial model can be written as

$$y(\mathbf{x}) = \beta_0 + \sum_{i=1}^{m} \beta_i \, x_i + \sum_{i=1}^{m} \beta_{ii} \, x_i^2 + \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \beta_{ij} \, x_i \, x_j \tag{3.1}$$

where $\beta_0, \beta_i, \ \beta_{ij}$ are the unknown parameters of the model that are determined from the observations. The same equation can be written in a vector form as $y(\mathbf{x}) = \mathbf{f}^T\mathbf{b}$. The vector $\mathbf{f}$ contains all the terms of $x_1, x_2, \ldots, x_m$ and vector $\mathbf{b}$

contains all the unknown coefficients. The least squares estimate of $\mathbf{b}$ is given by,

$$\hat{\mathbf{b}} = (\mathbf{F}^T\mathbf{F})^{-1}\mathbf{F}^T\mathbf{Y}, \tag{3.2}$$

where $\mathbf{F}$ is a matrix containing N rows, each row is a vector $\mathbf{f}^T$ evaluated at an observations and $\mathbf{Y}$ are the observed responses.

Response surface method is the simplest of the surrogate models and has been used by Wang [48, 49] as a surrogate model in EA, Lian and Liou [50] for shape design with CFD, Rickards *et al.* [51] for composite shell design. The main drawback of RSM is the large number of training solutions required. For m-dimensional data, a quadratic RSM model has $(m + 1)(m + 2)/2$ unknown coefficients that need to be determined and at least this number of solutions are required for training. This corresponds to 66 points for 10-D data, 231 points for 20-D data and 496 points for 30-D data. Response surface methods can be used very effectively in cases of a small number of design variables, but the number of training samples rapidly increases for a large number of design variables rendering it impractical.

Artificial Neural Network

Artificial Neural Networks (ANNs) (or Neural Networks as they are referred in literature) are effective in modeling non-linear relationship between input and output [52]. They have been used to model the functional relationships when mathematical models do not exist [53] and also as surrogate models to approximate nonlinear functions [54, 55, 56]. The two types of ANNs commonly used are radial basis function networks and multilayer perceptrons.

Radial Basis Function (RBF) networks are a popular choice for interpolating

functions [57]. The model for the response y is given by

$$y(\mathbf{x}) = \sum_{i=1}^{k} w_i \, \phi(\|\mathbf{x} - \mathbf{x}_i\|) \tag{3.3}$$

where $\phi(\cdot)$ are the radial basis functions, $\|\cdot\|$ is usually the Euclidean norm and w_i are the unknown weights. A radial basis function is symmetric around its associated center, in this case $\mathbf{x}_i$. 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.

Multi-layer perceptrons (MLP) also belong to the ANN class and consist of one input layer, one output layer and one or more hidden layers of neurons. Usually a single hidden layer of neurons is sufficient. The number of neurons in the input and output layers correspond to the number of input variables and responses respectively. The number of neurons in the hidden layer are determined by experience and there is no theoretical basis for the choice of the number of neurons. Often researchers do not specify the number of neurons or how the number was arrived at [58, 59]. In addition, NNs often overfit the data leading to very good prediction for seen data, but poor prediction on unseen data [52]. To prevent over-fitting of data, techniques like regularization need to be used [60]. Owing to these difficulties MLP cannot be used very easily as a black-box surrogate model. On the other hand RBF network has been used widely in engineering problems including airfoil design [61, 62], truss design [63],

bulk carrier design [64].

Kriging

Kriging, also known as design and analysis of computer experiments (DACE) or Gaussian process), is a spatial approximation technique with the form

$$y(\mathbf{x}) = \mu(\mathbf{x}) + Z(\mathbf{x}), \tag{3.4}$$

where, $\mu(\mathbf{x})$ is a regression model and $Z(\mathbf{x})$ is zero-mean random field with unknown covariance structure $\mathbf{V}$ [65]. The regression model $\mu(\mathbf{x})$ can be of the form as shown in Equation 3.1. The covariance function is given by

$$\text{cov}\left[Z(\mathbf{x}_1), Z(\mathbf{x}_2)\right] = \sigma^2 R(\mathbf{x}_1, \mathbf{x}_2),$$

where σ^2 is the process variance and $R(\mathbf{x}_1, \mathbf{x}_2)$ is the correlation between $Z(\mathbf{x}_1)$ and $Z(\mathbf{x}_2)$. The correlation between any two observations is modeled with a spatial correlation function. The Gaussian function, often used as a spatial correlation function, is given by

$$R(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\sum_{k=1}^{m} \theta_i \ |x_{i,k} - x_{j,k}|^p\right)$$

where θ_i and p are the hyper-parameters of the correlation function which are determined using maximum likelihood estimation (MLE). The regression function can be represented in the vector form $\mu(x) = \mathbf{f}^T\,\mathbf{b}$. The value of $\mathbf{b}$ is determined using the generalized least squares estimate as

$$\hat{\mathbf{b}} = (\mathbf{F}^T\mathbf{V}^{-1}\mathbf{F})^{-1}\mathbf{F}^T\mathbf{V}^{-1}\mathbf{Y},$$

and the best linear unbiased predictor (BLUP) for y is given by

$$\hat{y}(\mathbf{x}) = \mathbf{f}^T\hat{\mathbf{b}} + v^T(\mathbf{x})\,\mathbf{V}^{-1}(\mathbf{Y} - \mathbf{F}^T\hat{\mathbf{b}}),$$

where $\mathbf{F}$ is the matrix of N rows of vector $\mathbf{f}^T$ evaluated at each observation and v is the covariance between the design vector $\mathbf{x}$ and the observations $\mathbf{x}_1, \ldots, \mathbf{x}_k$ used for training. The value of the process variance using maximum likelihood estimate (MLE) is

$$\hat{\sigma}^2 = \frac{1}{N}(\mathbf{Y} - \mathbf{F}^T\hat{\mathbf{b}})\mathbf{R}^{-1}(\mathbf{Y} - \mathbf{F}^T\hat{\mathbf{b}}).$$

In ordinary Kriging, the regression function $\mu(\mathbf{x})$ is taken as a constant and the corresponding value of that constant is

$$\hat{\mu} = (\mathbf{1}^T\mathbf{V}^{-1}\mathbf{1})^{-1}\mathbf{1}^T\mathbf{V}^{-1}\mathbf{Y},$$

and the BLUP for $y(\mathbf{x})$ is given by

$$\hat{y}(\mathbf{x}) = \hat{\mu} + v^T(\mathbf{x})\,\mathbf{V}^{-1}(\mathbf{Y} - \hat{\mu}).$$

In the last decade Kriging has become quite popular due to the ability of Kriging models to represent non-linear functions accurately. Kriging has been used with EAs to speed up the convergence for numerical test problems [66, 67, 68, 69] and in engineering applications including satellite boom optimization [70], piezoelectric actuator design [71], airfoil shape design [72], welded beam design [73].

In the kriging model, the error in prediction can be estimated as

$$MSE[\hat{y}(\mathbf{x})] = \hat{\sigma}^2 \left[1 - v^T(\mathbf{x})\mathbf{V}^{-1}v(\mathbf{x}) + \frac{(1 - \mathbf{F}^T\mathbf{V}^{-1}v(\mathbf{x}))^2}{\mathbf{F}^T\mathbf{V}^{-1}\mathbf{F}}\right]. \quad (3.5)$$

Jones *et al.* [74] treated this error estimate as the uncertainty in prediction and developed an efficient global optimization (EGO) algorithm. In EGO, the uncertainty in prediction is converted into a probability value of finding better solutions (referred as expected improvement) and the design space is incrementally sampled at the most probable locations. The EGO algorithm has been extended to solve multi-objective optimization problems by Knowles as ParEGO [75].

Comparison

Many researchers have compared various surrogate modeling techniques to find significant merits of using one type of surrogate model against the other. Carpenter and Barthelemy [76] compared linear regression and ANN; Simpson *et al.* [77] and Wilson *et al.* [71] compared RSM versus Kriging; Rasheed, Ni and Vattam [78] compared RSM, RBF and NN. All the studies conclude that the performance of multiple surrogate models used in EA is similar and there is no single surrogate modeling technique better suited for all optimization problems.

3.3.2 Sampling Techniques

The statistical field of design of experiments (DOE) is devoted to locating the samples in the design space [79]. The popular sampling techniques include factorial design, central composite design and D-optimal designs. These sampling methods when used for linear regression improve the quality of the surrogate model by minimizing the variance of estimate error. The number of samples re-

quired for 2-level full factorial design and central composite design are 2^m and 3^m respectively for m independent variables. Orthogonal arrays [80, 81] popularized by Taguchi are fractional factorial designs and require a much smaller number of samples. The number of samples is dictated by the number of independent factors or variables, levels for each factor and specific 2-factor interactions to be studied. Latin hypercube sampling by McKay [82] is a statistical method of sampling from a multi-dimensional distribution. In this method N points are distributed in the design space such that there is a single point in each of N (uniformly distributed) planes normal to the coordinate axis. (In two-dimensions, it is similar to placing eight queens on the 8×8 chess board, such that along any row or column there is only a single queen.) Any number of samples can be chosen independent of the number of variables. Latin hypercube sampling can be based on orthogonal arrays to distribute points evenly with equal density within subspaces [83].

Within EA, many of these techniques are used to sample the initial population and train the surrogate model using those samples or use the off-line training where the surrogate model building exercise is external to EA. Wilson *et al.* [71] used central composite design and Latin hypercube sampling, Knowles used Latin hypercube sampling in ParEGO [75] and Booker *et al.* [84] used orthogonal array based Latin hypercube sampling. All the mentioned sampling methods are equally good in locating samples in the design space. Researchers tend to use a sampling technique based on personal preference.

The initial surrogate model trained with the limited number of training samples is often not good enough to accurately represent all the regions of the design space. To overcome this limitation, the surrogate model is updated with new samples or recreated during the optimization process. Then the question is how to sample the training data using the new solutions. One approach is to

discard the previous training data and replace it with the new samples from the current population. The new samples are chosen from the current population using k-Means clustering [63, 85, 86]. The other approach is to add new samples to the existing training data. Nain and Deb [59] use actual evaluations from a few generations as the new training data and update the ANN model for curve fitting problem. Ratle [87] compared six different strategies for updating the training samples for Kriging model. The strategies include replacing the existing training data completely, replacing a few of the samples from the existing training data and adding new samples to the training data. Ratle has also reported that adding new samples to the training data produced better results.

3.3.3 Global and Local Surrogate Models

Theory of DOE and RSM is used to characterize a physical system through careful experimentation and observation of responses [79]. The main purpose of the exercise was to gain sufficient understanding of the physical system in the form a surrogate model. The development of neural networks is similarly motivated by the need to capture the input-output relationships of a non-linear system [52]. The surrogate models built under these motivations are called *global surrogate models*. These models usually predict the general (or global) trends in the system response very well. The global surrogate models have been successfully used in EAs for various design problems – with RSM for compressor blade design [50]; with Kriging for beam design [88]; and with ANN for structural shape optimization [89], and hydrofoil optimization [90]. If an accurate global surrogate model can be built, it can drastically reduce the cost of the optimization process. But, creating a global surrogate model that can accurately represent the local variations along with the global trends is a challenging task. The

curse of dimensionality and the non-linearity in system response necessitate large numbers of training samples to accurately represent the system response. It is impractical to generate large numbers of training samples when expensive experiments or simulations are required to observe the system response and consequently, accurate global surrogate models cannot be built.

Local surrogate models, on the other hand, capture the system response in a small region of the design space. A local surrogate model is built using the training samples in the neighborhood of the current search location and is used only within that small neighborhood. If the search moves outside this region, then the current local surrogate model is discarded. Willmes *et al.* [91] used Kriging based local surrogate models built with 10 solutions in the neighborhood; Regis and Shoemaker [92] used $(m+1)(m+2)/2$ points to train a local surrogate using ANN. These approaches are limited by the availability of a sufficient number of solutions in the neighborhood where the surrogate model is built. An alternative is to start with a complete design domain and progressively reduce the size of the design domain. In the adaptive RSM approach Wang *et al.* [48] achieve space reduction using cutting planes, and Wang and Simpson [49] use fuzzy clustering to identify the region of interest.

Hierarchical Models

Hierarchical models are a combination of global and local surrogate models. A global surrogate model is used to either identify interesting regions of design space or to screen candidate solutions that are to be evaluated. Wang and Simpson [49] have used RSM/Kriging to identify the promising region of the design space and use Kriging as the local surrogate within that region. Zhou *et al.* [93, 94] have used Kriging as a global surrogate to identify promising solutions in the popu-

lation and those solutions are improved based on a RBF based local surrogate model built using samples in the neighborhood. A similar approach using RBF based global and local surrogate models is proposed by Tenne and Armfield [95]. Hierarchical models overcome the limitations of the global surrogate models using local surrogate models to capture local variations, thus getting the best of global and local models.

3.3.4 Surrogate Ensembles

To improve the prediction accuracy with limited training samples, multiple surrogate models can be used in place of a single surrogate model. Common use of multiple surrogates is in the form of surrogate ensembles, where a collection of surrogate models are used. These surrogate models can be of the same type or different types. Similar type of surrogate models are trained simultaneously with varying model parameters, or using different subsets of training samples by techniques such as bagging [96], boosting [97] and cross-validation. Abbass [98] used a multi-objective formulation for ANN training and the Pareto optimal ANNs to form an ensemble. Hamza and Saitou [99] have used RSM ensemble built using different subsets of training samples.

To combine the predictions from individual surrogate models, a simple average or a weighted average is used [86]. Zerpa *et al.* [100] and Goel *et al.* [101] have used surrogate ensemble using RSM, RBF and Kriging. The predicted response is a weighted average of the individual surrogate responses. The two approaches differ in the way the weights for individual surrogate models are determined for averaging. Zerpa *et al.* [100] calculate the weights based on prediction variance minimization and Goel *et al.* [101] use goodness of data to propose three schemes for global weights selection. Zhou *et al.* [102] reported the use of RSM and RBF

as local surrogate models and pick the best solution resulting from multiple local searches (as many as number of surrogate models). They even propose using a surrogate ensemble as one of the approximation models.

Zhao, Gao and Yang [103] have reported great improvement in the generalization performance of a neural network ensemble in data classification application. Lim *et al.* [104] reported that ensemble of surrogate models and multiple surrogate models yield improved solution quality for the same computational budget. Goel *et al.* [101] concluded that the surrogate model that approximates a response in the best way is dependent on the training data itself and an ensemble of surrogates may be more robust approximation method.

3.3.5 Surrogate training

One of the concerns when training surrogate models is that the global optimum of the surrogate model should coincide with the global optimum of the fitness function and an EA should be able to converge to that solution. It is difficult to build surrogate models that are accurate everywhere in the domain for functions with a large number of variables and using limited training samples [42]. Consequently, the global surrogate models are often retrained using additional training samples or multiple local surrogate models are trained with new training samples. The process of building a series of surrogate models sequentially in the optimization process is referred to as *model management* [84]. In the context of EAs, the use of surrogate models along with the actual fitness evaluation is called *evolution control* [105].

In fixed evolution control, the surrogate model is trained periodically every few generations of EA and for rest of the generations the surrogate model is used instead of actual evaluations. Jin *et al.* [106] identified two strategies for

the surrogate training – individual based and generation based. In individual based one or more solutions are evaluated using the actual analysis and used as new training data, in generation based the entire population is evaluated using the actual analysis and used as training data. Bull [107] evaluated the best individual in the population using the actual analysis, replaced the worst solution in the training data with it, and retrained the NN model every few generations. D'Angelo and Minsci [108] distinguished between the choice of best individuals in the population for training as best strategy and the random selection of individual as random strategy. They use best strategy to update the Kriging surrogate model. Nain and Deb [59] used the actual analysis for n generations and use the solutions in those n generations as the training data. The next $Q-n$ generations use the trained ANN model instead of the actual analysis and the process is repeated after Q generations. A similar method of using K generations with actual analysis and S generations with trained RBF model has been used by Ray and Smith [64].

As the surrogate model is updated using new training data in the evolutionary search, the function landscape changes and gives rise to a dynamic optimization problem [109]. The function values of solutions evaluated using the surrogate keep changing as the surrogate models are updated, hence it is necessary to introduce elitism to retain the true best solutions in the population. El-Beltagy *et al.* [88] maintained truly evaluated solutions in the population and compared new solutions using true analysis.

Surrogate Validation and Quality

Out of many approaches published in the literature for combining surrogate models in EA, very few actually have looked at the quality of surrogate models

built. It is important to validate the surrogate models to ensure that the prediction using the surrogate models approximately represents the function response. Otherwise, it is possible to identify a solution A better than a solution B based on the predicted fitness, but in reality it's the other way round. This behavior is known as ill-validation [110]. Wilson *et al.* [71] have used absolute error, average error and R^2 statistics [47] to measure the accuracy of RSM and Kriging models and if the validation errors are very large, either additional training samples are used or the size of the design space is reduced. Lim *et al.* [104] used root mean square error (RMSE) and correlation coefficient to relate the performance of surrogate assisted evolutionary algorithms to surrogate accuracy using RSM, RBF, Kriging and ANN models. Based on the four benchmark functions studied, they found that Kriging predictions had the least RMSE, whereas RBF predictions had better correlation values. Although, RSM did not exhibit good accuracy on both measures, the best solution quality was found using RSM. Tenne and Armfield [111] compared the performance of different accuracy assessment methods – holdout, 10-fold cross validation, leave-one-out cross validation (LOOCV) and 0.632 bootstrap estimate using RSM, Kriging and RBF surrogate models. They found that LOOCV lead to more accurate surrogate models for up to ten dimensions, but no one method is better for surrogate models of more than 10 variables.

3.4 Informed Operators

In addition to prediction of the fitness of the candidate solutions, surrogate models are also used to improve the efficiency of sampling and evolution operations (crossover and mutation) and such operators are referred to as informed op-

erators. Rasheed and Hirsh [112] have identified four types of informed operators – informed initialization, informed mutation, informed crossover and informed guided crossover. In an informed operator, the operation is carried out multiple types and the best candidate is selected according to the best fitness predicted by the surrogate model. In informed mutation, multiple solutions are created from the selected solution using mutation and the solution with the best fitness as predicted by the surrogate model is chosen as the mutated solution. Mutoh *et al.* [113] used crossover operator multiple times, predicted the fitness based on ANN surrogate model and retained the best solution as the result of the crossover operation. Kriging based surrogate model has been used to screen the solution in sampling [114] and to select best offspring in crossover [115].

The other approach is to use the surrogate models to search for the best candidate during the crossover or mutation operator. Anderson and Hsu [116] used approximations within their crossover operator as follows. To create an offspring solution from two parents, quadratic models for fitness are created using each design variable and the locations of the minimum obtained from each quadratic model are used as the coordinates of the offspring. In surrogate deterministic mutation, Abboud and Schoenauer [117] select promising solutions around the solution undergoing mutation, build a surrogate model using those solutions, and use a gradient based search to find a solution with the best fitness using the surrogate model.

3.5 Memetic Algorithms

In an attempt to improve the rate of convergence of evolutionary algorithms in general, memetic algorithms (MAs) have been reported in literature [118, 119].

These are hybrid algorithms, which couple local search techniques with EA to improve the efficiency of EA. To improve a solution, a local approximation model is built around that solution and gradient or heuristic search is applied to improve the solution using the approximation model. This is referred to as *Lamarckian* evolution.

Liang *et al.* [120, 121] used a local search for multimodal functions to improve the convergence in EA. Studies by Ku *et al.* [122] indicate that it may be worthwhile to apply local search to all the individuals in the population if the local search is computationally inexpensive. Ong *et al.* have obtained faster convergence in MA using a trust-region framework for approximation and gradient based search [123, 124]; and using RBF surrogate model and gradient search [125]. However, it is imperative that the algorithms employing local search strategies should have explicit means to maintain diversity to avoid premature convergence.

One can combine the surrogate models in EA and local search to further reduce the number of actual function evaluations. Zhou *et al.* [94] have used a global Kriging model for the evolutionary search and a local RBF model for a local search.

3.6 Summary

A review on the use of approximation models in evolutionary algorithms is presented. Three types of surrogate models – RSM, RBF and Kriging are used in this study. There are many issues which need to be addressed when using surrogate models in EA. Some of the salient points highlighted in the literature review are as follows.

1. There is no single type of surrogate model suitable for all problems.

2. There are many well established sampling strategies for off-line training of surrogate models, but not many for on-line training.

3. Constructing a global surrogate model is quite difficult for nonlinear functions of many variables. A Kriging based surrogate model can be used as a global model, but it is limited in the number of training samples due to the high cost of training.

4. A local surrogate model is preferred to a global surrogate model. In the context of EA, a single local surrogate model is not sufficient since the region of the optimum solutions is not known a priori. Building multiple local models might require lot of samples for training.

5. Using multiple surrogates of different types can improve the prediction accuracy over a single type of surrogate model. When multiple surrogates are used in ensembles, aggregating multiple predictions can be an issue.

6. It is important to validate the surrogate model to ensure that the search is not misguided, but there is no way unique way of assessing the prediction accuracy.

7. In addition to replacing expensive evaluations, surrogate models can also be used in sampling and offspring creation to improve the quality of solutions.

8. Combining local search based on surrogate models can further improve the convergence of EA.

In the next chapter, a spatial surrogate modeling technique using multiple types of surrogate models is proposed. This approach tries to address the high-

lighted issues.

Chapter 4

Surrogate Assisted EA

4.1 Overview

For optimization problems involving computationally expensive simulations, the number of function evaluations one can afford is limited. Evolutionary algorithms (EAs) typically require large numbers of function evaluations to converge to the optimum. In an attempt to keep the computational cost of optimization affordable, surrogate models are used in lieu of the expensive analysis within EAs. Surrogate models are computationally cheaper alternatives to expensive simulations and approximate the responses of simulations. The use of surrogate models in EAs is complicated due to a number of factors. These include the choice of different types of surrogate models, sampling methods to generate the training data, global versus local modeling, determining the accuracy of the prediction, etc. Many approaches have been proposed in the literature to address one or more of these issues, but there is no single best method to solve all classes of optimization problems.

In this chapter, *spatial surrogate modeling*, a generic framework for surrogate

modeling within evolutionary algorithms, is proposed. Spatial surrogate models (also referred to simply as spatial surrogates) make use of multiple types of surrogate models for better approximation. Evolutionary algorithms generate candidate solutions as part of the evolutionary search, and these solutions are evaluated using the expensive simulations. These solutions are stored in an archive and used to train spatial surrogates. Spatial surrogates as the name suggests are located in the design space and their region of influence gets localized based on the evolved population over generations. The details of spatial surrogates are presented in Section 4.2. The solutions in the archive are used to build either a single spatial surrogate model; or they are partitioned into multiple clusters and a spatial surrogate model is built on each partition. Two algorithms are proposed using spatial surrogate models in Section 4.3 – surrogate assisted evaluation (SAE-EA) and surrogate assisted recombination (SAR-EA). The parameters required for spatial surrogates are discussed in Section 4.4. The performance of SAE-EA and SAR-EA are tested on a set of numerical test problems and the results are presented in Section 4.5. The summary of the findings is outlined in Section 4.6.

4.2 Spatial Surrogate Models

Spatial surrogate models are trained periodically during the course of an EA. The candidate solutions evaluated using the expensive simulations are stored in an archive and are used to train the surrogate models. These surrogate models are then used to evaluate the candidate solutions. Every few generations a population is evaluated using the expensive simulations. As the evolutionary search progresses, the population moves towards the region(s) where the better

solutions are located. Consequently, the archive gets more solutions from these regions. Progressively more and more solutions from these regions contribute to periodic retraining of spatial surrogates and the spatial surrogates get better prediction capability in these regions. As the population moves through the design space over generations, the regions where spatial surrogates predict better also move with the population. Thus, the spatial surrogates are "located" in specific regions of the design space. In case of multiple spatial surrogates, they are distributed in the design space.

4.2.1 The Archive of Solutions

In evolutionary algorithms, an offspring population of candidate solutions is generated from the parent population in each generation and the offspring population is evaluated using the actual analysis. The candidate solutions and their respective objective and constraint values are stored in an *archive*. Over the generations the archive size keeps on increasing. One of the uses of the archive is to re-use the objective and the constraint values if a solution that is already evaluated appears in the offspring generation. This prevents re-evaluation of the same candidate solution across generations. Only unique solutions are stored in the archive.

The main use of the archive is to train the spatial surrogate models. If two solutions used for surrogate training are very close to each other, numerical difficulties can arise in training due to ill-conditioning. To avoid such a pit-fall, a new solution which is very close to any other solution in the archive is not considered for surrogate training.

4.2.2 Surrogate Training

Spatial surrogates are trained using a fraction of the solutions in the archive to prevent over-fitting [52], and the remaining solutions are used for validation. The K-Means clustering [126] algorithm is used to identify the training solutions from the archive.

K-Means Clustering

The K-Means clustering algorithm partitions given solutions in K clusters such that the average Euclidean distance between the clusters is maximized and the average Euclidean distance within the cluster is minimized. The K-means clustering uses Lloyd's algorithm as described in Algorithm 4.1. The algorithm starts with a random assignment of solutions as K cluster centres (line 1). Each solution is assigned to a cluster with the closest centre (lines 3-8). After assignment the cluster centres are recomputed using the solutions assigned to each cluster (lines 9-13). The process of assignment and updating cluster centres is repeated till the assignment of solutions becomes fixed. Thus, K-Means clustering algorithm results in K cluster centres and assignment of solutions in each cluster.

Based on a predefined fraction value, the number of solutions (assumed K in this case) from the archive to be used for surrogate training is decided. Using K-Means clustering, the solutions in the archive are partitioned in K clusters. For each cluster, a solution closest to the cluster centre is picked and the training set of K solutions is formed.

Training and Validation

Once the training solutions are identified, the surrogate models are built for each of the objective and the constraint functions. For each function, multiple types

Algorithm 4.1 k-Means Clustering Algorithm

Require: $\{\mathbf{x}^1, \ldots, \mathbf{x}^n\}$ $\{n$ data points$\}$
Require: $k > 0$
1: $C = \{c_1, \ldots, c_k\} = \{\mathbf{x}^1, \ldots, \mathbf{x}^k\}$
2: **repeat**
3: **for** $i = 1$ to n **do**
4: $\psi(c_j|\mathbf{x}^i) = 0, \quad j = 1, \ldots, k$
5: $l = \arg\min_j \|x_i - c_j\|^2$
6: $\psi(c_l|\mathbf{x}^i) = 1$
7: $w(\mathbf{x}^i) = 1$
8: **end for**
9: **for** $i = 1$ to k **do**
10: $A = \sum_{j=1}^{m} \psi(c_i|\mathbf{x}^j)\, w(x_j)\, \mathbf{x}^j$
11: $B = \sum_{j=1}^{m} \psi(c_i|\mathbf{x}^j)\, w(x_j)$
12: $c_i = A/B$
13: **end for**
14: **until** ψ is constant

of surrogate models (e.g. RSM, RBF, Kriging) are constructed. The remaining solutions in the archive are then used as a validation set of solutions to assess the accuracy of each type of surrogate model. The best surrogate model is the one with the smallest prediction error on the validation set. The process is repeated for all the objectives and the constraints. These surrogate models are then used in lieu of the simulations to evaluate new candidate solutions.

If the best type of surrogate model selected has a large prediction error, a search based on a such a model is likely to be misguided and hence an additional surrogate validity check is introduced. The best surrogate model (the best among the various types of surrogate models) is considered valid only if the prediction error on the validation set is less than a user defined threshold. The prediction error used is the root mean squared error (RMSE) normalized by the range of function values as given in Equation 4.1, where y_i is the actual response and $\hat{y}_i$ is the predicted response. A prediction error threshold of 0.05 would correspond

to 5 percent error in the prediction on the validation set.

$$\text{normal RMSE} = \frac{1}{\max_i(y) - \min_i(y)} \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2} \tag{4.1}$$

In the first few generations of EA, the solutions in the archive are distributed throughout the design space and the spatial surrogate model trained using such points would be a global surrogate model. As EA converges, more training and validation solutions are available in the region of the optimum solutions. Then, the prediction error is contributed predominantly from the validation solutions in the region of the optimum as more number of validation solutions are from that region. The errors in the prediction on fewer solutions away from the region would get averaged by the total number of validation solutions. In this case, the spatial surrogate model would behave more like a local surrogate model. Thus the prediction accuracy of a spatial surrogate progressively increases in the region of optimality with generations of EA.

For a zero prediction error threshold, the surrogate model is forced to be accurate everywhere in the domain making it behave like a global surrogate model. It is often not possible to satisfy the zero prediction error criterion for nonlinear functions with high dimensionality using limited training data (which is the reason why global surrogate models are not preferred). Therefore, very small values of the prediction error threshold are avoided. When using a moderate value for the prediction error threshold (e.g. 0.01 to 0.1 corresponding to 1 to 10 percent error), the focus is on capturing the overall trend. As the number of training and validation samples in the region of optimum increases, the surrogate model can capture the local trends accurately. Higher prediction errors on few solutions outside the region of interest are not severely penalized due to the

averaging.

Only if the spatial surrogates for all the objectives and the constraints are valid, the spatial surrogates are used in lieu of the actual evaluation. If any of the surrogate models is invalid, then the actual simulation is used to evaluate new solutions and the results are added to the archive.

4.2.3 Prediction

The spatial surrogate models have a good prediction capability (within the prediction error threshold) in the region covered by the training data. Outside of this region, the prediction accuracy of the surrogate model cannot be ensured. To prevent any extrapolation using the surrogate model outside this region, it is essential to restrict the applicability of the surrogate model. Since the spatial surrogate model is trained using all the solutions in the archive, the potential region is the entire design space. As the EA converges, the spatial surrogates are accurate in a smaller region. In that case, the applicability of the surrogate model is restricted using a distance based criterion and identification of the neighborhood of the training data. The neighborhood is defined using the center of the training data (similar to the cluster centroid) and a radius of neighborhood. The radius is defined in terms of the size of the design space *i.e.* the percentage of the solid diagonal of the (normalized) design space. The radius of the neighborhood is a user defined value. If a new solution is within the neighborhood of the training data, the objectives and the constraint values are predicted using the surrogate model (provided the surrogate model is valid), otherwise actual analysis is used.

4.2.4 Single Surrogate (SS)

The steps involved in building a spatial surrogate are listed in Algorithm 4.2. The surrogate models are built using the archive ($\mathcal{A}$) of the solutions evaluated using the simulation. The solutions in the archive are split into a training set of solutions ($\mathcal{A}_{\text{train}}$) and a validation set ($\mathcal{A}_{\text{validate}}$) using K-Means clustering (line 1). The fraction of the solutions used as the training set are determined by a user defined input parameter α. For each response (objectives and constraints), multiple types of surrogate models are built and the prediction error is calculated (lines 3-6). The best surrogate model is the one with the least prediction error (line 7). The process is repeated for each function. The algorithm results in a collection of surrogate models ($\mathcal{S}$) and the overall error (E) for this collection is the maximum prediction error across the individual surrogate models.

Algorithm 4.2 Building a spatial surrogate

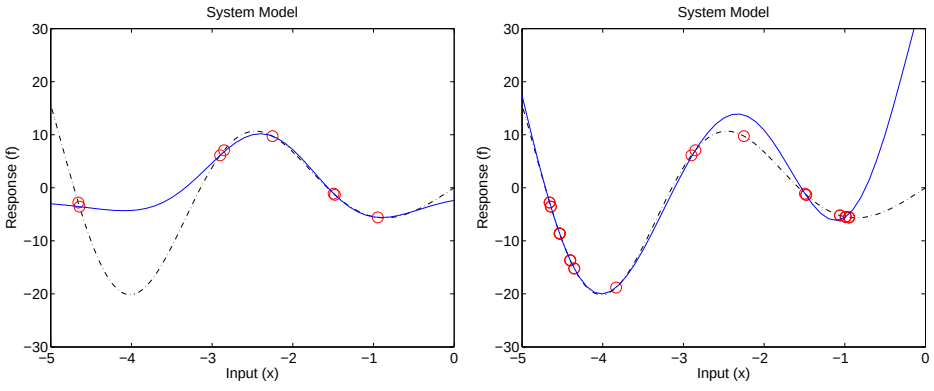
Require: $\mathcal{A}$ {Archive of true evaluations}
Require: $0 < \alpha < 1$ {Fraction of archive solutions used for training}
Require: $\mathcal{M}$ {Surrogate model types}
Require: $y_1, \ldots, y_n$ {n responses to be approximated}
1: $\mathcal{A}_{\text{train}}, \mathcal{A}_{\text{validate}} = \text{Split}(\mathcal{A}, \alpha)$
2: **for** $i = 1$ to n **do**
3: **for all** $M \in \mathcal{M}$ **do**
4: $\mathcal{S}_M = \text{SurrogateTrain(M, } \mathcal{A}_{\text{train}}, y_i)$
5: $e_M = \text{SurrogatePredict}(M, \mathcal{S}_M, \mathcal{A}_{\text{validate}}, y_i)$
6: **end for**
7: $M^* = \arg\min_M e_M$ {Best model for response y_i}
8: $\mathcal{S}_{y_i} = \mathcal{S}_{M^*}$
9: $e_{y_i} = e_{M^*}$
10: **end for**
11: $\mathcal{S} = \{\mathcal{S}_{y_1}, \ldots, \mathcal{S}_{y_n}\}$
12: $E = \max_i e_{y_i}$ {Surrogate accuracy}

The concept of a spatial surrogate is illustrated using a simple example. Consider the minimization of a function $f(x)$ of a single variable defined on

the interval [-5,0].

$$f(x) = \sin(x^2) \times (x^2 + 0.01x^3 + 5)$$

For a population size of eight, the initial population and the surrogate model built using RBF are plotted in Figure 4.1(a). The archive consists of eight solutions. The surrogate model is trained using six solutions (using 80% of solutions in the archive for training and the rest for validation) and remaining two solutions are used for validation. The surrogate model has very good accuracy in the range [-3,-1] as there are sufficient points in the region to capture the behavior. After three generations of EA, the archive consists of 14 solutions. The new solutions close to the solutions in the archive are not added to the archive. The surrogate model is retrained using eleven solutions (identified using K-Means clustering) and the remaining three solutions are used for validation. The resulting surrogate model is shown in Figure 4.1(b). The evolutionary search has found the region of the minimum and subsequently the surrogate model has very good prediction accuracy in the range [-5,-3]. Thus the surrogate model has migrated to the region of the minimum based on the evolutionary search.



(a) Initial Population (b) After three generations

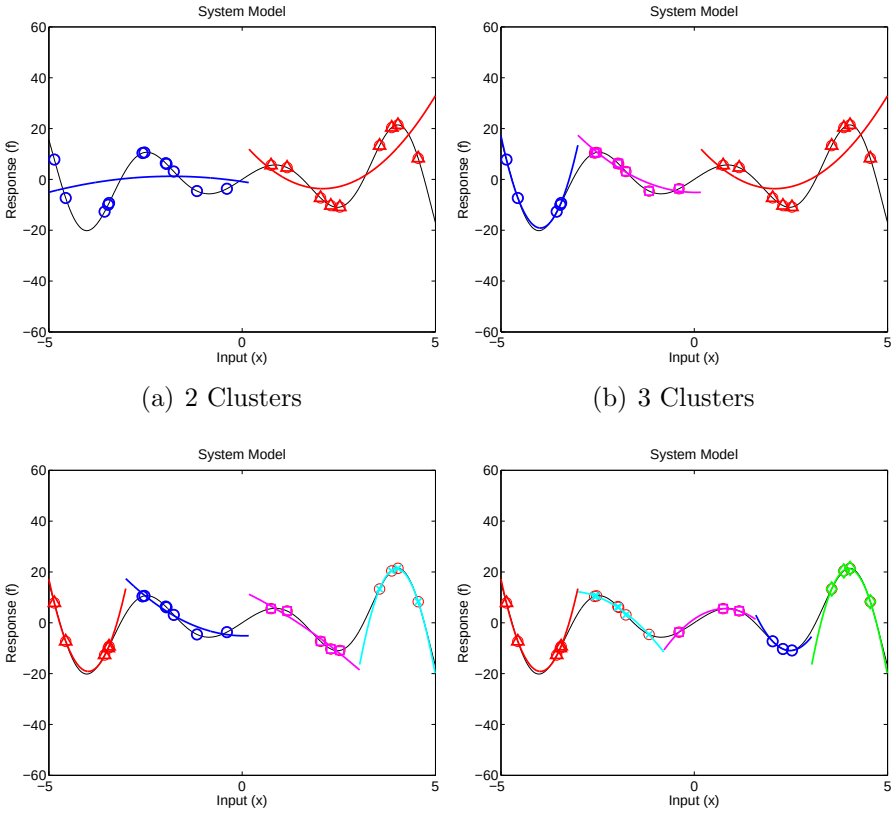
Figure 4.1: Spatial Surrogate model built using initial population and the archive after three generations

4.2.5 Multiple Surrogates (MS)

For highly nonlinear functions with multiple local minima, the evolutionary search can be divided into multiple regions and each region gets explored through the search. In such cases, a single surrogate model may not be adequate to capture the response in different regions of the design space. Each region can be approximated using a single surrogate model defined only on that region. Thus multiple surrogate models are required to be built. Another advantage in using multiple surrogates is that the nonlinear response is split into multiple regions and it can be modeled with simple surrogate models in each region. This is illustrated using a single variable function defined on the interval [-5,5] as shown in Figure 4.2. Using the points in the archive (represented by circles) two to five surrogate models are built. The points are partitioned using k-Means clustering and a RSM based surrogate model is created using the points in each partition. It is seen that by using four or five clusters, the entire function is very well approximated by simple quadratic models.

The training and validation of the multiple surrogate models are exactly the same as the single surrogate model. As shown in Algorithm 4.3, the solutions in the archive are partitioned into K partitions (line 1). A single spatial surrogate is built for each partition (line 3) using the Algorithm 4.2. In each partition, the solutions are divided into a training set and a validation set. The surrogate model on the partition is valid if it meets the prediction error threshold criterion, otherwise the surrogate model defined on that partition is considered invalid. It is possible that when using a large number of partitions, there may not be sufficient solutions in each partition to build a surrogate model (e.g. when using RSM). In that case no surrogate model is built for that partition.

To predict the response at a new point, the surrogate model built using the



(a) 2 Clusters (b) 3 Clusters

(c) 4 Clusters (d) 5 Clusters

Figure 4.2: Effect of the number of clusters on multiple spatially distributed surrogates

cluster closest to the new point is used. The clustering algorithm identifies a cluster centroid for each partition. For each cluster, the Euclidean distance between the new point and the centroid is calculated and the partition corresponding to the closest centroid is chosen. If the surrogate model corresponding to the closest centroid is invalid or not built, then the solution is evaluated using actual analysis instead. To prevent extrapolation using multiple surrogates, the neighborhood criterion is used as described in Section 4.2.3.

The main drawback of multiple surrogates is the requirement to specify the number of partitions. It is difficult to guess the appropriate number of partitions to be used in any problem. For some of the problems there can be significant

Algorithm 4.3 Building multiple spatial surrogates

Require: $\mathcal{A}$ {Archive of true evaluations}
Require: K {Number of partitions}
Require: ϵ {Prediction accuracy threshold}
1: $\mathcal{A}_1, \ldots, \mathcal{A}_K = \text{K-Means}(\mathcal{A}, K)$
2: **for** $i = 1$ to K **do**
3: $\mathcal{S}_i, E_i = \text{BuildSingleSurrogate}(\mathcal{A}_i)$
4: **if** $E_i < \epsilon$ **then**
5: $\mathcal{S}_i$ is valid
6: **else**
7: $\mathcal{S}_i$ is invalid
8: **end if**
9: **end for**

benefit in choosing the correct number of partitions. For an optimization problem with disjoint feasible regions, selecting the correct number of partitions can capture the function response in each disjoint feasible region.

4.2.6 Multiple Adaptive Surrogates (MAS)

Multiple adaptive surrogates use adaptive partitioning based on prediction accuracy to determine the number of partitions. The steps to find the number of partitions are outlined in Algorithm 4.4. The algorithm iteratively builds multiple spatial surrogates for partitions varying from one to K_{max}. For each number of partitions (K), Algorithm 4.3 is used to build multiple spatial surrogates (line 2). If the surrogate models on each partition are valid then the prediction error corresponding to the number of partitions (E_K) is defined as the sum of prediction errors on each partition. The number of partitions K that has the least corresponding prediction error E_K is chosen as the appropriate number of partitions.

The maximum number of partitions is determined based on the number of solutions in the archive N as given in Equation 4.2. The value is rounded to the

Algorithm 4.4 Determination of Number of Partitions for MAS

Require: $\mathcal{A}$ {Archive of the truly evaluation solutions}
Require: K_{max} {Upper limit on the number of partitions}
1: **for** $K = 1$ to K_{max} **do**
2: $\mathcal{S}_1, E_1, \ldots, \mathcal{S}_K, E_K = \text{BuildMultipleSurrogates}(\mathcal{A}, K)$
3: $\mathcal{S}_v = \{\mathcal{S}_i; \quad \mathcal{S}_i \text{ is valid}, i \in [1, K]\}$
4: **if** $|\mathcal{S}_v| == K$ **then**
5: $E_K = \sum_i E_i$
6: **end if**
7: **end for**
8: $K^* = \arg\min_K E_K, \quad K \in [1, K_{max}]$

nearest integer.

$$K_{max} = \left\lfloor \frac{\sqrt{N/5}}{2} \right\rceil \tag{4.2}$$

This corresponds to a maximum of two partitions for 100 points, five partitions for 500 points and seven partitions for 1000 points.

4.3 Spatial Surrogates in EA

Using the spatial surrogate modeling technique, two surrogate assisted evolutionary algorithms are proposed. In the first algorithm, *surrogate assisted evaluation*, spatial surrogates are used to replace the simulations with approximations in the evaluation of objective and constraint functions. In the second algorithm, *surrogate assisted recombination*, spatial surrogates are used in the evolution process to create an offspring population. Large numbers of solutions are generated during the evolution process and the solutions with best fitness as predicted by the spatial surrogates are selected as the offspring population.

4.3.1 Surrogate Assisted Evaluation

A framework for EA with surrogate assisted evaluation (SAE-EA) is shown in Algorithm 4.5. The surrogate models are periodically trained using the solutions in the archive and used in lieu of the actual analysis to evaluate the objectives and the constraints. The major steps of the SAE-EA framework are the same as the EA framework (Algorithm 2.1). The population in the first generation is initialized randomly and evaluated. The steps of evolution of the offspring population, evaluation of the offspring population and preservation of elite solutions to form next generation population are repeated for a prescribed number of generations. In SAE-EA, there are additional steps to update the archive of solutions evaluated using actual analysis, to train the surrogate models using the archive and to evaluate the offspring population using the surrogate models.

SAE-EA requires a parameter, the periodic train interval (I_{train}), that controls how often the surrogate models are trained and used (lines 6-8). For the first few generations no surrogate model is built and used as there are too few solutions in the archive. Only after the first I_{train} generations, the training of the surrogate model takes effect. Every I_{train} generations, the offspring population is evaluated using the actual analysis (line 11) and for the generations in between the trained surrogate models are used for evaluating the objectives and the constraints (line 13). During this phase, it is possible that the new solutions are considered better as per the surrogate evaluations and these replace the current best solutions. If the new solutions are actually not better than the ones they replaced, then the search will be misguided. To avoid promoting a worse solution, the best solutions evaluated using the actual analysis are always retained in the population. The number of solutions preserved across generations is decided by the retain count (N_{RC}). If a new solution is predicted to be better than any of the top ranking

Algorithm 4.5 Surrogate Assisted Evaluation in EA Framework

Require: $N_G > 1$ {Number of Generations}
Require: $I_{train} > 0$ {Periodic Surrogate Training Interval}
Require: $N_{RC} > 1$ {Number of truly evaluated solutions retained}

```
1: Initialize(P_1)
2: Evaluate(P_1)
3: Rank(P_1)
4: A ← P_1
5: for i = 2 to N_G do
6:    if i > I_train and modulo(i, I_train) == 0 then
7:       do_training = 1
8:    end if
9:    C_i = Evolve(P_{i-1})
10:   if do_training == 1 then
11:      Evaluate(C_i)
12:   else
13:      EvaluateSurrogate(C_i, S)
14:      Rank(C_i)
15:      for j = 1 to N_RC do
16:         if Soln_{rank=j}(C_i) is better than Soln_{rank=1}(P_{i-1}) then
17:            Evaluate(Soln_{rank=j}(C_i))
18:         end if
19:      end for
20:   end if
21:   A ← A ⋃ C_i
22:   Rank(P_{i-1} + C_i)
23:   P_i = Reduce(P_{i-1} + C_i)
24:   if do_training == 1 then
25:      S = BuildSurrogate(A)
26:   end if
27: end for
```

N_{RC} solutions, that solution is evaluated using the actual analysis and verified that it is indeed better (lines 16-18). Any new solutions evaluated using the actual analysis (either the whole population or only a few select solutions) are added to the archive (line 21). The surrogate models are trained every I_{train} generations using the solutions in the archive (line 25).

A single spatial surrogate model, multiple spatial surrogate models or multiple

adaptive spatial surrogate models can be used in this framework. The periodic surrogate training interval is typically small, e.g. 3–5, to ensure that the evolutionary search does not migrate the entire population to the region of local optimum of the spatial surrogate. The primary focus is on the exploration of the design space rather than the exploitation. The focus can be shifted to exploitation by increasing the periodic training interval and letting the population converge to the regions of optimum solutions of the surrogate models. This idea is explored with the proposal of a surrogate assisted recombination algorithm.

4.3.2 Surrogate Assisted Recombination

The framework for surrogate assisted recombination in EA (SAR-EA) is presented in Algorithm 4.6. The main difference between SAE-EA and SAR-EA is the use of spatial surrogates for evaluation versus evolution. In SAR-EA, the solutions are always evaluated using the actual analysis and not the surrogate models. Also, the surrogate models are trained in every generation of SAR-EA and used for recombination on the similar lines as informed operators.

Algorithm 4.6 Surrogate Assisted Recombination in EA Framework

Require: $N_G > 1$ {Number of Generations}
1: Initialize(P_1)
2: Evaluate(P_1)
3: Rank(P_1)
4: $\mathcal{A} \leftarrow P_1$
5: **for** $i = 2$ to N_G **do**
6: $\mathcal{S}$ = BuildSurrogate($\mathcal{A}$)
7: C_i = EvolveSurrogate($P_{i-1}, \mathcal{S}$)
8: Evaluate(C_i)
9: $\mathcal{A} \leftarrow \mathcal{A} \bigcup C_{i-1}$
10: Rank($P_{i-1} + C_i$)
11: P_i = Reduce($P_{i-1} + C_i$)
12: **end for**

The initial steps of SAR-EA are the exactly same as that of SAE-EA (lines 1-4). The next few steps (lines 6-11) are repeated for a prescribed number of generations. These include training of the spatial surrogate models (line 6) and evolving an offspring population based on these surrogate models (line 7). All solutions in the offspring population are evaluated using the actual analysis (line 8) and are added to the archive (line 9).

The evolution process of SAR-EA uses embedded evolutionary algorithm (referred to as Sub-EA). In SAE-EA, the evolution process uses crossover and mutation operations to create an offspring population. However, in SAR-EA, the evolution process creates an offspring population, evaluates using the spatial surrogate models, and evolves them further using crossover and mutation. Conceptually SAR-EA is similar to SAE-EA. A Sub-EA embedded in SAR-EA does the evolutionary search similar to the generations in SAE-EA between two surrogate training cycles, where the surrogate models are used to evaluate the fitness of the solutions in lieu of the actual analysis.

Recombination using Sub-EA

In the Sub-EA evolution mechanism, the population of SAR-EA is allowed to evolve using an embedded evolutionary algorithm with conventional crossover and mutation operators as described in Section 2.5. The population size for sub-EA can be larger than the population size in SAR-EA for better exploration. The initial population of sub-EA is seeded from the parent population of SAR-EA. Within a single generation of SAR-EA, Sub-EA population is evolved over a number of generations. In the sub-EA, the objectives and the constraints are evaluated using the spatial surrogate models only. No actual evaluations are used in this process. The best solutions in the final population of Sub-EA are

taken as the offspring population in SAR-EA.

If any of the surrogate models are not valid (*i.e.* the prediction error over the validation set is larger than the prescribed error threshold), the population is evolved using the crossover and the mutation as in SAE-EA and no sub-EA is used. When the surrogate models are valid, the sub-EA population converges to the regions of optima of the surrogate models, thus exploiting the spatial surrogate models to achieve faster convergence.

4.4 Parameter Choice

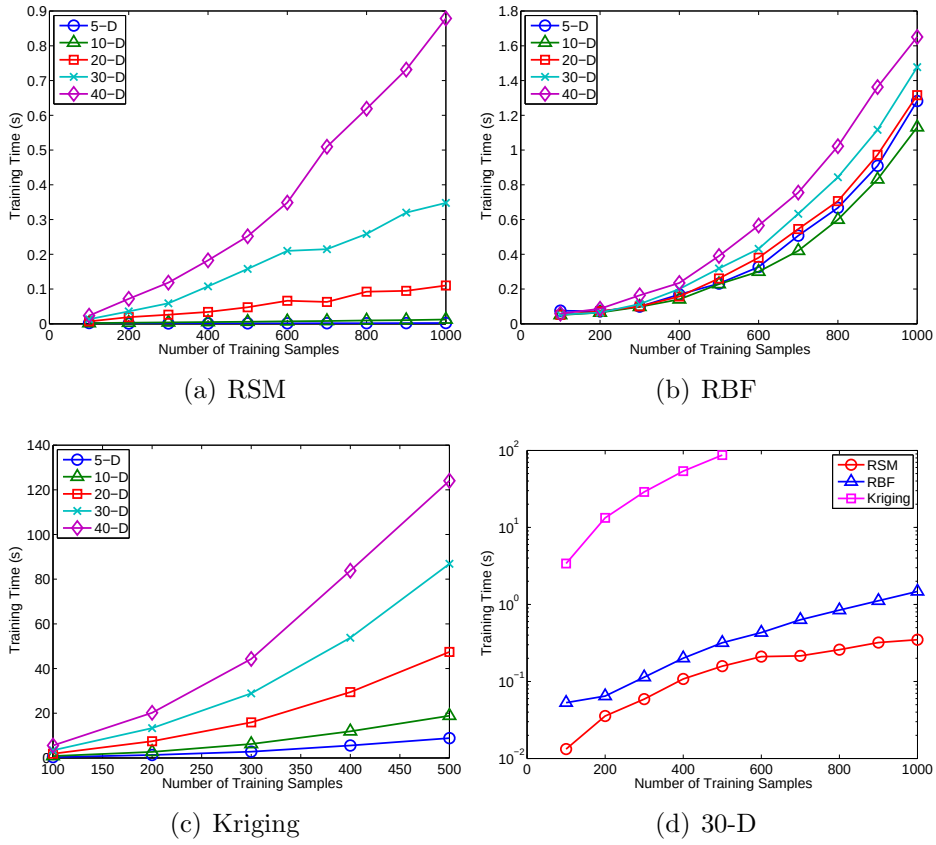
The proposed algorithms SAE-EA and SAR-EA require additional parameters for spatial surrogate modeling. These include types of surrogate models, maximum number of training samples and prediction error threshold.

4.4.1 Choice of Surrogate Models

The training process itself can be computationally expensive for certain types of surrogate models. A comparison of the training times for RSM, RBF, and Kriging are shown in Figure 4.3. The surrogate models are trained using five scalable functions – quadratic model, quartic model, generalized Schwefel function, generalized Rosenbrock function and generalized Rastrigin function. (These functions are described in Appendix A as f01, f07, f08, f05, and f09 respectively). For each test function, design points are randomly sampled from the range [0,1] and evaluated. The number of design variables used are 5, 10, 20, 30, and 40. The number of training designs are varied from 100 to 1000 in steps of 100. The surrogate models are implemented using Matlab R2008a.[1] The surrogate training

[1]RSM is implemented using linear algebra functions. For RBF and MLP, neural network toolbox functions newrbe and newff/train are used respectively. The Kriging toolbox for Matlab

is repeated 10 times for each function and the average of 50 training cycles are used for comparison.



(a) RSM (b) RBF

(c) Kriging (d) 30-D

Figure 4.3: Surrogate training time (in seconds) for RSM, RBF and Kriging.

For RSM, the training time is proportional to the number of dimensions and the number training designs. For 40 design variables, the average training time for RSM is 0.6 seconds for 1000 training samples (Figure 4.3(a)). The training time for RBF is dependent only on the number of training designs as shown in Figure 4.3(b). RBF training using 1000 points for 40 design variables requires around 1 second. The number of training designs are limited to 500 for Kriging due to higher training time. Kriging is computationally the most expensive model

written by Hans Nielsen [127] is used for Kriging.

of the three types. It takes more than 120 seconds to train a Kriging model using 500 designs with 40 design variables. The training cost for Kriging is directly proportional to the number of training designs and the number of design variables (Figure 4.3(c)).

The benchmark test problems studied in this thesis have up to 30 design variables, and 500 training solutions should be adequate to build the three types of surrogate models. (RSM requires at least 496 points to train 30-D data.) The comparison of the surrogate training cost for the three surrogate model types to train 30-D data is shown in Figure 4.3(d) (The y-axis is in log scale). For 500 training solutions, the training costs of RSM and RBF are around 0.1 and 0.2 seconds, whereas for Kriging it is around 83 seconds. To keep the training cost low, only RSM and RBF are used for numerical test problems.

4.4.2 Maximum Number of Training Samples

As seen from Figure 4.3, the training cost of surrogate models increases with an increase in the number of training samples. In the proposed algorithms, the number of solutions in the archive (*i.e.* the solutions evaluated using simulations) gradually increase over the generations and so does the cost of surrogate training. The maximum number of training samples are specified to limit the surrogate training cost in the proposed algorithms. If there are more solutions in the archive, then the latest additions to the archive are considered. This ensures that the surrogate models are built on the interesting regions identified by the evolutionary search.

A single spatial surrogate with no limit on the maximum number of training samples, is almost equivalent to a global surrogate model. The difference is that not all the solutions in the archive are used, but only a fraction. Even

then the solutions would be spread throughout the design space and the single surrogate would be limited similarly as a global surrogate. On the other hand, a single surrogate with smaller limit on the training samples would focus the surrogate modeling effort solely to the region currently being explored by the evolutionary search. The surrogate model is then equivalent to a local surrogate model. Although a local surrogate model is preferred over a global model, the evolutionary search can get stuck in a region of a local optimum and can never get out. A moderate value for the maximum number of training samples will ensure that all the solutions used for training are not restricted to a small region and the evolutionary search can migrate to other regions if the current best solution turns out to be a local minimum but not the global minimum.

4.4.3 Prediction Error Threshold

As described in Section 4.2.2, the prediction error threshold controls the accuracy of the surrogate models. For a zero prediction error threshold, the predictions on the validation set should exactly match the computed values of objectives and constraints. This creates an interpolating fit for the validation data. A non-zero value for prediction error threshold, on the other hand, would create a smoothing fit as the predictions on the validation set need not match the computed values. Although the smoothing fit is inaccurate for the solutions in the validation set, the error in prediction is bounded by the error threshold. Also, an interpolating fit can require a large number of training samples to capture the nonlinear response, whereas a smoothing fit can do with a comparatively fewer number of training samples.

4.5 Results of Numerical Benchmarks

The performance of the proposed algorithms is studied for unconstrained and constrained, single and multi-objective optimization test problems. For single objective optimization, unconstrained f-series problems (refer to Appendix A for details) and constrained g-series problems (refer to Appendix B for details) are used. Multi-objective optimization problems used are ZDT (refer to Appendix C for details) and CTP (refer to Appendix D for details). All the problems are solved as minimization problems using EA, SAE-EA and SAR-EA. For SAE-EA, single surrogate and multiple adaptive surrogate models are used and are referred to as SAE-SS and SAE-MAS respectively.

4.5.1 Experimental Setup

For single objective optimization problems the number of actual evaluations is limited to 1,000 and the population size used is 40. This corresponds to 25 generations for EA and SAR-EA, whereas SAE-EA is run till the number of actual evaluations reaches 1,000. It is possible to pick another combination of population size and generations that result in 1,000 evaluations. A large population would be limited in the number of generations over it can be evolved. On the other hand a small population would lack sufficient diversity to effectively search the design space. For multi-objective optimization problems the number of actual evaluations is fixed at 5,000 and population size is 100. The corresponding generations for EA and SAR-EA are 50, and SAE-EA is run till the number of actual evaluations reaches 5,000.

Multiple runs are performed for each problem by varying the parameters. The parameters required for EA are crossover probability, mutation probability,

crossover distribution index, mutation distribution index and random seed. The values used for each parameter are listed in Table 4.1. With two values for each of the first four parameters and four values for the random seed, the total number of combinations is $(2^4 \times 4 =)$ 64. Thus, each problem is solved exhaustively for each of the 64 combinations.

Table 4.1: Parameters for evolutionary algorithm

Parameter	Values
Crossover probability	0.8, 0.9
Mutation probability	0.05, 0.1
Distribution index for crossover	10, 15
Distribution index for mutation	10, 20
Random seed	10, 20, 30, 40

The parameters required for spatial surrogate models are *training data fraction*, *maximum number of training samples* and *prediction accuracy threshold*. The fraction of archive solutions used for training surrogate models is set to 0.8 (corresponding to 80% of the solutions for training and the remaining for validation) and the maximum number of training samples is fixed at 500. The training time for RSM and RBF using 500 training samples is less than a second, so RSM and RBF are used in the spatial surrogates. In addition, SAE-EA uses two more parameters, namely *periodic training interval* and *retain count*. The values used for all surrogate parameters are listed in Table 4.2. The retain count parameter is only varied for multi-objective optimization problems. For single objective optimization, the retain count parameter is set to 1. With these parameters, SAE-EA is run for $(64 \times 2^2 =)$ 256 separate parameter combinations for single objective optimization problems and 512 combinations for multi-objective optimization problems. The only parameter for SAR-EA is the

prediction accuracy threshold for spatial surrogates and the number of parameter combinations are 128. The population size used for Sub-EA in SAR-EA is 80, twice the size of the original population. The Sub-EA is evolved over 100 generations.

Table 4.2: Parameters for for SAE-EA and SAR-EA

Parameter	Values
Training data fraction	0.8
Maximum number of training samples	500
Prediction accuracy threshold	0.02, 0.05
Periodic training interval	3, 5
Retain count[2]	10, 20

The performance of single objective optimization problems is assessed using the best, the average and the worst objective values across multiple runs. For the multi-objective optimization, the non-dominated solutions are compared using the displacement metric. The minimum, the average and the maximum values of the displacement metric are reported for multiple algorithms.

4.5.2 Unconstrained Single Objective Optimization

Before comparing the performance of EA and surrogate assisted EA, it is important to verify that valid surrogate models are actually built and used. Shown in Table 4.3 are the statistics on the number of experiments in which valid surrogate models were trained. For optimization problems f04, f08 and f09 spatial surrogate models could not be built using RSM or RBF with any surrogate parameter combinations. The problems f04 and f08 have functions that use the absolute

[2]Retain count is only used for multi-objective optimization problems. For single objective optimization problems it is set to 1.

values of the design variables and the problem f09 uses a generalized Rastrigin function. As the spatial surrogate models are not used for these problems, the results of SAE-EA and SAR-EA are identical to those of EA and are omitted. For problems f02, f10 and f12 the number of runs in which the surrogate models are built, is less than half the total number of runs. For these problems, spatial surrogate models could not be built for a prediction error threshold value of 0.02.

Table 4.3: Number of runs in which surrogate models were built using SAE-EA and SAR-EA for f-series problems

Problem	SAE-SS (256)	SAE-MAS (256)	SAR (128)
f01	256	256	128
f02	82	75	58
f03	256	256	128
f04	0	0	2
f05	136	147	87
f06	256	256	128
f07	145	140	88
f08	0	0	0
f09	0	0	0
f10	70	76	58
f11	256	256	128
f12	114	116	74
f13	130	129	125

The best objective values obtained using EA, SAE-EA and SAR-EA on the remaining f-series problems are listed in Table 4.4. The following observations can be made:

- The best objective values obtained using SAE-EA with single surrogate (SAE-SS) are consistently better than those obtained with EA for all the problems. For problems f01, f03, f05, f06, f12 and f13 the objective values are better by orders of magnitude.

- The best objective values obtained using SAE-EA with multiple adaptive surrogates (SAE-MAS) are better than those obtained by EA on all the problems. Compared to the best objective values obtained using single surrogate, however, those obtained by multiple adaptive surrogates are generally higher but of the same order of magnitude.

- The optimum solutions (within the tolerance of 1.e-6) are obtained using SAR-EA for problems f01, f03, f06, f07 and f11. For f12 and f13 the objective values are multiple orders of magnitude better than SAE-EA with single and multiple adaptive surrogates.

Table 4.4: Best objective values for f-series problems using EA, SAE-EA and SAR-EA

Problem	Optimum	EA	SAE-SS	SAE-MAS	SAR
f01	0.0	645.17	5.68	9.0	0.0
f02	0.0	6.21	5.78	6.13	5.46
f03	0.0	5261.63	827.79	1108.27	0.0
f05	0.0	185849.0	49795.5	69537.3	4226.43
f06	0.0	1029.0	11.0	5.0	0.0
f07	0.0	0.08	0.04	0.05	0.0
f10	0.0	7.1	6.09	6.47	3.64
f11	0.0	6.81	1.07	1.08	0.0
f12	0.0	148.77	15.57	33.03	2.17
f13	0.0	158180.0	2965.05	3618.83	9.44

The average objective values for f-series problems using EA, SAE-EA and SAR-EA are listed in Table 4.5. The average performance of SAE-EA using single surrogate and multiple adaptive surrogates is better than EA for problems f01, f03, f05, f06, f11, f12, f13 and on par with EA for problems f02, f07, f10. The performance of multiple adaptive surrogates is similar to single surrogates as only a single partition is used in multiple adaptive surrogates for building surrogate

models. With accurate spatial surrogate models, SAR-EA can converge to the optimum value quickly as seen for problems f01 and f03 and near optimum for problems f06 and f11. However, the premature exploitation of the surrogate models can misguide SAR-EA and the average performance can be poorer than EA as seen for problem f12.

Table 4.5: Average objective values for f-series problems using EA, SAE-EA and SAR-EA

Problem	Optimum	EA	SAE-SS	SAE-MAS	SAR
f01	0.0	2208.51	53.57	51.13	0.00
f02	0.0	11.94	12.68	13.35	14.76
f03	0.0	15086.28	5182.19	5142.67	0.00
f05	0.0	1564560.33	920083.25	1062907.84	1036372.43
f06	0.0	2113.5	62.66	55.59	6.45
f07	0.0	0.41	0.35	0.35	0.58
f10	0.0	9.81	9.6	9.44	9.4
f11	0.0	20.54	1.58	1.52	0.13
f12	0.0	508433.03	308345.4	286165.1	897432.9
f13	0.0	1864012.2	1436457.1	1596279.2	4784331

Shown in Table 4.6 are the worst objective values obtained using all the algorithms. The objective values obtained using single surrogate and multiple adaptive surrogates in SAE-EA are much lower than those obtained by EA for problems f01, f03, f05, f06, f11, and f12. The performance is similar for the rest of the problems. The advantages of exploitation of spatial surrogates in SAR-EA are evident from the objective values close to the optimum obtained for problems f01, f03, f06, and f11. At the same time poorer performance due to premature convergence is evident for large objective values for problems f02, f05 and f12.

Average convergence trends for f-series problems are shown in Figure 4.4 and Figure 4.5. The performances of SAE-EA and SAR-EA in the initial generations

Table 4.6: Worst objective values for f-series problems using EA, SAE-EA and SAR-EA

Problem	Optimum	EA	SAE-SS	SAE-MAS	SAR
f01	0.0	4370.97	462.31	199.76	0.0
f02	0.0	21.5	37.58	28.55	46.37
f03	0.0	24072.1	14877.5	11383.1	0.0
f05	0.0	4956490	4201190	4770500	15190300
f06	0.0	3933.0	295.0	330.0	164.0
f07	0.0	1.5	2.03	1.21	6.34
f10	0.0	12.33	12.97	12.24	13.44
f11	0.0	40.34	3.91	3.4	1.05
f12	0.0	7084780	4586400	2953640	22462200
f13	0.0	11035000	10667700	15086300	74036700

(for lower values of function evaluations) are poorer than that of EA as the numbers of solutions in the archive are not sufficient to build valid spatial surrogates. Once the spatial surrogates are built, the convergence improves for both SAE-EA and SAR-EA and is faster than that of EA.

4.5.3 Constrained Single Objective Optimization

For g-series problems, the numbers of runs in which valid surrogate models are built and used are shown in Table 4.7. For problem g02, the spatial surrogates are built only for a single parameter combination. For problem g08, the number of runs with valid surrogate models are very few indicating difficulties in approximating the function response. Problem g02 has an objective function with a product of all the design variables and the objective function in problem g08 has a product of trigonometric terms. The results of SAE-EA and SAR-EA for problems g02 and g08 are primarily obtained from the evolutionary search and not using spatial surrogates, hence they are omitted. For the rest of the problems valid surrogate models were built in all the runs with the exception of problem

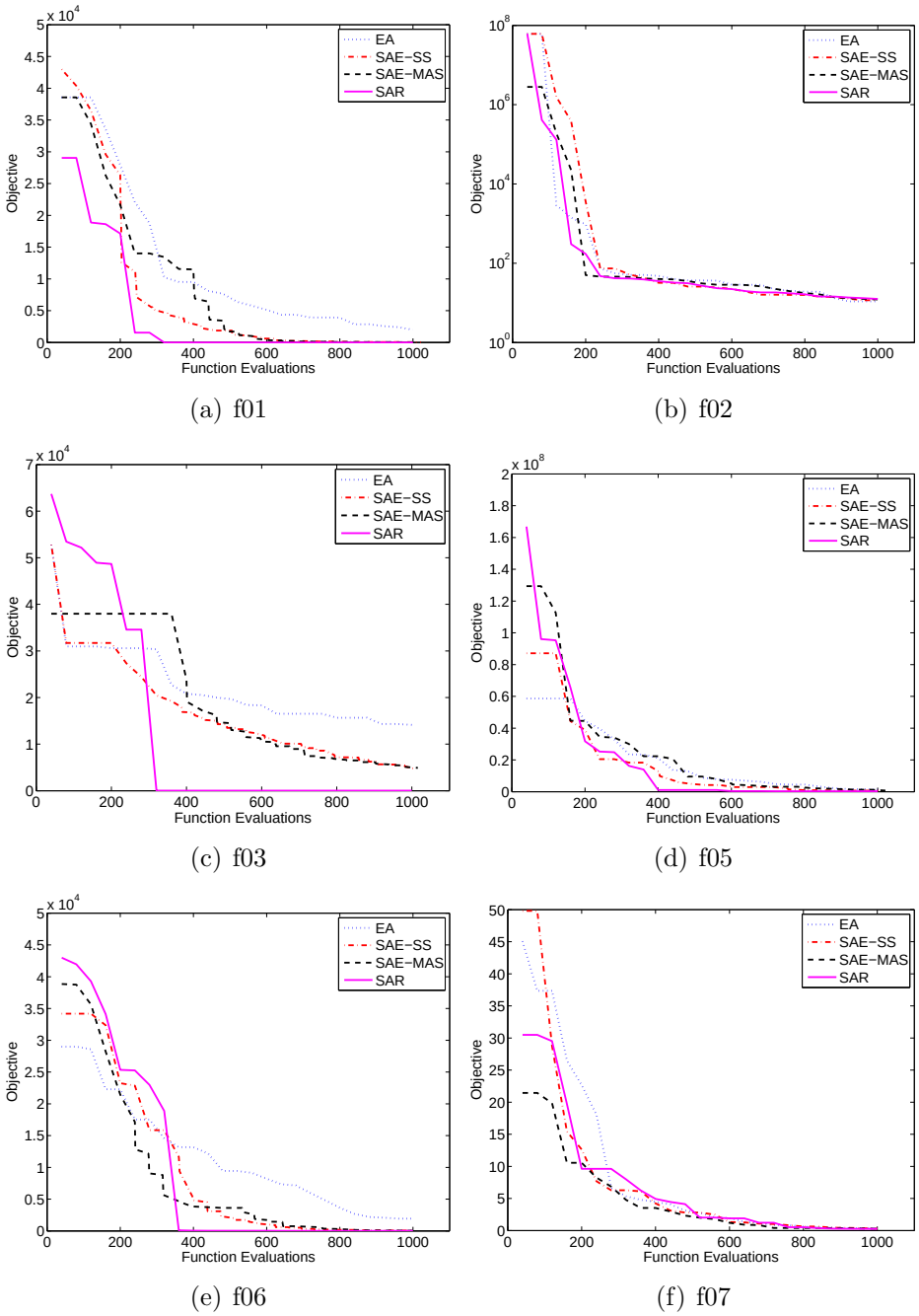
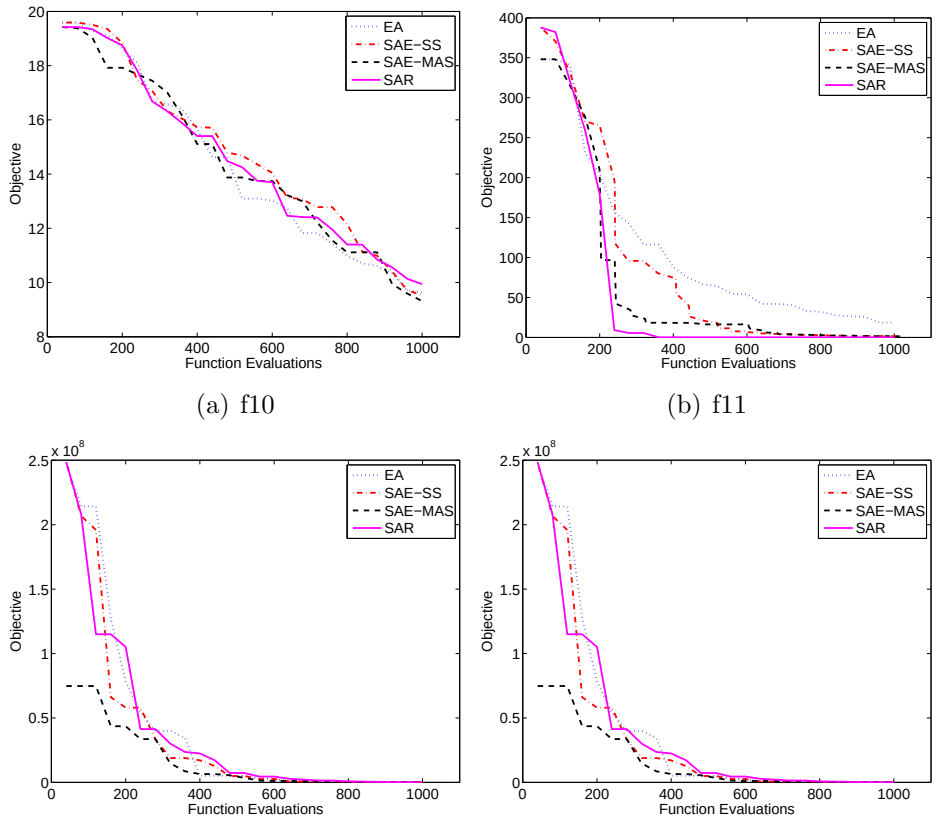


Figure 4.4: Average convergence trend of EA, SAE-EA and SAR-EA for f-series problems (f01–f03 and f05–f07)

g09. For problem g09, valid spatial surrogate models were built in 100 runs of SAE-EA with the prediction error threshold value of 0.05, and 10 runs with the



(a) f10 (b) f11

(c) f12 (d) f13

Figure 4.5: Average convergence trends of EA, SAE-EA and SAR-EA for f-series problems (f10–f13)

prediction error threshold of 0.02.

The best, the average and the worst objective values obtained for the remaining g-series problems using EA, SAE-EA and SAR-EA are listed in Table 4.8, Table 4.9 and Table 4.10 respectively. The best objectives values obtained by SAE-EA and SAR-EA are much closer to the optimum values than those obtained by EA. The exception is problem g06, where the objectives values obtained using SAE-EA are higher than that of EA. For the first three problems (g01, g04 and g06), the known optimum solutions are obtained using SAR-EA. For the other problems the objective values reported are very close to the known optimum.

Table 4.7: Number of runs in which surrogate models were built using SAE-EA and SAR-EA for g-series problems

Problem	SAE-SS (256)	SAE-MAS (256)	SAR (128)
g01	256	256	128
g02	1	1	0
g04	256	256	128
g06	256	256	128
g07	256	256	128
g08	16	18	18
g09	110	109	91
g10	256	256	128

Table 4.8: Best objective values for g-series problems using EA, SAE-EA and SAR-EA

Problem	Optimum	EA	SAE SS	SAE MAS	SAR
g01	-15	-10.46	-14.87	-14.82	-15
g04	-30665.539	-30385.4	-30631.2	-30593.6	-30665.5
g06	-6961.81	-6835.64	-6788.16	-6880.6	-6961.81
g07	24.306	74.45	25.91	25.77	24.33
g09	680.63	702.4	690.03	693.06	704.59
g10	7049.33	8536.85	7489.95	7429.35	7056.06

Similar trends are observed for the average objective values obtained by SAE-EA and SAR-EA (see Table 4.9). The average performance of SAE-EA using single and multiple adaptive surrogates is better than EA for problems g01, g04, g06, g07 and g10 and similar for problem g09. Using SAR-EA, the population converges to the region of the optimum very quickly for problems g01, g04, g06, g07 and g10. For problem g09, the average values across all the algorithms are similar and indicate convergence to a region of a local minimum. The errors in the prediction of the constraint values can identify an infeasible solution as a feasible solution and the search can get misguided. In addition,

both RSM and RBF models have difficulty in approximating the responses for problem g09 as seen in Table 4.7.

Table 4.9: Average objective values for g-series problem using EA, SAE-EA and SAR-EA

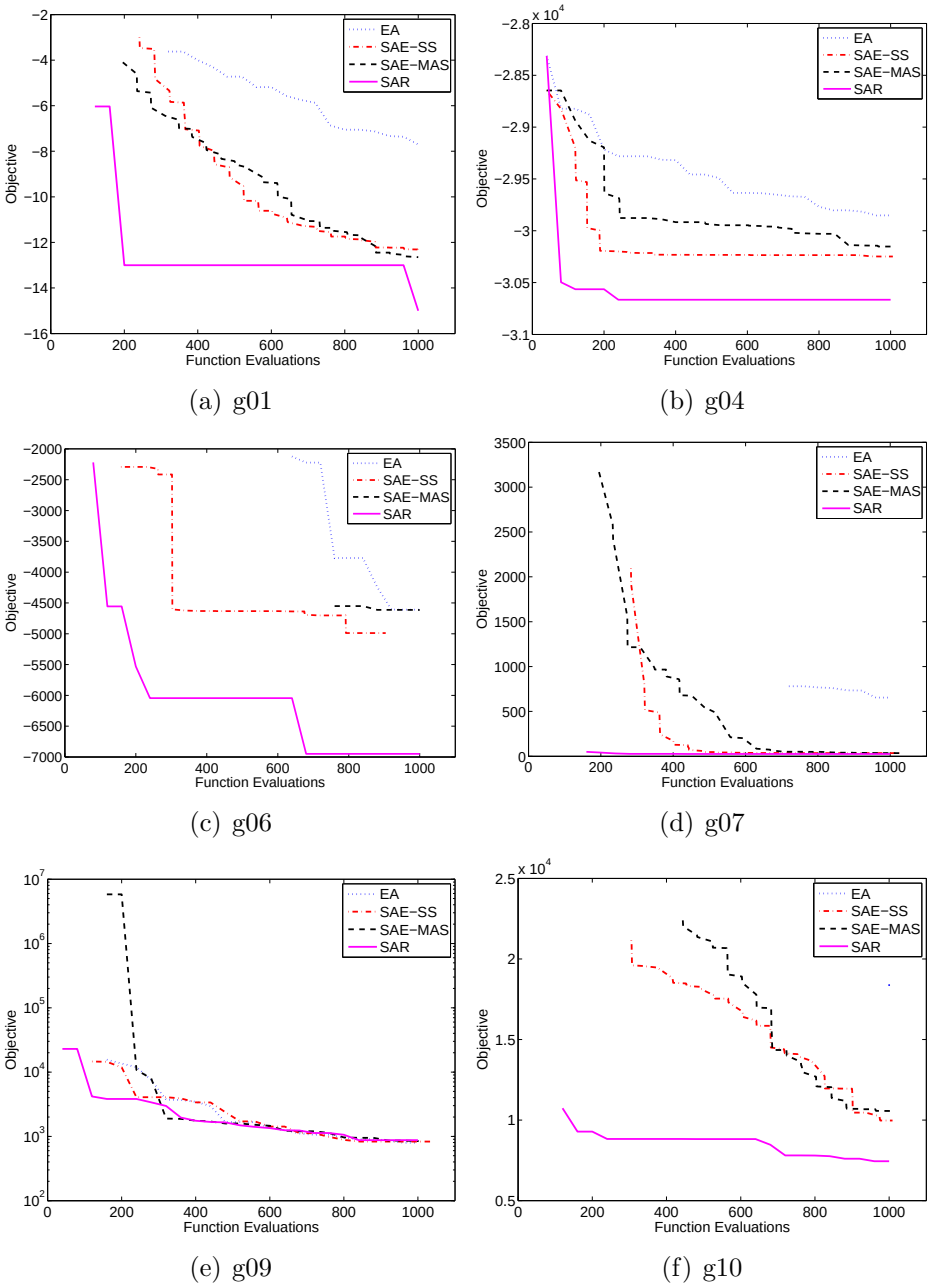
Problem	Optimum	EA	SAE-SS	SAE-MAS	SAR
g01	-15	-7.38	-12.21	-12.4	-14.97
g04	-30665.539	-29872.83	-30212.34	-30133.43	-30665.26
g06	-6961.81	-4217.64	-4576.55	-4215.13	-6782.17
g07	24.306	868.65	54.64	69.48	26.67
g09	680.63	858.09	956.78	955.84	998.19
g10	7049.33	18313.6	10772.69	11166.16	7507.48

Table 4.10: Worst objective values for g-series problems using EA, SAE-EA and SAR-EA

Problem	Optimum	EA	SAE-SS	SAE-MAS	SAR
g01	-15	-3.75	-7.42	-7.56	-13
g04	-30665.539	-29176.3	-29676.5	-29341.4	-30650.3
g06	-6961.81	-1240.13	-1229.72	-1214.77	-5748.64
g07	24.306	2733.68	815.41	1483.27	31.96
g09	680.63	1474.49	4266.72	2895.92	3172.63
g10	7049.33	25036	20665.1	25531.4	8567.12

Comparing the best and the worst objective values obtained for g-series problems (see Table 4.8 and Table 4.10), it can be seen that the differences are much larger for EA as compared to SAE-EA and SAR-EA. This indicates that the performances of SAE-EA and SAR-EA are more consistent than that of EA. The average convergence trends for g-series problems are depicted in Figure 4.6. It can be seen from Figure 4.6, that SAE-EA and SAR-EA converge much quicker than EA for all the g-series problems except problem g09. Also it can be observed that the feasible solutions are found much faster using SAE-EA and SAR-EA than EA

for problems g01, g06, g07 and g10.



(a) g01 (b) g04

(c) g06 (d) g07

(e) g09 (f) g10

Figure 4.6: Average convergence trends of EA, SAE-EA and SAR-EA for g-series problems

4.5.4 Unconstrained Multi-objective Optimization

Out of the five ZDT problems, spatial surrogate models using RSM and RBF were built for ZDT1, ZDT2 and ZDT3 and the corresponding results for only those problems are reported. For ZDT1 and ZDT2, the surrogate models are built in all the runs whereas for ZDT3 the surrogate models are built in roughly half the number of runs. These runs correspond to the prediction error threshold of 0.05.

The minimum values of the displacement metric using EA, SAE-EA and SAR-EA are shown in Table 4.11. The displacement values for SAE-EA and SAR-EA are very close to zero for problems ZDT1 and ZDT2, indicating that the non-dominated solutions are very close to the corresponding Pareto optimal fronts. For ZDT3, the results of SAE-EA and SAR-EA are slightly better than that of EA.

Table 4.11: The best displacement metric values for ZDT problems using EA, SAE-EA and SAR-EA

Problem	EA	SAE-SS	SAE-MAS	SAR
ZDT1	0.0030	0.0007	0.0007	0.0006
ZDT2	0.0063	0.0003	0.0004	0.0001
ZDT3	0.0022	0.0017	0.0021	0.0018

The average values of displacement metric obtained using EA, SAE-EA and SAR-EA are shown in Table 4.12. The performance using spatial surrogates is definitely better for problems ZDT1 and ZDT2. A similar trend is seen for the worst values of the displacement metric as listed in Table 4.13.

The non-dominated solutions obtained for problems ZDT1, ZDT2 and ZDT3 using all the algorithms are shown in Figure 4.7. The non-dominated solutions

Table 4.12: The average displacement metric values for ZDT problems using EA, SAE-EA and SAR-EA

Problem	EA	SAE-SS	SAE-MAS	SAR
ZDT1	0.0044	0.0018	0.0019	0.0043
ZDT2	0.0288	0.0020	0.0021	0.0003
ZDT3	0.0046	0.0066	0.0058	0.0048

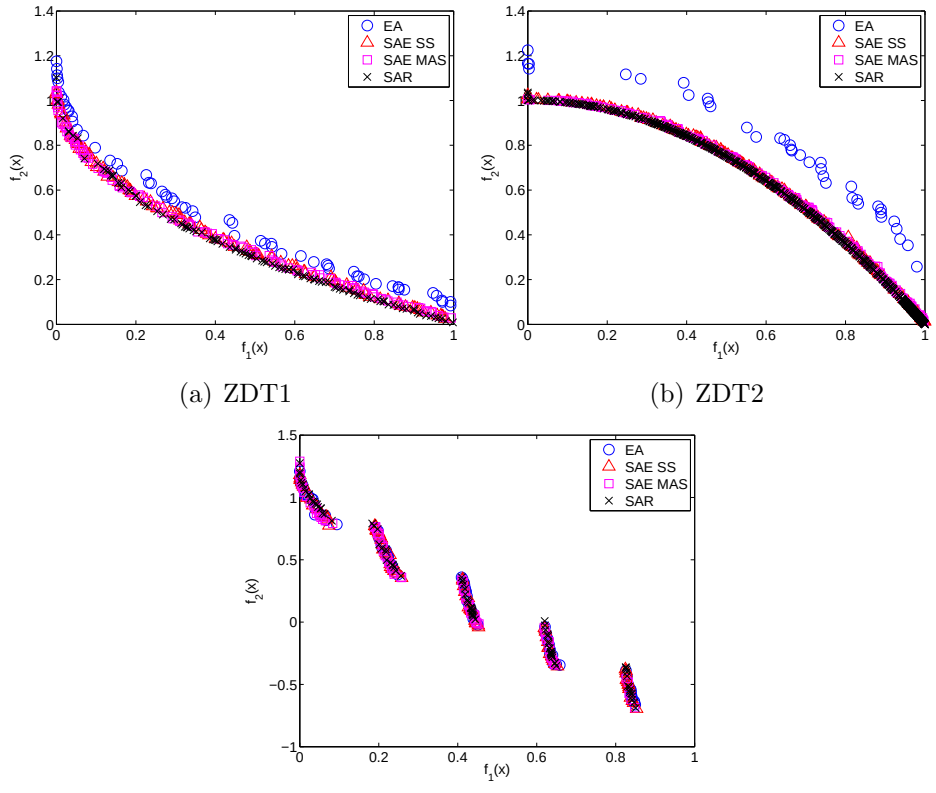
Table 4.13: The worst displacement metric values for ZDT problems using EA, SAE-EA and SAR-EA

Problem	EA	SAE-SS	SAE-MAS	SAR
ZDT1	0.0077	0.0060	0.0086	0.0091
ZDT2	0.0512	0.0313	0.0320	0.0069
ZDT3	0.0090	0.0180	0.0167	0.0122

are taken from a run corresponding to the average displacement performance. The non-dominated solutions for SAE-EA and SAR-EA have converged closer to the Pareto front than EA for ZDT1 and ZDT2. The performance of all the algorithms is comparable for ZDT3.

4.5.5 Constrained Multi-objective Optimization

From the seven CTP problems, spatial surrogate models could be built for first five problems. The results for problems CTP7 and CTP8 are similar to EA in the absence of any valid surrogate models and are not presented. All the CTP problems use generalized Rastrigin function which is one of the unconstrained single-objective optimization problems, f09. In the problem f09, 30 design variables are used, whereas in CTP problems, ten design variables are used. This demonstrates that 500 training solutions are inadequate to build a spatial surrogate for a 30-variable Rastrigin function, but are insufficient for a



(a) ZDT1 (b) ZDT2

(c) ZDT3

Figure 4.7: Non-dominated solutions obtained using EA, SAE-EA and SAR-EA for ZDT problems

ten-variable Rastrigin function.

The best values of the displacement metric for the CTP problems are listed in Table 4.14. The displacement values for EA and SAE-EA with multiple adaptive surrogates are comparable. The performance using a single surrogate model is worse than multiple adaptive surrogates. For problems CTP2–CTP5, the Pareto front is disconnected and a single surrogate has difficulty capturing the Pareto front. However, multiple surrogates have a better chance at capturing the different regions of the Pareto front.

The average values of the displacement metric for CTP problems are shown in Table 4.15. The constraint function in CTP problems CTP2–CTP5 is highly

Table 4.14: The best displacement metric values for CTP problems using EA, SAE-EA and SAR-EA

Problem	EA	SAE-SS	SAE-MAS	SAR
CTP2	0.0010	0.0030	0.0022	0.0009
CTP3	0.0141	0.0221	0.0139	0.0166
CTP4	0.0666	0.0731	0.0691	0.0700
CTP5	0.0015	0.0022	0.0016	0.0011
CTP6	0.0103	0.0130	0.0032	0.0011

nonlinear and multi-modal. The performance of SAE-EA and SAR-EA is worse than EA for these problems. Multiple adaptive surrogates perform better than a single surrogate model for CTP2–CTP5. Also, for constrained problems incorrect determination of feasibility can lead to the rejection of those solutions from the population. For CTP problems, the Pareto solutions are on the constraint boundary and even small error in prediction of constraint values can transform an infeasible solution into a feasible one and vice versa. This problem is evident in the results of problems CTP2–CTP5. On the other hand, the convergence of SAE-EA and SAR-EA is better than EA for problem CTP6, which has a linear Pareto optimal front. The worst displacement metric values for CTP problems are reported in Table 4.16. It can be seen that displacement values for SAE-EA and SAR-EA are smaller than EA. This indicates that the spatial surrogates are able to identify the regions of Pareto solutions correctly and the population quickly migrates to those regions. The exploitation of spatial surrogates in SAR-EA results in lower values of displacement metric using SAR-EA as compared to SAE-EA. Once the solutions are closer to the Pareto front, the evolutionary search is driven by feasibility and constraint handling.

The non-dominated solutions obtained using EA, SAE-EA and SAR-EA for

Table 4.15: The average displacement metric values for CTP problems using EA, SAE-EA and SAR-EA

Problem	EA	SAE-SS	SAE-MAS	SAR
CTP2	0.0261	0.0360	0.0348	0.0345
CTP3	0.1313	0.1996	0.1901	0.1652
CTP4	0.1825	0.2381	0.2294	0.2595
CTP5	0.0170	0.0247	0.0233	0.0219
CTP6	0.1350	0.1120	0.1001	0.0219

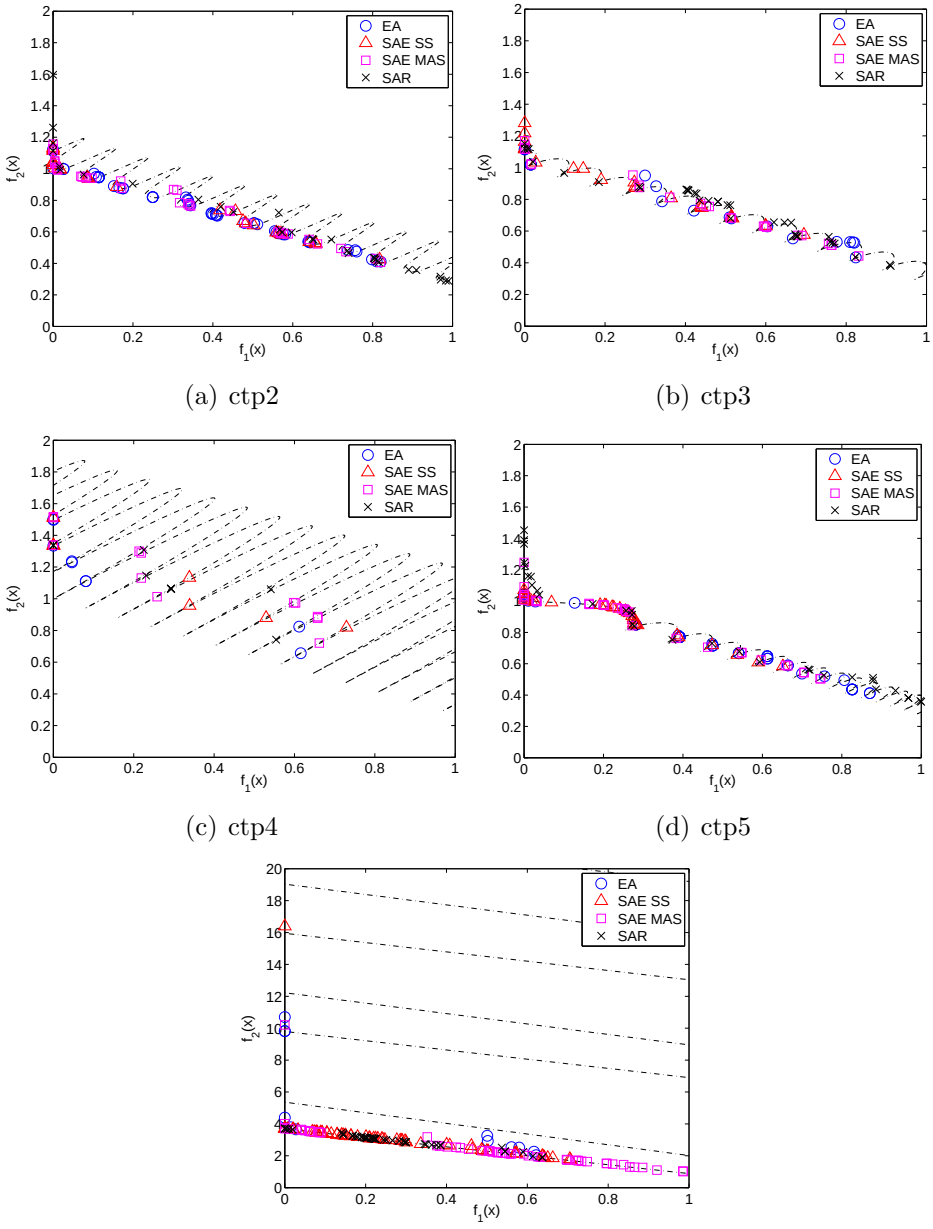
Table 4.16: The worst displacement metric values for CTP problems using EA, SAE-EA and SAR-EA

Problem	EA	SAE-SS	SAE-MAS	SAR
CTP2	0.1292	0.1154	0.1157	0.1194
CTP3	0.6385	0.6879	0.5126	0.6187
CTP4	0.6385	0.7590	0.7486	0.7412
CTP5	0.1141	0.1213	0.0837	0.1209
CTP6	0.4042	0.4159	0.3688	0.1209

a run corresponding to the average displacement metric are shown in Figure 4.8. (The dash-dot line in figures show the constraint boundary.) The spatial surrogates have difficulty capturing multi-modal variation of the constraint function everywhere and are able to capture only some of the regions of the disconnected Pareto fronts.

4.6 Summary

Spatial surrogate modeling, a framework for surrogate modeling in EA using multiple types of surrogate models is proposed. Spatial surrogate models are trained using an archive of all the solutions generated during the evolutionary search and evaluated using actual analysis. As the population migrates towards



(a) ctp2 (b) ctp3

(c) ctp4 (d) ctp5

(e) ctp6

Figure 4.8: Non-dominated solutions obtained using EA, SAE-EA and SAR-EA for CTP problems

the region(s) containing optimum solutions as a result of the evolutionary search, spatial surrogates trained using these solutions approximate the regions (containing optimum solutions) better and can converge faster using the spatial surro-

gate models. The offspring solutions are periodically evaluated using the actual simulations to ensure that the spatial surrogate assisted search is not misguided. Spatial surrogates use a fraction of the solutions in the archive for training and the rest of the solutions are used to validate the spatial surrogate models to ensure good prediction generalization and avoid over-fitting. In the spatial surrogate modeling framework, either a single or multiple surrogate models can be built. The K-Means clustering algorithm is used to partition the solutions in the archive into multiple clusters and individual spatial surrogate model is built using the solutions belonging to each cluster. The specification of number of partitions for multiple spatial surrogates is difficult in general and an adaptive approach based on prediction accuracy is proposed for multiple adaptive surrogates.

Two surrogate assisted evolutionary algorithms are proposed using spatial surrogates – surrogate assisted evaluation and surrogate assisted recombination. In SAE-EA, the spatial surrogates are used in lieu of the expensive analysis for evaluating the fitness of the candidate solutions generated in the evolutionary search. In SAR-EA, the candidate solutions are evaluated using the actual analysis, but the candidate solutions are selected using the fitness derived from spatial surrogates. The proposed algorithms are tested on a number of unconstrained and constrained, single and multi-objective optimization problems. The findings are listed below.

1. Spatial surrogate models using RSM and RBF are able to approximate the responses for most of the optimization problems except functions f04, f08, f09, g02, g08, ZDT4, ZDT5, CTP7 and CTP8. One reason for surrogate models unable to approximate these functions is the limited number of training samples. In this study, the maximum number of training solutions used is 500. Spatial surrogates are able to approximate Rastrigin function

with ten variable using 500 training solutions, but not when there are 30 variables. Using other types of surrogate models, e.g. Kriging or MLP, might help in approximating the responses at the cost of increased training time or requiring specification of additional parameters.

2. With a small value for the prediction error threshold (0.02 in this case), spatial surrogate models could not be built for f02, f10, f12, g09 and ZDT3 in many of the runs. However, the prediction error threshold value of 0.05 has worked across all the problems.

3. The surrogate assisted evaluation (SAE-EA) algorithm outperforms EA for unconstrained and constrained single objective optimization problems. The objective values obtained using SAE-EA for single objective optimization problems, in many cases, are orders of magnitude better than EA.

4. In surrogate assisted recombination (SAR-EA) algorithm, spatial surrogate models are exploited to find solutions very close to the optimum solutions within 1,000 evaluations for single objective problems f01, f03, f06, f07, g01, g04, g06 and g07.

5. In the case of unconstrained multi-objective optimization problems, SAE-EA and SAR-EA outperform EA for problems ZDT1 and ZDT2.

6. For CTP problems, handling a multi-modal constraint function is quite challenging. Multiple adaptive surrogates perform better than single surrogates for problems CTP2–CTP5. The non-dominated solutions obtained using SAE-EA and SAR-EA reach the regions of the Pareto optimal solutions faster than EA. Closer to the Pareto optimal front, the convergence is dictated by the accuracy of approximation of the constraint function. Solu-

tions assessed feasible by spatial surrogates can be in reality infeasible and for problems CTP2–CTP5 the narrow feasible regions present additional difficulty.

The studies on the constrained optimization problems have highlighted the problem of misjudging feasibility of solutions using the spatial surrogate models. This can adversely affect the performance of spatial surrogate assisted EAs when the constraint functions are highly multi-modal. Instead of treating the solutions infeasible outright, "softer" handling of constraint violations may be required to keep up the performance of SAE-EA and SAR-EA for constrained optimization problems. A constraint handling method that does not discard infeasible solutions during evolutionary search is proposed in the next chapter and the benefits, if any, are studied in the context of an EA.

Chapter 5

Constraint Handling in EA

5.1 Overview

Real life optimization problems involve one or more constraints and in most cases, the optimal solutions to such problems lie on constraint boundaries. The performance of a constrained optimization algorithm is known to be largely dependent on the mechanism used for constraint handling. Evolutionary algorithms in general lack explicit mechanisms to handle constraints. However, many methods have been proposed in the literature to handle constraints within EA. The proposals include use of a composite penalty function that converts the values of the objectives and the constraints into a single fitness value, treating the constraint as additional objectives and solving the constrained problem as a multi-objective optimization problem. The existing methods of constraint handling are discussed in Section 5.2.

Most of the constraint handling techniques focus on generating the best feasible solution of the problem and hence a feasible solution is always considered better than an infeasible solution during the course of the evolution. This

fundamental assumption always drives the population first towards feasibility and then towards a constraint boundary from within the feasible region. In reality a designer is often interested to look at the solutions that might be marginally infeasible in addition to the optimum solutions. Hence, a marginally infeasible solution near the constraint boundary is more desirable than a poor feasible solution away from the constraint boundary.

In this chapter, a constraint handling method is proposed that searches through the infeasible as well as the feasible regions. The constraint handling method maintains infeasible solutions in the population and ranks a few infeasible solutions higher than feasible solutions to focus the search along the constraint boundary. The mechanism of maintaining infeasible solutions in the population is described in Section 5.3. An infeasibility driven evolutionary algorithm (IDEA) using the proposed constraint handling method is proposed in Section 5.4. In IDEA, the original constrained optimization problem is converted into an unconstrained optimization problem with an additional objective that represents constraint violation measure. The performance of IDEA is compared to EA on a number of single and multi-objective constrained optimization problems in Section 5.5. The summary of findings is presented in Section 5.6.

5.2 Constraint Handling Methods

The most common approach for constraint handling in EA is to use a penalty function. A penalty function converts a constrained optimization problem into an unconstrained optimization problem. There are two types of penalty functions used in optimization: interior and exterior with respect to the feasible design space. Using an interior penalty function, a small penalty is imposed on feasible

solutions far away from the constraint boundary and the penalty approaches infinity as the solutions approach the constraint boundary. Since interior penalty methods require a feasible starting point, they are not preferred. The exterior penalty methods can handle infeasible solutions and are used in EA. An exterior penalty function is formulated as,

$$f_p(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{m} w_i * G(g_i(\mathbf{x})) \tag{5.1}$$

where $f_p(\mathbf{x})$ is the penalty function, $f(\mathbf{x})$ is the original objective, $g_i(\mathbf{x})$ are the constraints and w_i are the penalty factors for each constraint. The function $G(\cdot)$ is 0 if the constraint is satisfied and takes the value of the constraint violation if the constraint is not satisfied. (See [128] for detailed discussion on penalty functions.) The various EA implementations using penalty function include – static penalty where the penalty function is fixed [129, 130]; dynamic penalty where the penalty factors are derived from the current EA generation [131]; annealing penalty where the penalty is increased over time in the fashion of SA [132, 133]; adaptive penalty where the penalty is scaled based on the success or failure of the evolutionary search [134, 135, 136, 137]; and death penalty where the solution is assigned zero fitness if it violates any of the constraints [138]. The penalty function approach though simple, requires assignment of penalty factors which are often obtained based on trial and error and the result of optimization is known to be highly sensitive to the choice of the penalty factors. Runarsson and Yao [139] have proposed stochastic ranking to balance between the objective and the penalty. Although this approach does not need any penalty factors, it uses a probability value (between 0.4 and 0.5) to compare infeasible individuals. Xiao and Zu [140] extended this method for multi-objective optimization.

In an attempt to alleviate the problem of specifying penalty factors, a number of alternate approaches have been proposed. These include special representation schemes to maintain feasibility [141], search for the constraint boundary by crossing the boundary back and forth using special operators [142], transforming the original problem into easier to optimize function using homomorphous mapping [143], and use of repair algorithms [144, 145, 146, 147]. The main drawbacks of these approaches are the need to develop problem specific representation schemes and corresponding operators, problem specific repair mechanisms (to produce feasible solutions from infeasible solutions), extra computational effort in the operators, and early loss of diversity.

Another method for constraint handling is to handle feasible and infeasible solutions separately. Powell and Skolnick [148] mapped the feasible solutions in the interval $(-\infty, 1)$ and infeasible solutions in the interval $(1, \infty)$ to construct a single fitness function. A similar approach by Deb [149] used the objective value as the fitness for the feasible solutions and the total constraint violation as fitness for infeasible solutions. This constraint handling is now part of NSGA-II. For single objective optimization, Coello [150] proposed to split the population into $1+m$ (where m is the number of constraints) sub-populations, each population using either the objective or one of the constraints as the fitness function.

Dominance based constraint handling approaches convert the constrained optimization problem (with k objectives) into multi-objective ($k+1$ objectives or $k+m$ objectives, where m is the number of constraints) unconstrained optimization problem. Surry and Radcliffe [151] proposed COMOGA, in which the solutions are Pareto ranked based on the sum of constraint violation and the binary tournament that uses either the ranks of the solutions or the objective function values dynamically based on a parameter. Camponagara and Talukdar [152] used

the sum of constraint violations as the additional objective. For single objective optimization, Coello [153] used constraints as additional objectives and solved the multi-objective optimization problem using NPGA. For multi-objective problems Vieira *et al.* [154, 155] used constrains as additional objectives with modified Parks & Miller elitist technique. In the above approaches a significant time may be spent on non-dominated sorting in the presence of large number of constraints. There is also a risk of generating solutions excellent objective function values but with poor constraint satisfaction.

Few researchers have proposed maintaining a proportion of infeasible solutions in the population during the evolution. Hamida and Schoenauer [156] developed adaptive segregational algorithm (ASCHEA), where the proportion of infeasible solutions in the population was controlled using an adaptive penalty. The approach used single penalty coefficient for all constraints, and was later extended to incorporate a separate penalty coefficient for each constraint [157]. Hinterding and Michalewicz [158] proposed CONGA for constraint handling using effective parent matching, where mating was done between two infeasible solutions satisfying different constraints to create offspring which would satisfy all the constraints. Mezura-Montes and Coello [159] suggested a simple multi-membered evolution strategy (SMES) where the "best" infeasible solution determined by its objective function value is allowed to be copied into the next generation. Though these algorithms effectively illustrated the benefits of preserving infeasible solutions in the population, their scope was limited to single-objective optimization problems.

5.3 Maintaining Infeasible Solutions

To enable EA to search through the feasible and the infeasible regions of the design space, the population must contain infeasible solutions. In the absence of infeasible solutions, the search will always be restricted to the feasible region. In NSGA-II (EA used in this study), the infeasible solutions are eliminated from the population since the elite preservation mechanism prefers feasible solutions to infeasible solutions. In the proposed approach, the elite preservation mechanism is modified to select a few infeasible solutions along with the feasible solutions. Then the question is how many infeasible solutions are to be retained. A user defined parameter, *infeasible ratio*, is introduced to determine the number of infeasible solutions in the population. The infeasible ratio denotes the percentage of the population to be retained infeasible.

The evolutionary search in NSGA-II is controlled by the selection, crossover and mutation operators. The selection process chooses the solutions for mating. The creation of offspring is dictated by the crossover operation. Two feasible solutions undergoing crossover are most likely to produce feasible offspring. If only the feasible parents are used in the evolution process, the search will be restricted to the feasible region. To be able to search through the infeasible space, the evolution process must create infeasible offspring and therefore must select infeasible solutions are parents. Selection is another place in NSGA-II where the feasible solutions are preferred over the infeasible solutions. The selection process uses a binary tournament between two solutions to determine a parent as follows.

1. If both solutions are feasible, the solution with the better objective value wins.

2. If both solutions are infeasible, the solution with the smaller constraint

violations wins.

3. If one solution is feasible and other is infeasible, the feasible solution wins.

Using this mechanism, an infeasible solution can be selected as a parent only if both the solutions undergoing binary tournament are infeasible.

The solutions to the constrained problems are often on the constraint boundary and searching near the constraint boundary can increase the chances of finding the optimum solutions. The region around the constraint boundary can explored by crossover between a feasible and an infeasible solutions. In the proposed approach this is achieved by promoting the infeasible solutions over feasible solutions. Few of the infeasible solutions in the population are ranked higher than the feasible solutions as depicted in Figure 5.1. Let N, N_f, N_{inf}, α be the population size, the numbers of feasible and infeasible solutions, and the infeasible ratio respectively. Let $N_r = \alpha \times N$ be the number of infeasible solutions to be retained in the population. The feasible and the infeasible solutions are ranked separately using appropriate techniques. The combined ranks are assigned in the following way. If the number of infeasible solutions (N_{inf}) is more than N_r, then the first N_r solutions are selected based on the rank; otherwise all the infeasible solutions are selected. These solutions are ranked from 1 to N_r. Then all the feasible solutions are ranked from $N_r + 1$ to $N_r + N_f$ based on their rank. Any infeasible solutions left are assigned ranks starting from N_r+N_f+1. Additionally, the binary tournament is modified to compare only the ranks of the solutions and pick the solution with a lower rank as the winner.

The proportion of infeasible solutions in the population is controlled by the parameter infeasible ratio. It takes values between zero and one. A zero value for the infeasible ratio means no infeasible solutions and a value of one means all

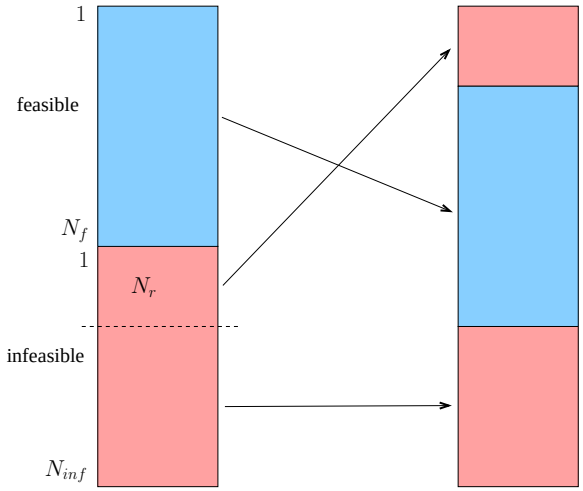


Figure 5.1: Ranking procedure to retain infeasible solutions in the population

the solutions are infeasible. In the case of multi-objective optimization problems, if a large proportion of solutions are infeasible, then the entire Pareto front may not be represented by the small number of feasible solutions. In case of single objective optimization, there is no such limitation on the number of infeasible solutions in the population. As the primary motivation of maintaining infeasible solutions in the population is to aid the search through infeasible space, even a small fraction of infeasible solutions is adequate.

5.4 Infeasibility Driven EA (IDEA)

The proposed infeasibility driven evolutionary algorithm uses a multi-objective formulation of the optimization problem for constraint handling. The original k objective constrained optimization problem is reformulated as $k+1$ objective unconstrained optimization problem as given in Equation 5.2. The first k objectives are the same as in the original optimization problem. The additional objective

represents a measure of constraint violation.

$$\begin{aligned} \text{Minimize} \quad & f'_1(\mathbf{x}) = f_1(\mathbf{x}), \ldots, f'_k(\mathbf{x}) = f_k(\mathbf{x}) \\ & f'_{k+1}(\mathbf{x}) = \text{Violation measure} \end{aligned} \tag{5.2}$$

The main steps of IDEA are the same as EA as outlined in Algorithm 2.1. The feasible solutions are ranked using non-dominated sorting and crowding distance for more than one objectives. For the feasible solutions, the constraint violation measure is zero and non-dominated sorting with $k+1$ objectives is the same as the non-dominated sorting with original k objectives. In the case of single objective optimization, the feasible solutions are sorted based on the objective value. The infeasible solutions are ranked with $k+1$ objectives using non-dominated sorting and crowding distance. Since the violation measure value for feasible solutions is zero and for infeasible solutions greater than zero, all the solutions can be sorted together. The combined ranks are assigned as shown in Figure 5.1. Two constraint violation measures are considered.

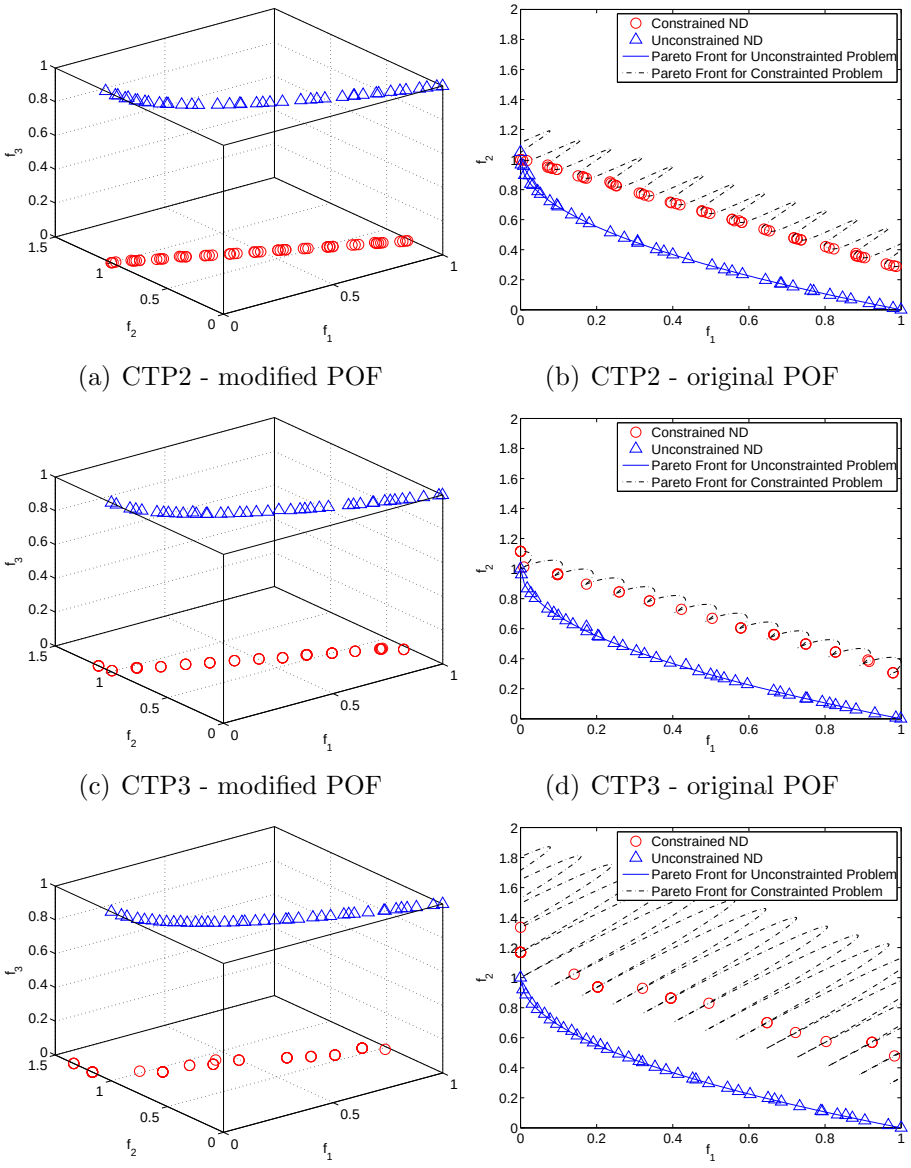
5.4.1 Constraint Violation Count

To anchor the infeasible solutions on the infeasible region of the design space, the solutions to the unconstrained version (obtained by dropping constraints) of the optimization problem are considered. For the constrained optimization problem, the optimum solutions lie on the constraint boundary (provided the constraints are active). The solutions of the unconstrained problem obtained by dropping the constraints in the original problem would lie in the infeasible region of the original problem. These solutions would violate one or more constraints in the original problem. If the constraints are active but redundant, then the solutions

to the original problem and the unconstrained problem coincide. Thus, one of the ways to find infeasible solutions is to consider the number of violated constraints as additional objective. This forms the *Constraint violation count* measure.

The results of CTP problems (CTP2–CTP4 and CTP6–CTP8) using IDEA with constraint violation count are shown in Fig 5.2 and Figure 5.3. The figures on the left hand side show the non-dominated solutions obtained using additional objective in the modified formulation, the figures on the right are the IDEA solutions plotted in the objective space of the original problem. These results are obtained for a population size of 100 evolved over 200 generations using an infeasible ratio of 0.4. The non-dominated solutions obtained by IDEA for a modified CTP2 problem is shown in Figure 5.2(a). The solutions corresponding to $f_3 = 0$ (zero constraint violation) are the solutions of the original constrained CTP2 problem. The solutions corresponding to $f_3 = 1$ represent the solutions to the unconstrained problem with a single constraint violation. The same solutions are plotted in the objective space for the original CTP2 problem in Figure 5.2(b). For CTP3 and CTP4, the Pareto optimal front comprises of discontinuous regions with a single point in each region. The search from the feasible side has to move along the narrow regions of the feasible space for CTP4 (Figure 5.2(f)) and many evolutionary algorithms have difficulty to find the Pareto optimal solutions for such problems.

CTP6 has many local Pareto fronts and the solutions can get trapped in the non-optimal Pareto fronts. The presence of infeasible solutions helps the feasible solutions to move towards the Pareto optimal front early in the search. The non-dominated solutions for the constrained and the unconstrained CTP6 problem are shown in Figure 5.3(a) and Figure 5.3(b). The Pareto optimal solutions for CTP7 form a subset of the solutions to the unconstrained CTP7



(a) CTP2 - modified POF

(b) CTP2 - original POF

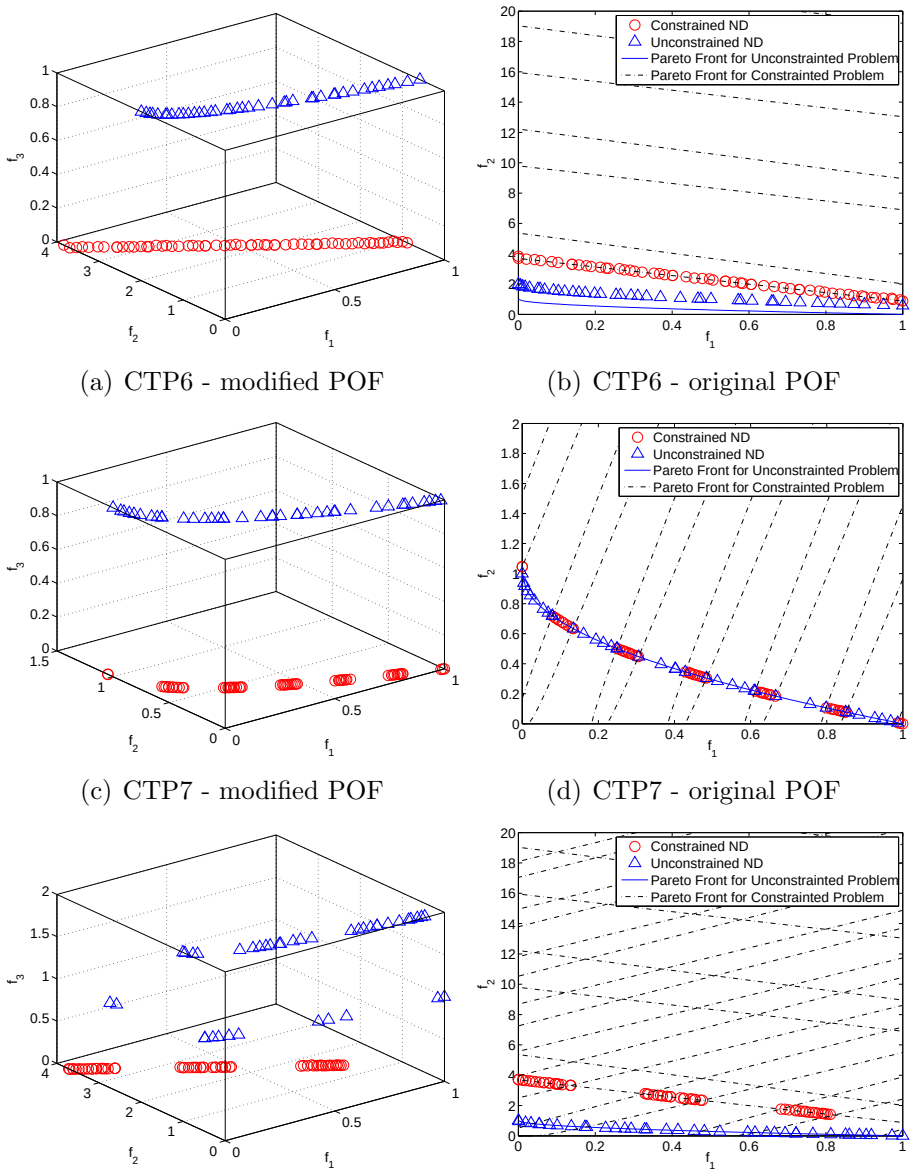
(c) CTP3 - modified POF

(d) CTP3 - original POF

(e) CTP4 - modified POF

(f) CTP4 - original POF

Figure 5.2: IDEA results for CTP2, CTP3 and CTP4 using constraint violation count measure.



(a) CTP6 - modified POF

(b) CTP6 - original POF

(c) CTP7 - modified POF

(d) CTP7 - original POF

(e) CTP8 - modified POF

(f) CTP8 - original POF

Figure 5.3: IDEA results for CTP6, CTP7 and CTP8 using constraint violation count measure.

problem (Figure 5.3(d)). The solutions of the constrained CTP7 problem appear as holes in the non-dominated set of solutions for the unconstrained problem ($f_3 = 1$) as shown in Figure 5.3(c). For CTP8, the solutions corresponding to a single constraint violation ($f_3 = 1$) and both constraint violations ($f_3 = 2$) are shown in Figure 5.3(e). As seen in Figure 5.3(f), the solutions to the unconstrained CTP8 problem correspond to either one or both constraint violations.

5.4.2 Constraint Relative Rank

The infeasible solutions obtained using constraint violation count are on the Pareto optimal front of the unconstrained version of the optimization problem. This Pareto optimal front can be quite far from the constraint boundary, or it might intersect partially or fully with the constraint boundary. Infeasible solutions far away from the constraint boundary are of not much interest to the designers as compared to marginally infeasible solutions. To restrict the infeasible solutions moving too far away from the constraint boundary a *constraint relative rank* measure is proposed.

The constraint relative rank measure is based on the amount of relative constraint violation among the population members. The constraint relative rank of a solution is based on the constraint violation levels for all constraint values for that solution. Consider one of the constraints (g_i). All the solutions in the population are sorted in ascending order based on the value of the constraint violation for g_i. The solution with the least constraint violation value gets first rank. The solutions that do not violate the constraint g_i have zero constraint violation and get zero rank. Each solution is assigned a constraint violation rank for constraint g_i based on the sorted list. Solutions with the same value of constraint violation get the same rank. This ranking procedure is repeated for all

the constraints. The constraint relative rank for each solution is then calculated as the sum of the ranks (based on constraint violation level) obtained for all the constraints.

The process of determining constraint relative ranks is illustrated using an example. Consider an optimization problem with three constraints (g_1, g_2, g_3). A sample population of ten individuals is shown in Table 5.1. For each solution, the first three columns contain the value of the constraint violation (which is the absolute value of the constraint function). The constraint violation values are sorted for each constraint and each solution is assigned relative ranks for the constraints. The relative ranks are shown in the next three columns. Solutions 3, 7 and 9 do not violate constraint g_1 and get a zero relative rank. Solution 4 with the least constraint violation value of 1.25 for g_1 gets first rank and solution 6 with the highest constraint violation value of 100.70 gets seventh rank. The constraint violation measure as the sum of the ranks with respect to each constraint is given in the last column.

Table 5.1: Calculation of constraint relative rank (CRR)

Solution	Violations			Relative ranks			CRR
	g_1	g_2	g_3	g_1	g_2	g_3	
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

It can be seen that the violation measure favors solutions with good ranks for most (or all) of the constraints. As a result, a solution with a large violation in only one of the constraints would roughly have the same preference as a solution with marginal violations of multiple constraints. The violation measure is used as the additional objective in IDEA to rank the infeasible solutions using non-dominated sorting. Consequently, the final population will contain solutions with marginal constraint violations.

5.5 Results of Numerical Benchmarks

The performance of IDEA using the constraint relative rank procedure is compared to EA using single and multi-objective constrained optimization test problems. The test problems used for single objective optimization are g-series problems (refer to Appendix B for details) and CTP (refer to Appendix D for details) for multi-objective optimization. All the problems are solved as minimization problems.

5.5.1 Experimental Setup

For single objective optimization problems the number of function evaluations is limited to 4,000 with the population size of 40. Both the algorithms, EA and IDEA, are run for 100 generations. For multi-objective optimization problems the number of function evaluations used is 20,000 with the population size of 100 evolved over 200 generations.

Multiple runs are performed for EA and IDEA by varying the parameters similar to the surrogate assisted EA as described in Section 4.5.1. Two values each are considered for EA parameters – crossover probability, mutation probability,

crossover distribution index and mutation distribution index. Four values of random seed are considered. The parameters values used are the same as listed in Table 4.1. The number of independent runs for each problem using EA is 64. In IDEA, there is an additional parameter, the infeasible ratio. Two values are considered for the proportion of infeasible solutions retained in the population, 10% and 20%. The number of independent runs for each problem using IDEA are $(64 \times 2 =)$ 128.

5.5.2 Single Objective Optimization

The best objective values obtained for g-series problems with EA and IDEA are listed in Table 5.2. The objective values using IDEA are better than EA for problems g01, g02, g04 and g10 and similar for the rest of the problems. The average performance of IDEA is better than EA in all the problems except g01 as seen from Table 5.3. For g01, the average values of the objective function are close. A similar trend can be observed for the worst values reported using EA and IDEA in Table 5.4. The worst objective values obtained using IDEA are much lower than the values obtained using EA. For problem g02 the worst objective values are the same for EA and IDEA.

The convergence trends for EA and IDEA on g-series problems are shown in Figure 5.4 and Figure 5.5. The convergence rate of IDEA is better than EA for problems g02, g04, g07 and g10 as the objective values continue to fall over generations (or equivalently number of function evaluations) more rapidly in IDEA than EA. For the rest of the problems the trends are similar without significant improvement.

One of the advantages of IDEA is that the final population contains a few infeasible solutions that marginally violate one or more constraints. Such trade-off

Table 5.2: Best objective values for g-series problems using EA and IDEA

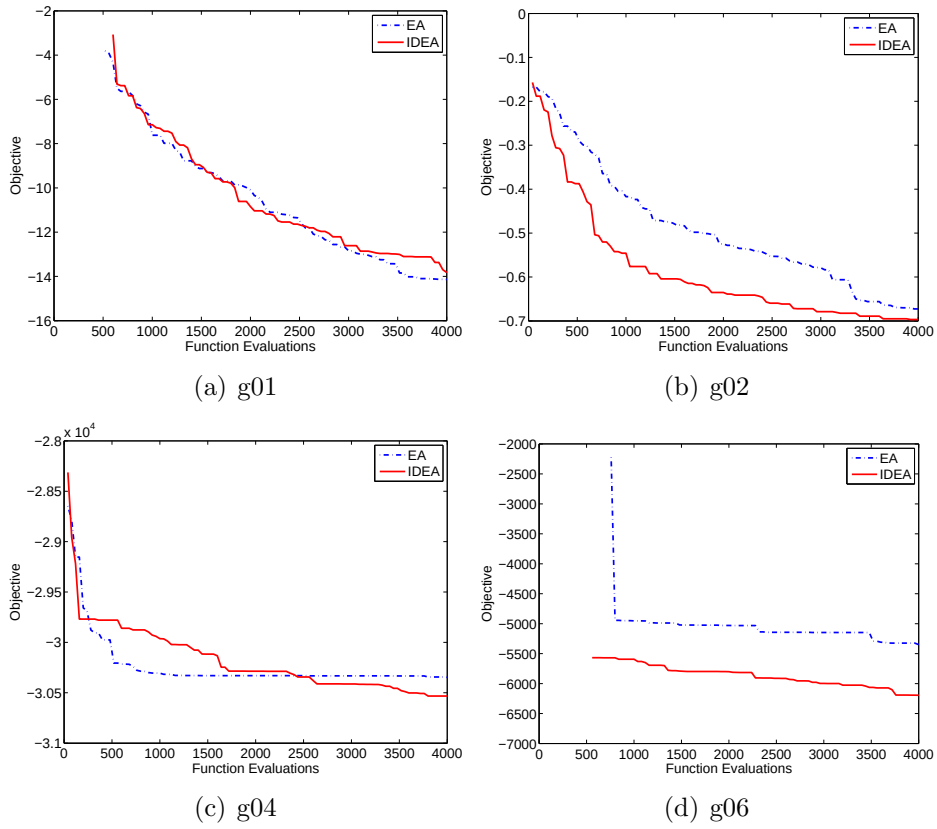
Problem	Optimum	EA	IDEA
g01	-15	-14.72	-14.83
g02	-0.8036	-0.74	-0.78
g04	-30665.539	-30578.3	-30658.5
g06	-6961.81	-6847.95	-6932.74
g07	24.306	25.76	25.62
g08	-0.0958	-0.0958	-0.0958
g09	680.63	681.81	681.11
g10	7049.33	7415.57	7451.15

Table 5.3: Average objective values for g-series problems using EA and IDEA

Problem	Optimum	EA	IDEA
g01	-15	-12.92	-12.78
g02	-0.8036	-0.62	-0.65
g04	-30665.539	-30225.23	-30440.94
g06	-6961.81	-4901.43	-5863.24
g07	24.306	62.44	32.99
g08	-0.0958	-0.0751	-0.0816
g09	680.63	697.83	690.46
g10	7049.33	11193.19	10704.61

Table 5.4: Worst objective values for g-series problems using EA and IDEA

Problem	Optimum	EA	IDEA
g01	-15	-7.22	-7.31
g02	-0.8036	-0.44	-0.44
g04	-30665.539	-29799.2	-29891.9
g06	-6961.81	-1240.14	-4764.37
g07	24.306	779.72	55.66
g08	-0.0958	-0.0227	-0.0258
g09	680.63	794.04	727.02
g10	7049.33	22587.5	16290.9

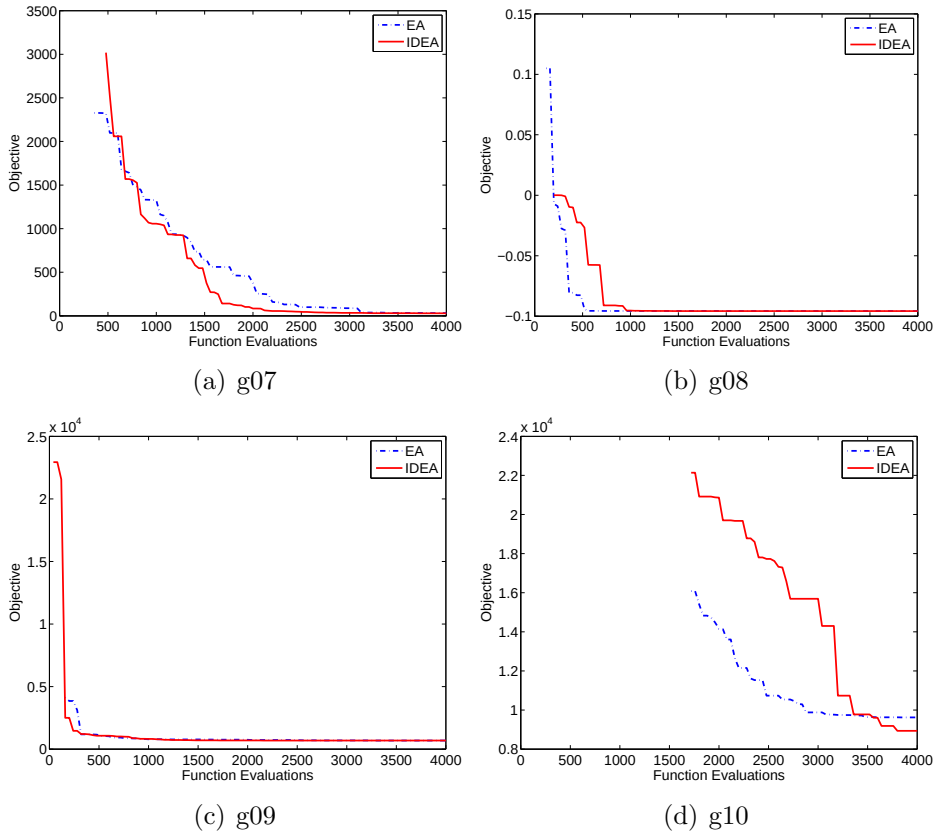


(a) g01 (b) g02

(c) g04 (d) g06

Figure 5.4: Average convergence trends of EA and IDEA for g-series problems (g01, g02, g04 and g06)

solutions can be of importance to the designers as these solutions illustrate the quantifiable improvement in the objective value if one or more constraints are relaxed. Listed in Table 5.5 are some of the infeasible solutions obtained by IDEA, the objective values and the constraint violations. The optimum objective value for problem g06 is -6961.81. Better objective values can be obtained by relaxing one or both the constraints as seen in Table 5.5. The constraint violations values for both the constraints are orders of magnitude smaller than the average constraint violation values in the initial population of EA and IDEA. The constraint violation values are, in fact, of the same order as the constraint values of the feasible solutions close to the optimum. Thus, allowing marginal



(a) g07 (b) g08

(c) g09 (d) g10

Figure 5.5: Average convergence trends of EA and IDEA for g-series problems (g07–g10)

constraint violation can improve the objective value by up to 5%.

5.5.3 Multi-objective Optimization

A summary of the displacement metric values for the non-dominated solutions obtained using EA and IDEA for CTP problems is listed in Table 5.6. It can be seen from the minimum values of the displacement metric that the best non-dominated solutions obtained using EA and IDEA are very close to the Pareto front. The average values of the displacement metric for IDEA are consistently better than EA. It indicates that IDEA has better convergence for

Table 5.5: Marginally infeasible solutions obtained using IDEA for problem g06

x_1	x_2	f	C1	C2
14.0951	0.8403	-6964.7233	0.0	0.0236
14.0587	0.8028	-7007.9283	0.3235	0.0
14.0622	0.7722	-7041.6570	0.0021	0.0635
14.0589	0.7001	-7121.5879	0.0	0.6213
13.9484	0.5316	-7317.2921	0.0398	0.3330
13.8099	0.2998	-7590.2439	0.2931	0.2769

CTP problems than EA. Also the worst performance of IDEA is as good as EA for problems CTP2, CTP3, CTP4 and CTP5 and better than EA for problems CTP6, CTP7 and CTP8.

Table 5.6: Summary of Displacement metric for CTP problems using EA and IDEA

	EA			IDEA		
Problem	Min	Average	Max	Min	Average	Max
CTP2	0.0001	0.0174	0.0830	0.0002	0.0131	0.0830
CTP3	0.0052	0.0930	0.4118	0.0051	0.0680	0.4118
CTP4	0.0397	0.1331	0.4118	0.0274	0.0904	0.4118
CTP5	0.0005	0.0142	0.0767	0.0004	0.0124	0.0767
CTP6	0.0008	0.0543	0.3460	0.0008	0.0027	0.0253
CTP7	0.0001	0.0263	0.1432	0.0001	0.0165	0.0615
CTP8	0.0063	0.2005	0.3996	0.0004	0.0126	0.0672

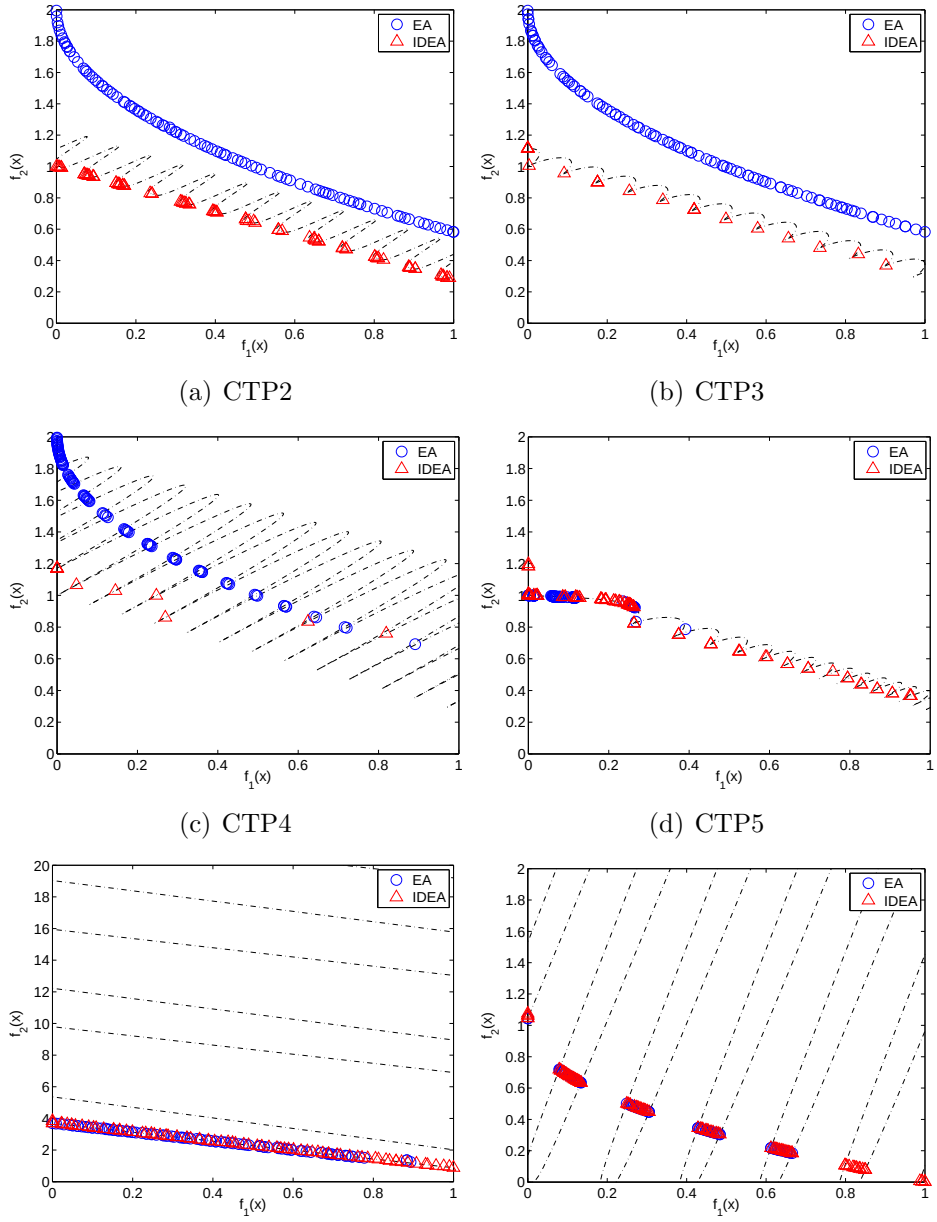
The non-dominated solutions in a runs corresponding to the average displacement metric values of EA and IDEA for CTP problems are shown in Figure 5.6. It can be seen that EA solutions have converged to a local Pareto front for problems CTP2 and CTP3. For problem CTP4, EA has difficulty searching along the narrow feasible regions whereas IDEA can reach the tips of the feasible region traversing through the infeasible region. For problem CTP5, EA solutions have converged only on the small part of the Pareto front. Once the population has

converged, EA has to rely on mutation operation (which is usually performed with a very small probability) to spread the solutions across the Pareto front. In IDEA, the presence of infeasible solutions can generate solutions across the entire Pareto front using crossover operation. This behaviour is also seen for problem CTP6. For problem CTP7, EA solutions have not converged during 200 generations, but IDEA solutions have reached the Pareto front.

The benefit of maintaining infeasible solutions in the population can be seen by observing the evolution of IDEA population across generations. Shown in Figure 5.7 are the non-dominated solutions obtained by EA and IDEA in few generations for problem CTP2. Also, the infeasible solutions in the population using IDEA are shown as crosses. In generation 70, the non-dominated solutions of EA have covered the few disconnected regions of the Pareto front. The same is true for non-dominated solutions by IDEA. In generations 100, 130, 160 and 190 IDEA solutions have progressively spread to other regions of the Pareto front, whereas EA solutions spread much more slowly. With the infeasible solutions close to the constraint boundary, the IDEA population can move across the infeasible regions and cover the entire Pareto front easily.

5.6 Summary

A novel algorithm, Infeasibility Driven evolutionary algorithm (IDEA), is proposed for constrained optimization problems. IDEA maintains infeasible solutions in the population and assigns higher ranks to a few infeasible solutions to actively search through the infeasible regions of the design space. Two methods are proposed to assign ranks – constraint violation rank and constraint relative rank. Using constraint violation rank, one can solve the unconstrained and

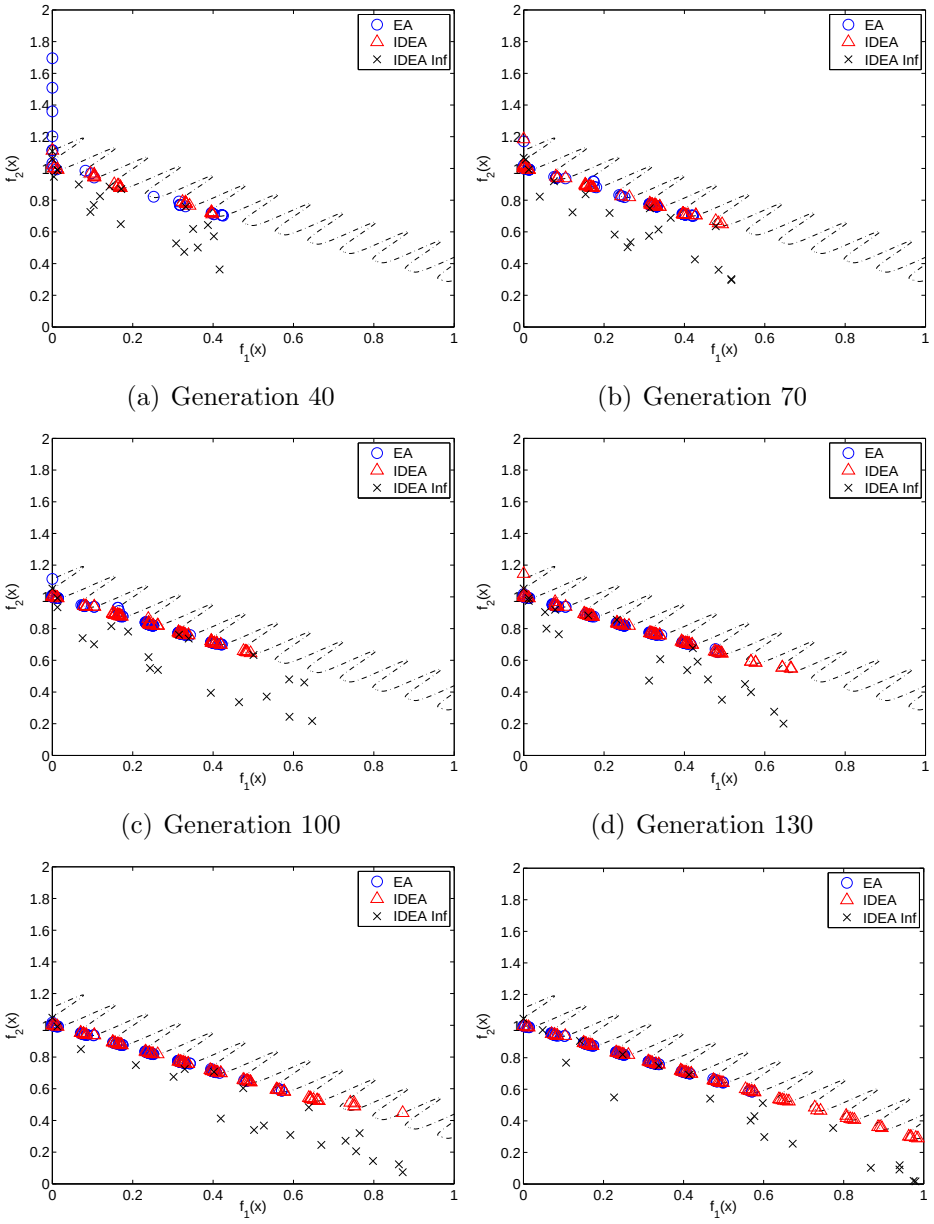


(a) CTP2 (b) CTP3

(c) CTP4 (d) CTP5

(e) CTP6 (f) CTP7

Figure 5.6: Non-dominated solutions obtained using EA and IDEA for CTP problems in a median run



(a) Generation 40 (b) Generation 70

(c) Generation 100 (d) Generation 130

(e) Generation 160 (f) Generation 190

Figure 5.7: Evolution of Non-dominated solutions over generations for problem CTP2 using EA and IDEA

the constrained versions of the optimization problem simultaneously. Although, IDEA can search through the infeasible region using this measure, the infeasible solutions (which are the solutions of the unconstrained problem) can have very large constraint violations and are of not much value to the designers. Constraint relative rank measure on the other hand promotes marginally infeasible solutions, thus retaining solutions that are close to the constraint boundaries.

The performance of IDEA is compared with EA on a number of single and multi-objective constrained optimization problems. The average objective values obtained for single objective optimization problems using IDEA are better than those obtained with EA. Also the average convergence of IDEA is much faster than EA for most of the single objective problems. The added advantage for single objective optimization problems is that IDEA has a set of infeasible solutions in the final population that can be used for trade-off study. These solutions can provide significant improvement at the cost of small constraint violations. For problem g06 the objective value can be improved by up to 5% by relaxing the constraints marginally.

The benefits of simultaneous search through the infeasible as well as the feasible regions are clearly evident from the results of CTP problems. The evolutionary search in IDEA is focused near the constraint boundary resulting in a faster convergence to the Pareto front for all CTP problems. This also prevents solutions from getting stuck in local minima. Another advantage of IDEA is that the non-dominated solutions can spread easily through the infeasible regions to the neighboring disjoint parts of the Pareto front as seen in Figure 5.7.

Chapter 6

Engineering Examples

6.1 Overview

A set of engineering design examples are presented in this chapter. The problems include Belleville and compression spring design, speed reducer gear design, design of a pressure vessel, welded beam design and airfoil design. These problems are well studied in the literature and are widely used to compare performance of optimization algorithms. These engineering examples are single objective optimization problems. The airfoil design problem is also formulated and solved as a bi-objective optimization problem. The individual problems are discussed in following Sections, while a summary of results for all the examples is presented in Section 6.9.

6.2 Experimental Setup

All the engineering problems are solved using EA, SAE-EA, SAR-EA and IDEA. For single objective optimization the number of function evaluations is limited

to 1,000 and for the bi-objective optimization problem it is set at 5,000. The population size for single objective and bi-objective problems is set to 40 and 100 respectively. For each problem, ten independent runs are performed by varying the random seed. The spatial surrogate models are built using RSM, RBF and Kriging. The prediction error threshold for spatial surrogates is set to 0.05. The retain count parameter is set to one for single objective problems and 20 for the bi-objective problem. In SAE-EA, the spatial surrogates are periodically trained every five generations.

6.3 Belleville Spring Design

The Belleville spring design problem is to find the minimum weight spring subject to the stress and displacement constraints [1]. The configuration of the Belleville spring is shown in Figure 6.1. 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 6.1.

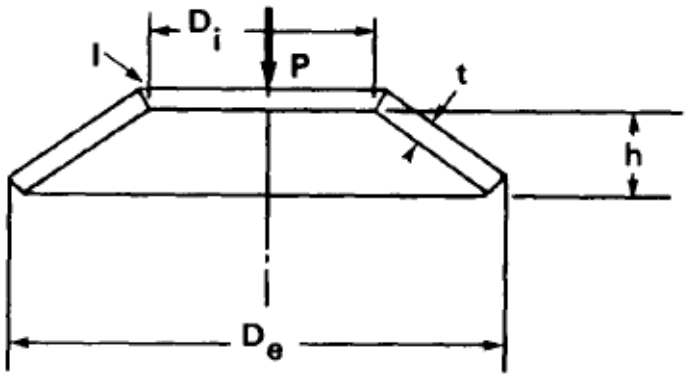


Figure 6.1: Belleville spring configuration (Source: Siddall (1982) [1])

Table 6.1: 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

The Belleville spring design optimization problem is defined as follows:

$$\begin{aligned}
\text{Minimize} \quad & f = \frac{0.283\pi(D_e^2 - D_i^2)t}{4} \\
\text{subject to} \quad & g_1 = \sigma_D - \sigma \geq 0 \\
& g_2 = P - P_{max} \geq 0 \\
& g_3 = \delta_l - \delta_{max} \geq 0 \\
& g_4 = l - h - t \geq 0 \\
& g_5 = D_{max} - D_e \geq 0 \\
& g_6 = D_e - D_i \geq 0 \\
& g_7 = 0.3 - \frac{h}{D_e - D_i} \geq 0
\end{aligned}$$

where, σ is the stress at the upper edge (marked I in Figure 6.1), 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:

$$\alpha = \frac{6}{\pi \ln K}\left(\frac{K-1}{K}\right)^2, \quad \beta = \frac{6}{\pi \ln K}\left(\frac{K-1}{\ln K} - 1\right), \quad \gamma = \frac{6}{\pi \ln K}\left(\frac{K-1}{2}\right)$$

The results for the Belleville spring design obtained using EA and surrogate assisted EA are given in Table 6.2. The best spring design with a weight of 2.29 lb is obtained using SAE-EA with multiple adaptive surrogates which corresponds to 9% improvement in the weight. The average and the worst spring weights obtained using SAE-EA and SAR-EA are less than that of EA corresponding to 10–15% improvement. The performance of IDEA is comparable to SAE-EA performance, which shows that the constraints play an important role in the design of the Belleville spring. The average convergence trends for all the algorithms are depicted in Figure 6.2. The convergence of IDEA is much faster as compared to other algorithms.

Table 6.2: Results of Belleville spring design using EA, SAE-EA, SAR-EA and IDEA

	EA	SAE-SS	SAE-MAS	SAR	IDEA
Best	2.89	2.62	2.29	2.78	2.43
Average	3.83	3.31	3.73	3.63	3.39
Worst	7.15	6.22	7.49	5.76	5.64

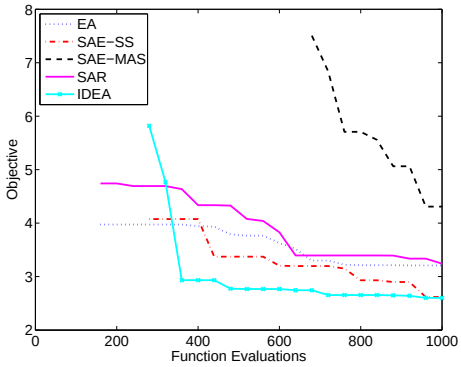


Figure 6.2: Average convergence of EA, SAE-EA and SAR-EA for Belleville spring design

6.4 Design of Coil Compression Spring

A tension/compression spring is shown in Figure 6.3. The design of the spring involves minimizing the weight of the spring subject to constraints on minimum deflection, shear stress, surge frequency [1]. The design variables are the mean coil diameter D, the wire diameter d and the number of active coils N.

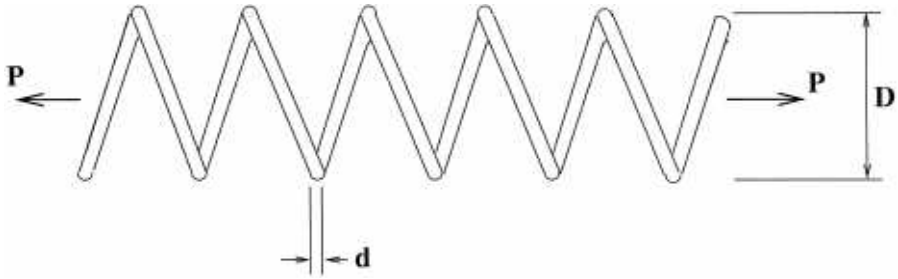


Figure 6.3: Tension/Compression spring

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 6.3. The variable bounds are $0.05 \leq D \leq 2.0$, $0.25 \leq d \leq 1.3$, $2 \leq N \leq 20$.

The results obtained using EA, surrogate assisted EA and IDEA are listed

Table 6.3: 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

in Table 6.4. The best weight obtained for compression spring using all the algorithms is similar. The average performance of IDEA is better than EA (more than 5% improvement) which is better than that of SAE-EA and SAR-EA, which indicates that the design space is highly constrained. The average convergence trends for EA, SAE-EA, SAR-EA and IDEA are shown in Figure 6.4. It can be observed that the feasible solutions are found after approximately 200 function evaluations.

Table 6.4: Results of compression spring design using EA, SAE-EA, SAR-EA and IDEA

	EA	SAE-SS	SAE-MAS	SAR	IDEA
Best	2.72	2.79	2.81	2.71	2.71
Average	3.62	3.91	4.3	3.72	3.41
Worst	5.03	8.95	10.71	5.93	5.24

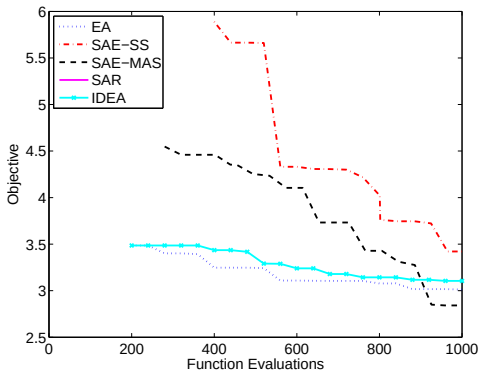


Figure 6.4: Average convergence of EA, SAE-EA and SAR-EA for compression spring design

6.5 Speed Reducer Design

A speed reducer configuration and a typical gear are shown in Figure 6.5. 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 [2].

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,

$$d_{p_{(i)}} = d_{(i)} + 2e_{(i)}, \quad e_{(i)} = 0.7 d_{(i)},$$

$$d_{st_{(i)}} = m z_{(i)} - 2.4m, \quad d_{w_{(i)}} = d_{st_{(i)}} - 4m.$$

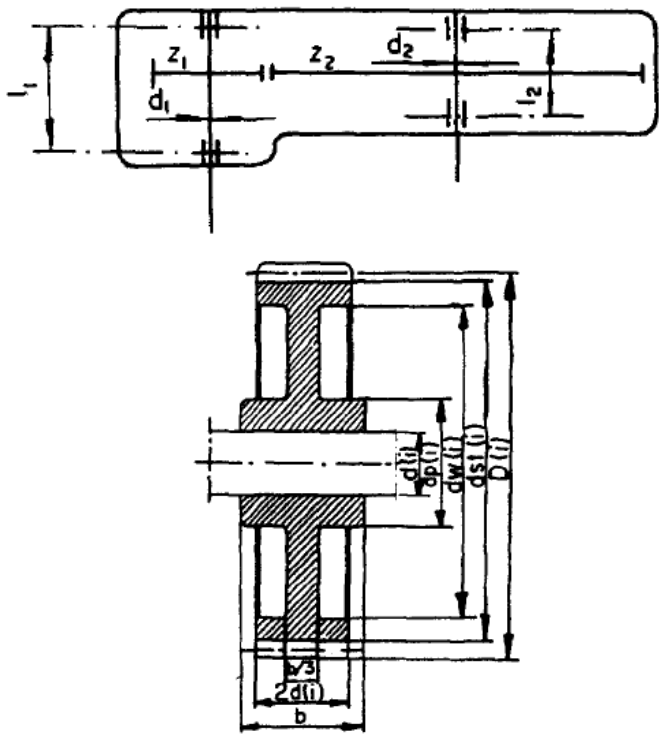


Figure 6.5: Speed Reducer and typical gear (Source: Golinski (1970) [2])

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 6.5.

Table 6.5: 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

The results of the speed reducer design using EA, SAE-EA, SAR-EA and IDEA are listed in Table 6.6. All the designs obtained using SAE-EA and SAR-EA have converged very close. The spread of the solutions is within 0.1% of the average objective value for SAE-EA and SAR-EA, whereas the spread of solutions for EA is more than 7% of the average objective value. The spatial surrogates are able to create very good approximations for the objective and the constraints, which result in a faster convergence to the best value as seen in Figure 6.6. Although IDEA converges slower than SAE-EA and SAR-EA, the performance of IDEA is better than EA as seen from the best, the average and the worst objective values in Table 6.6.

6.6 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 6.6: Results of speed reducer design using EA, SAE-EA, SAR-EA and IDEA

	EA	SAE-SS	SAE-MAS	SAR	IDEA
Best	3002.73	2994.43	2994.67	2994.39	2999.6
Average	3060.56	2995.93	2995.41	2994.42	3017.82
Worst	3234.31	2999.16	2996.14	2994.46	3037.23

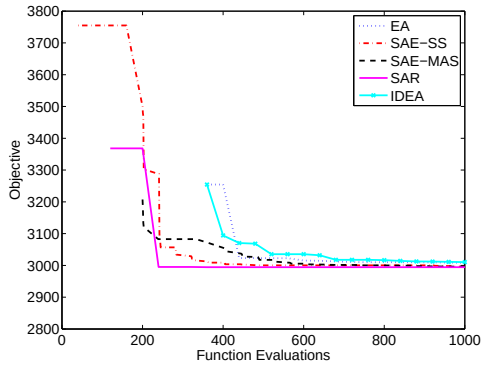


Figure 6.6: Convergence plots for speed reducer design

load [160]. 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 6.7.

The mathematical formulation of the optimization problem with five constraints is as follows:

$$\begin{aligned} \text{Minimize} \quad & f = 1.10471h^2l + 0.04811tb(14 + l) \\ \text{Subject to} \quad & g_1 = 0.25 - \delta \geq 0 \\ & g_2 = 13600 - \tau \geq 0 \\ & g_3 = 30000 - \sigma \geq 0 \\ & g_4 = b - h \geq 0 \\ & g_5 = P_c - 6000 \geq 0 \end{aligned}$$

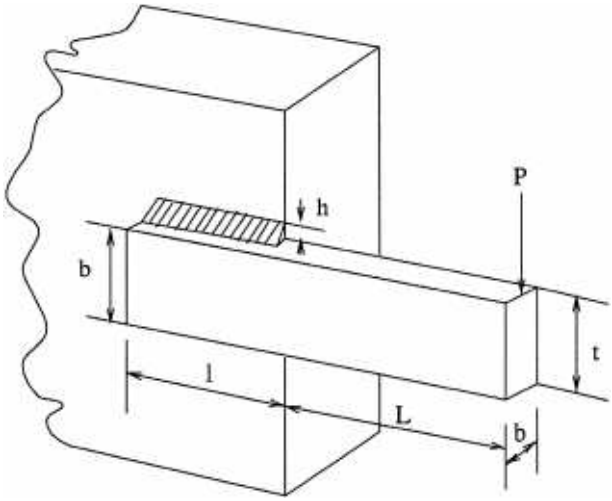


Figure 6.7: Welded-Beam Problem Configuration

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 $0.1 \leq b \leq 10.0$.

The designs obtained using EA, SAE-EA, SAR-EA and IDEA are listed in Table 6.7. The best design, obtained using SAE-EA, represents a cost of 2.44 which is slightly better than 2.46 obtained using EA. The average cost of the

welded beam obtained using SAE-EA and SAR-EA is better than that of EA. For this problem, IDEA has better convergence among all algorithms as seen from the small difference in the best and the worst objective values and the average improvement in the cost by 17%. The average convergence trends are shown in Figure 6.8. Even though the designs in the initial generations of SAE-EA and SAR-EA have high cost, the solutions converge quickly.

Table 6.7: Results of welded beam design using EA, SAE-EA, SAR-EA and IDEA

	EA	SAE-SS	SAE-MAS	SAR	IDEA
Best	2.46	2.44	2.45	2.57	2.51
Average	3.59	3.53	3.48	3.36	2.96
Worst	5.09	4.77	4.39	4.38	3.52

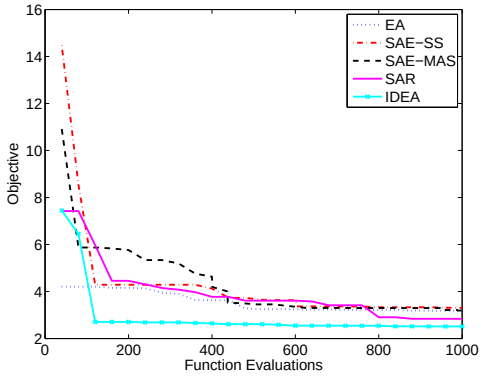


Figure 6.8: Optimization of welded beam

6.7 Design of a Pressure Vessel

A cylindrical vessel is capped at both ends by hemispherical heads as shown in Figure 6.9. The objective is to minimize the total cost, including the cost of material, forming and welding [161]. 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.

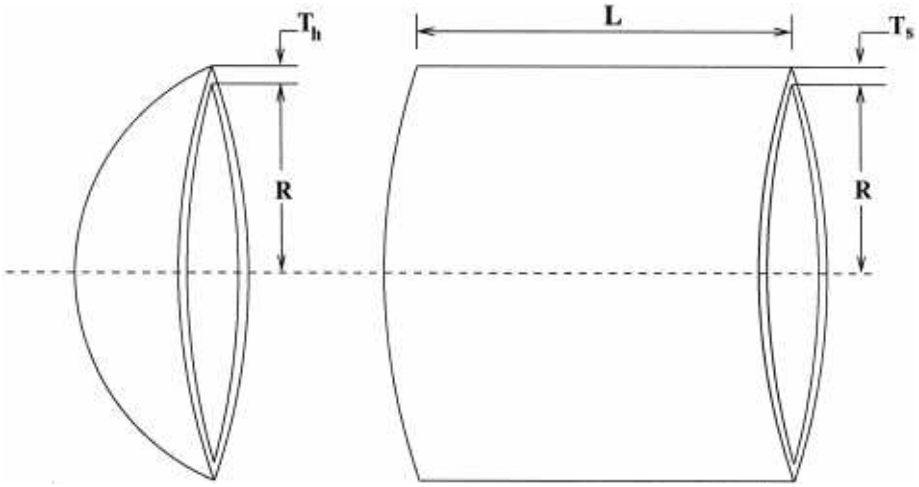


Figure 6.9: Center and End section of the pressure vessel

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 $10 \leq L, R \leq 200$. The results of the pressure vessel design using all the algorithms are

listed in Table 6.8. It can be seen that the surrogate assisted EAs perform poorly for this problem. The thicknesses T_h and T_s are discrete variables in this problem and the spatial surrogate models are not able to approximate the responses with discrete sampling of thickness variables. Even though the average performance of IDEA is worse than that of EA, the best design is obtained using IDEA.

Table 6.8: Results of pressure vessel design using EA, SAE-EA, SAR-EA and IDEA

	EA	SAE-SS	SAE-MAS	SAR	IDEA
Best	6194.01	6792.78	6305.11	7598.75	6119.97
Average	7150.93	9581.23	8521.01	10627.23	7415.53
Worst	7770.53	12680.5	12680.5	12847	9204.8

6.8 Airfoil Design

The airfoil shape is represented using a PARSEC formulation [162] as shown in Figure 6.10. 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}).

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

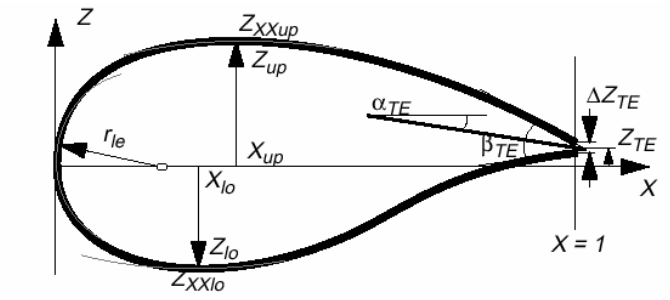


Figure 6.10: PARSEC representation for 2-D airfoil

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 [163]. 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 6.9.

The results of the airfoil optimization using EA, SAE-EA, SAR-EA and IDEA are listed in Table 6.10. The objective and the constraint functions are highly non-linear functions of the geometry variables and spatial surrogates are able to approximate these functions reasonably well as seen from the better performance of SAE-EA and SAR-EA. The best and the worst values of drag coefficient

Table 6.9: 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

obtained using SAE-EA and SAR-EA are lower than that of EA with the average improvement of 10–20%. For the airfoil design, even small improvements in the drag coefficient can result in significant benefits in the airfoil application. The optimum design in this problem is a constrained design and IDEA has the best convergence among all the algorithms as seen from the small difference between the best and the worst objective values. The average improvement in the airfoil designs using IDEA is more than 30%. The average convergence of SAE-EA, SAR-EA and IDEA is significantly better than EA as seen from Figure 6.11.

Table 6.10: Results of airfoil design using EA, SAE-EA, SAR-EA and IDEA

	EA	SAE-SS	SAE-MAS	SAR	IDEA
Best	2.0287e-3	1.9585e-3	1.9303e-3	1.9633e-3	1.9365e-3
Average	3.2731e-3	2.3696e-3	2.4785e-3	2.2650e-3	2.0550e-3
Worst	5.9429e-3	3.8730e-3	5.2073e-3	3.0907e-3	2.2374e-3

The airfoil design problem is also posed as a bi-objective optimization problem with an additional objective the pitching moment coefficient (C_m) being minimized. The non-dominated solutions obtained across all the runs using EA,

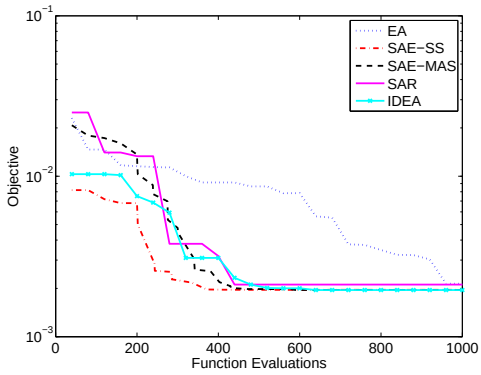


Figure 6.11: Convergence plots for airfoil design

SAE-EA, SAR-EA and IDEA are shown in Figure 6.12. The non-dominated solutions obtained using SAE-EA with multiple adaptive surrogates dominate the solutions obtained using other algorithms. The non-dominated solutions of IDEA dominate the solutions obtained using EA. As the Pareto optimal solutions are not known for this problem, the non-dominated solutions from the solutions of all the algorithms across all the runs are used as a reference and the displacement metric is calculated for each algorithm. The summary of the displacement metrics for EA, SAE-EA, SAR-EA and IDEA are presented in Table 6.11. The best non-dominated solutions are obtained using IDEA with the smallest value for the displacement metric. The average performance of SAE-EA with multiple adaptive surrogates and IDEA is better than EA with lower values of the displacement metric.

Table 6.11: Summary of Displacement metric for airfoil design using EA, SAE-EA, SAR-EA and IDEA

	EA	SAE-SS	SAE-MAS	SAR	IDEA
Minimum	1.1730e-5	4.8999e-5	1.5096e-5	2.1825e-5	9.9655e-6
Average	3.5279e-5	7.3890e-5	3.3321e-5	4.0438e-5	2.8876e-5
Maximum	6.2441e-5	1.8163e-4	5.1825e-5	6.2517e-5	6.4632e-5

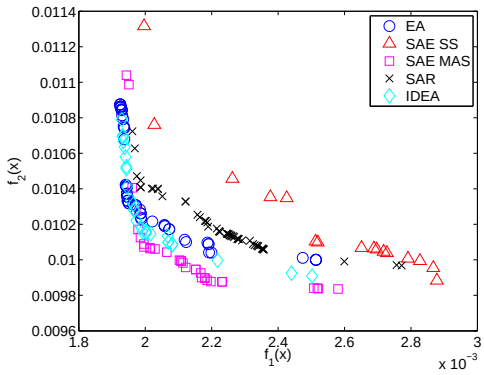


Figure 6.12: Non-dominates solutions obtained for airfoil design using EA, SAE-EA, SAR-EA and IDEA

6.9 Summary

Surrogate assisted EAs (SAE-EA and SAR-EA) and IDEA are tested on a set of engineering problems and the results are compared with EA for a fixed number of function evaluations across ten independent runs. For each engineering problem, better performing algorithms for the best, the average and the worst objective values are listed in Table 6.12. If more than one algorithm have similar objective values, all are listed. The summary of results indicates that IDEA outperforms the other algorithms in half the number of cases, and SAE-EA and SAR-EA in four cases each. Real-life design problems are driven by constraints and improved constraint handling in IDEA achieves better results as evident from the results.

Spatial surrogate assisted algorithms, SAE-EA and SAR-EA, achieve better results as compared to EA for Belleville spring design, speed reducer design, welded beam design and airfoil design problems. There is 5–20% improvement in the average objective values for these problems. The spatial surrogates have difficulty in approximating the responses in pressure vessel design problem, which has discrete design variables. In case of bi-objective airfoil design problem,

Table 6.12: Summary of algorithms with the best performance on engineering problems

	Best	Average	Worst
Belleville spring	IDEA	SAE-SS	IDEA
Compression spring	SAR/IDEA	IDEA	EA
Speed reducer	SAE-SS/SAR	SAR	SAR
Welded beam	SAE-SS	IDEA	IDEA
Pressure vessel	IDEA	EA	EA
Airfoil	SAE-MAS	IDEA	IDEA

SAE-EA with multiple adaptive surrogates and IDEA are able to find better non-dominated solutions as compared to EA.

Chapter 7

Truss Design

7.1 Overview

The design of discrete structures such as trusses involves sizing and topology optimization in the presence of applied loads, and stress and displacement constraints. For truss structures, sizing optimization refers to finding the optimum cross-sectional areas of the truss elements, keeping the connectivity of truss elements fixed. Topology optimization deals with finding the optimum connectivity of the truss elements. A truss design problem is often represented as a minimum weight design problem as described in Section 7.2. Topology optimization is characterized by the changes in the number of structural elements and the connectivity between those elements. Typically changes in topology are represented as Boolean variables (1 to denote presence of a truss element and 0 for absence) in the optimization problem. Often, the truss elements are designed using pipes of available sizes, which makes the cross-sectional areas of the truss elements discrete variables. Evolutionary algorithms (EAs) can easily handle Boolean and discrete variables making them ideally suited to solve truss design

problems. A review of the existing truss design methods using EAs is presented in Section 7.3.

Structural design uses finite element analysis (FEA) which can take a few seconds to a few minutes or even hours for complex structures. Since, EAs require evaluations of large number of designs to search for the optimum design, structural design using EAs can be computationally quite expensive. Use of surrogate models in lieu of FEA is complicated by the changes in the number of elements during topology optimization of structures. On the other hand sizing optimization can employ surrogate models to reduce the number of FEA.

In this chapter a Discrete Structures Optimization (DSO) algorithm is proposed for sizing and topology of the discrete structures. In DSO, topology optimization and sizing optimization are handled separately in two phases. In the first phase, the optimum topology is identified by removing the elements of the structure which are not load-bearing. The second phase is sizing optimization which uses EA for optimization. The discrete structures optimization algorithm is presented in Section 7.4. The use of spatial surrogates in the second phase constitutes proposed Surrogate-assisted Discrete Structures Optimization (SDSO), which is presented in Section 7.4.3. The algorithms are tested on a number of truss design examples in Section 7.5. The summary of the results is presented in Section 7.6.

7.2 Optimization Problem Statement

The minimum weight design optimization problem for a truss structure with m nodes and n elements is expressed as

$$\begin{aligned} \text{Minimize} \quad & W = \sum_{i=1}^{n} \rho_i \cdot A_i \cdot L_i \\ \text{subject to} \quad & \sigma_{min} \leq \sigma_i \leq \sigma_{max} \quad i = 1, \ldots, n \\ & \delta_{min} \leq \delta_i \leq \delta_{max} \quad i = 1, \ldots, m \\ & A_{min} \leq A_i \leq A_{max} \quad i = 1, \ldots, n_g \end{aligned}$$

where W is the weight of the structure, A_i is the cross-sectional area, L_i is the length and ρ_i is the material density for ith element. The areas of the elements are bounded in the range A_{min} and A_{max}. For symmetric truss structures the similar elements are grouped in n_g groups and the same cross-section area limits are applied. The design is usually constrained by the stress limits and/or displacement limits. The stress limits are given by σ_{min} for stress in compression (-ve) and σ_{max} for stress in tension. The displacements limits are given by δ_{min} and δ_{max} for deflections in the -ve and +ve axial directions.

7.3 Topology Optimization Using EA

The early use of EAs in truss design was for the sizing optimization [164, 165, 166]. Later, EAs were used for solving simultaneous sizing and topology optimization problems [167, 168, 169] and combined sizing, shape and topology optimization of trusses [170, 171].

In topology optimization of truss structures, the elements and their connec-

tivity is identified. In the optimization process, the elements can be added to or removed from the structure. This aspect of topology optimization requires special representation schemes to describe the truss design using a fixed set of design variables. In addition, the cross-sectional areas of truss elements are often selected from a set of possible areas. This makes the design variables corresponding to the cross-sectional areas discrete and they also need a representation scheme.

Many researchers have applied evolutionary algorithms to solve the truss design problem using various representation schemes for describing topology and the cross-sectional areas. Wu and Chow [166], Kaveh and Kalatjari [168], and Coello [165] have all used binary-coded variables to represent the cross-sectional areas. In binary coding, discrete values are represented by n bits (each bit taking a value of 0 or 1), giving rise to 2^n possible combinations. Discrete values that are not powers of 2 cannot be represented exactly. In [165], there are 42 discrete values of cross-sectional areas, binary coded using 6-bits. This corresponds to a total of 64 combinations with the first 42 combinations representing prescribed cross-section area values and the remaining 22 combinations are randomly assigned to the discrete values. Thus, some of the area values are repeated. In the case of Ghasemi, Hinton and Wood [172], the remaining 22 values repeat the first 22 values. Kaveh and Kalatjari [168] use an additional bit per truss member to represent the presence (1) or absence (0) of the truss members in the structure.

Real-coded variables are used by Deb [173, 167] to represent the cross-sectional area. Negative values of the cross-sectional areas are used to represent the absence of that truss member. For discrete values of cross-sectional area, discrete versions of simulated binary crossover (SBX) and mutation operators are used to ensure that recombination and mutation generate only discrete values from the prescribed set of values. Turkkan [174] uses the real-coding of the area variables

using the real recombination and mutation operators. The area computed from recombination and mutation is mapped using the integer part of that value into an array of discrete values.

Tang, Tong and Gu [171] have used mixed representation, integer coding to represent the discrete values of the cross-sectional area and binary-coding to represent the topological variables (one for each of the truss members). They also propose recombination and mutation operators to work with the integer coding. Topology is represented using additional binary-coded variables. Rajeev and Krishnamoorthy [170] have used variable length representation in Variable String Length Genetic Algorithm (VGA) to represent different topologies for truss design. They use the cut-and-splice operator as in messy genetic algorithm [175] to generate offspring with differing chromosome length from their parents.

Xie and Steven [176] introduced Evolutionary Structural Optimization (ESO) technique in 1993. It is based on the idea that the optimal structure (which is a fully stressed design) can be produced by gradually removing material from the design domain. The technique has also been applied to simultaneous sizing and topology optimization of discrete structures [177]. In ESO for trusses, the area of the truss elements is reduced in very small steps (of the order of (original area)/1000 or less) provided that the stress in a truss element is below the target. This process requires a lot of small steps to reach the optimum structure and consequently a large number of FEA evaluations.

The original ESO technique is applicable to problems with stress considerations only and do not consider stiffness constraints in the form of displacement constraints. To alleviate this problem Chu *et al.* [178] proposed the use of sensitivity numbers, which indicate the change in overall stiffness (in the form of strain energy) to remove the elements. The structural elements with the least

strain energy are removed from the structure, thus achieving a faster convergence to optimum topology in the case of continuous structures. Based on the ESO approach, Tanskanen [179] has proposed a multi-objective modification to ESO with modified objective as a function of compliance volume and strain energy, and utilized the approximations to the gradient of the objective with respect to the design variables. This method can only be used for continuous design variables. Similar methods involving material removal for truss design are presented in [180, 181]. These methods are used for the truss design with discrete design variables.

7.4 Discrete Structures Optimization

Discrete Structures Optimization (DSO) is derived from the ESO technique by Xie and Steven [176]. In DSO, material is removed based on the least strain energy rather than the least amount of stress. The strain energy E_i for ith truss element is given by,

$$E_i = \frac{1}{2}\{u_i\}^T[K_i]\{u_i\},$$

where $\{u_i\}$ is the nodal displacement vector of the ith element and $[K_i]$ is the element stiffness matrix. Use of strain energy instead of stress allows for rapid material removal for the structural elements that do not carry any of the structural loads. Once the structural elements have reached the limit of material removal, they are deleted. Discrete Structures Optimization is a two stage process unlike ESO. In the first stage, the optimum topology of the structure is identified. The second stage of DSO is the sizing optimization using the optimum topology obtained in the first stage.

7.4.1 First Stage: Topology Optimization

The steps involved in the first stage of DSO are described as follows.

1. Start with the ground structure with all the elements initialized to the upper limit of material (*i.e.* maximum area possible).
2. Perform FEA to calculate stresses, displacements and strain energy.
3. If the structure is infeasible for the initial structure, there is no possible solution to the given problem. Stop.
4. If the structure is feasible, find the element with the least amount of strain energy that is less than the threshold. Remove material from that element. Repeat from step 2.
5. If the element has reached lower limit for the material, freeze the element.
6. If the structure becomes infeasible, revert the material back (to the element from which it was removed) and freeze the element.
7. If there are no more elements with the strain energy less than the threshold, Stop. If not, repeat from step 2.

Once the structural elements are frozen, they are not considered for material removal. At the end of the first stage, all the frozen elements that are at the lowest limit of the material are removed from the structure to form the optimum topology of the structure. The algorithm requires a user defined parameter, *strain energy threshold*, to identify an element from which material is removed. The strain energy threshold depends on the structure. Typically the strain energy for the structurally redundant element is found to be much less than the average

strain energy of the elements. For a truss structure with 10 elements, the average strain energy is 10% of the total strain energy and a strain energy threshold of 1% of the total strain energy would be adequate to remove the redundant elements.

For faster removal of material or reducing the cross-sectional area (in step 4), a polynomial mutation operator [38] is used as given in Equation 7.1.

$$
\begin{aligned}
y &= x + (\overline{x} - \underline{x})\,\bar{\delta} \\
\bar{\delta} &= \begin{cases} (2r)^{1/(\eta_m+1)} - 1, & \text{if } r < 0.5, \\ 1 - [2(1-r)]^{1/(\eta_m+1)}, & \text{if } r \geq 0.5. \end{cases}
\end{aligned} \tag{7.1}
$$

For $r < 0.5$, $\bar{\delta}$ is negative and y value is smaller than x value. For $r \geq 0.5$, $\bar{\delta}$ is positive and y value is larger than x. To reduce the area x, a random number is generated between 0 and 0.5 and used as the value for r. The polynomial mutation operator constructs a density function between the lower limit $\underline{x}$ and specified x peaking at x. The shape of the density function is controlled by mutation distribution index η_m. Higher the value of η_m, peakier is the density function and lower the value of η_m, flatter is the density function. The use of the mutation operator avoids the need to specify the area decrements absolutely as in the case of ESO [177] and large steps can be considered for material removal.

7.4.2 Second Stage: Sizing Optimization

The topology of the structure is fixed after the first stage of DSO. As the number of truss elements are fixed, so is the number of design variables. The second stage of DSO is sizing optimization to obtain the minimum weight structure by varying the cross-sectional areas of truss elements. This problem can be posed as a standard optimization problem with fixed number of design variables and

solved as such. Since the element areas can be discrete, an evolutionary algorithm is used for sizing optimization.

7.4.3 Surrogate assisted DSO

Surrogate assisted Discrete Structures Optimization (SDSO) is an extension to DSO where spatial surrogates are used in the sizing optimization stage. Similar to DSO, SDSO is also a two stage process of topology optimization and sizing optimization. The first stage of SDSO, topology optimization, is the same as that of DSO. In the second stage of sizing optimization, surrogate assisted EA is used instead of EA. In this study, SAE-EA with single spatial surrogate model is used for sizing optimization in SDSO.

7.5 Results of Truss Design

Both the proposed algorithms (DSO and SDSO) are tested on 2-D and 3-D structures. Two-dimensional truss structures include ten-bar and seventeen-bar cantilever trusses. Three-dimensional structures include twenty-two-bar space truss and twenty-five-bar transmission tower.

The strain energy criterion for element removal is 1% of the total strain energy (which is sufficiently less than the average strain energy) for all the problems. For the sizing optimization phase, the number of FEA is fixed at 1,000. The population size of 40 is used for EA in DSO and SAE-EA in SDSO. Each truss design example is solved using ten different values for random seeds. The rest of the EA parameters are fixed. For SAE-EA, a single spatial surrogate model is trained using RSM, RBF and Kriging. The prediction accuracy threshold is set to 5% and training interval is set to five. The maximum number of training

samples used for training is 500.

7.5.1 Ten-bar 2D cantilever truss

A ground structure for a six-node, ten-bar 2-D cantilever truss is shown in Figure 7.1. The problem is to design the minimum weight truss structure using sizing and topology optimization subject to stress and displacement constraints. The truss is elastic with the modulus of elasticity E=10^7 lb/in^2, and the density 0.1 lb/in^3. The maximum allowed stress in compression or tension for all elements is 25,000 lb/in^2. The maximum vertical displacement allowed is 2 in. Two different loading conditions are considered in this example, namely, case 1, where $P_1 = 100$ kips and $P_2 = 0$, and case 2, where $P_1 = 150$ kips and $P_2 = 50$ kips. The areas of the truss elements are allowed to vary in the continuous range $[1, 35]$ in^2 for both cases. In addition to continuous area variation, discrete area variation is considered for Case 1. The area values are taken from the *American Institute of Steel Construction Manual* [170].

Area = (1.62, 1.80, 1.99, 2.13, 2.38, 2.62, 2.63, 2.88, 2.93, 3.09, 3.13, 3.38, 3.47, 3.55, 3,63, 3,84, 3.87, 3.88, 4.18, 4.22, 4.49, 4.59, 5.12, 5.74, 7.22, 7.97, 11.50, 13.50, 13.90, 14.20, 15.50, 16.00, 16.90, 18.80, 19.90, 22.00, 22.90, 26.50, 30.00, 33.50). All the values are in square inches.

In the first stage, the areas are reduced systematically for the elements using the least amount of strain energy in each generation. At the end of the first stage, elements 2, 5, 6, and 10 are eliminated and the remaining elements 1, 3, 4, 7, 8, and 9 form the optimum topology for Case 1. The variation of area and strain energy across generations for the elements 2, 5, 6, 10 for Case 1 is shown in Figure 7.2. The strain energy is shown as a fraction of total strain energy. It is seen that the strain energy for the elements falls well below 1% of the total

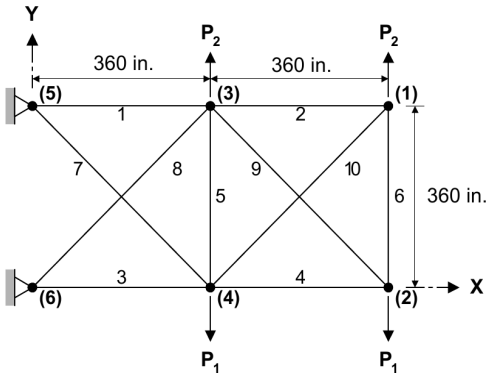
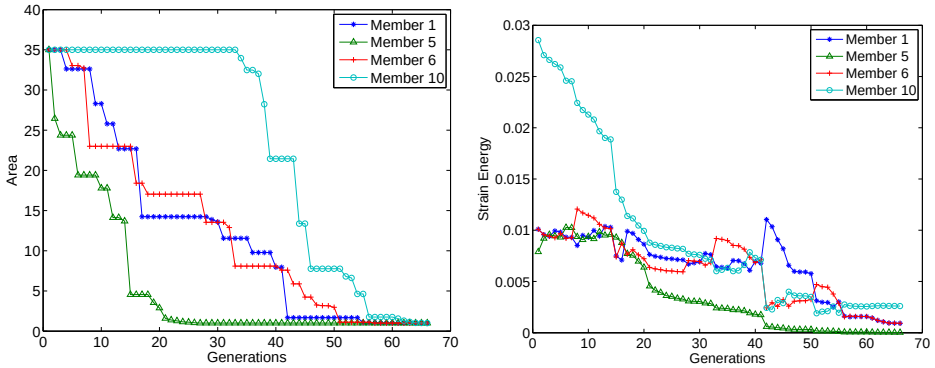


Figure 7.1: Ground structure for 6-Node, 10-Member 2-D Truss

strain energy by generation 45. Over the next 20 generations the areas of these elements are reduced to the minimum possible value. For Case 2, elements 2 and 10 are eliminated and elements 1, 3, 4, 5, 6, 7, 8, and 9 form the optimum topology. The number of evaluations required for the topology optimization for both the cases are listed in Table 7.1.



(a) Area (b) Strain Energy

Figure 7.2: Area and strain energy fraction variation with generations for 10-bar 2D truss (Case 1)

The summary of results of the sizing optimization using DSO and SDSO are listed in Table 7.2. The truss weights obtained using SDSO are are significantly lower that those obtained by DSO. In addition, the weight obtained for Case

Table 7.1: Number of FEA for first stage of DSO and SDSO for 2D 10-bar truss

	Case 1 (Continuous)	Case 1 (Discrete)	Case 2
Minimum	82	67	60
Average	89	79	64
Maximum	97	89	67

1 with continuous area variation is very close to the best reported weight of 4899.15 lb by Deb and Gulati [167], which was obtained for a population size of 220 evolved over 225 generations corresponding to 49,500 FEA evaluations. The weight of 4928.17 lb was obtained with SDSO using only (82 + 1000 =) 1,082 FEA evaluations. Similarly, the best weight reported for Case 1 with discrete area variation is 4962.1 lb by Kaveh and Kalatjari [168] using more than 15,000 FEA evaluations. A truss weight of 4981.18 lb was obtained with SDSO using only (67 + 1000 =) 1,067 FEA evaluations. These results indicate that the truss designs obtained by SDSO for Case 1 have weights within 6% and 1% of the best designs reported in the literature using only 3% and 7% of the function evaluations, which is a considerable saving. For Case 2, the best reported weight is 4668.81 lb by Lee and Geem [182] using 15,000 evaluations, whereas the weight obtained using SDSO, 4988.4 lb, is within 7% of the best reported weight with only 1,060 evaluations, which are 7% of the 15,000 evaluations used by Lee and Geem. The cross-sectional areas of the best designs obtained using SDSO are listed in Table 7.3.

7.5.2 Seventeen-bar 2-D cantilever truss

A ground structure for a nine-node, seventeen-bar 2-D cantilever truss is shown in Figure 7.3. The structure is elastic with the modulus of elasticity E=30,000

Table 7.2: Summary of 2D 10-bar truss results using DSO and SDSO

		Case 1 (Continuous)	Case 1 (Discrete)	Case 2
DSO	Best	5094.38	4981.18	7217.27
	Average	5959.01	5072.83	7900.28
	Worst	6510.48	5223.03	8548.51
SDSO	Best	4928.17	4981.18	4988.4
	Average	4976.97	5003.21	5186.31
	Worst	5085.68	5046.54	5781.38

Table 7.3: Best design for 2D 10-bar cantilever truss using SDSO

Element	Case 1 (Continuous) Area (in^2)	Case 1 (Discrete) Area (in^2)	Case 2 Area (in^2)
1	27.29	30.00	26.40
3	22.62	22.90	22.84
4	15.93	16.00	13.94
5	-	-	1.06
6	-	-	2.20
7	6.06	7.22	10.12
8	20.61	22.00	15.30
9	23.58	19.90	25.59
Weight (lb)	4928.17	4981.18	4988.4

ksi, and the density of 0.268 lb/in^3. The maximum allowed stress in compression or tension for all elements is 50 ksi and the maximum displacement of the nodes in both directions (x and y) is 2 in. A single load of 100 kips is applied vertically downward at node 9. The minimum area for the truss elements is 0.1 in^2.

In the topology optimization stage, six elements (2, 4, 8, 10, 12, and 16) are eliminated and the optimum topology consists of eleven elements. The minimum and the maximum number of FEA required in the first stage are 146 and 174 respectively. The summary of results for 17-member 2D truss are listed in Table 7.4. The truss weights obtained using SDSO are much smaller than

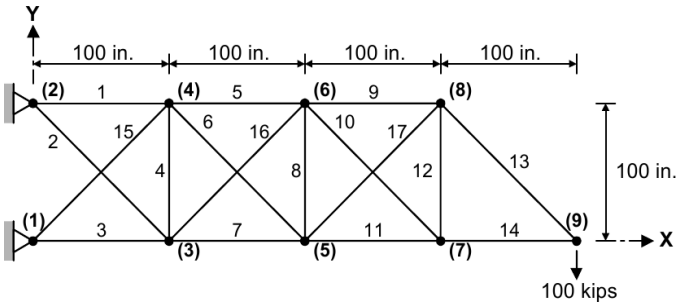


Figure 7.3: Ground structure for 9-Node, 17-Member 2-D truss

DSO. The minimum truss weight of 2671.39 lb is obtained after (146 + 1000 =) 1,146 FEA evaluations. The best design reported in the literature is by Lee and Geem [182] with the weight of of 2580.81 lb after 20,000 evaluations. The result indicates that using only 5% of the function evaluations, truss structure with the weight of within 4% of the best design was obtained using SDSO. The member areas for the best design obtained using SDSO are listed in Table 7.5.

Table 7.4: Summary of 2D 17-member truss results using DSO and SDSO

	DSO	SDSO
Best	4022.06	2671.39
Average	4688.89	2785.43
Worst	5148.63	3018.39

7.5.3 Twenty-two-bar space truss

A ground structure for a eight-node, twenty-two-bar space truss is shown in Figure 7.4. The modulus of elasticity and the material density for the elements are 10,000 kpsi and 0.1 lb/in^3 respectively. The element areas are linked into seven groups – (1) $A_1 \sim A_4$, (2) $A_5 \sim A_6$, (3) $A_7 \sim A_8$, (4) $A_9 \sim A_{10}$, (5) $A_{11} \sim A_{14}$, (6) $A_{15} \sim A_{18}$, and (7) $A_{19} \sim A_{22}$. Within each group the areas of the

Table 7.5: Best design for 2D 17-bar cantilever truss using SDSO

Element	Area (in^2)
1	15.32
3	10.12
5	6.45
6	6.61
7	13.35
9	8.64
11	4.98
13	8.56
14	4.91
15	5.75
17	4.46
Weight (lb)	2671.39

elements are equal. The truss is loaded with three load cases listed in Table 7.6. For each load case, the maximum displacement in any direction is 2 in. The maximum allowed stresses in compression and tension for each group of elements are as given in Table 7.7. The minimum cross-sectional area for all the elements is 0.1 in^2.

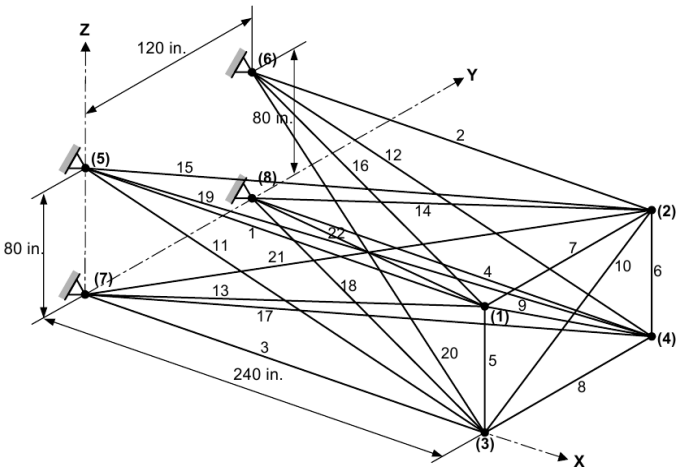


Figure 7.4: Ground structure for 8-Node, 22-Member space truss

Table 7.6: Load cases for 22-bar space truss

Load Case	Node #	F_x (kips)	F_y (kips)	F_z (kips)
1	1	-20.0	0.0	-5.0
	2	-20.0	0.0	-5.0
	3	-20.0	0.0	-30.0
	4	-20.0	0.0	-30.0
2	1	-20.0	-5.0	0.0
	2	-20.0	-50.0	0.0
	3	-20.0	-5.0	0.0
	4	-20.0	-50.0	0.0
3	1	-20.0	0.0	35.0
	2	-20.0	0.0	0.0
	3	-20.0	0.0	0.0
	4	-20.0	0.0	-35.0

Table 7.7: Stress limitations for 22-bar space truss

Group #	Area Variables	Compressive Stress Limitation (kpsi)	Tension Stress Limitation (kpsi)
1	$A_1 \sim A_4$	24.0	36.0
2	$A_5 \sim A_6$	30.0	36.0
3	$A_7 \sim A_8$	28.0	36.0
4	$A_9 \sim A_{10}$	26.0	36.0
5	$A_{11} \sim A_{14}$	22.0	36.0
6	$A_{15} \sim A_{18}$	20.0	36.0
7	$A_{19} \sim A_{22}$	18.0	36.0

For this problem, when the strain energy of all the elements in a group is less than 1% of the total strain energy, the entire group of elements is deleted; otherwise, all the elements in the group are retained. The topology optimization resulted in the elimination of group 3 and corresponding two elements 7 and 8. The average number of FEA evaluations required for topology optimization is

20. The results of sizing optimization in DSO and SDSO are given in Table 7.8. The minimum weight of 1031.15 lb was obtained after (20 + 1000 =) 1,022 FEA evaluations. The best design reported in the literature by Lee and Geem [182] has weight of 1022.23 lb after 50,000 evaluations. This corresponds to 98% saving in the computations to find the truss structure with weight within 1% of the best reported design. The cross-sectional areas of six area groups for the best design obtained using SDSO are shown in Table 7.9.

Table 7.8: Summary of 22-member space truss results using DSO and SDSO

	DSO	SDSO
Best	1075.65	1031.15
Average	1102.27	1063.59
Worst	1130.85	1081.51

Table 7.9: Results of 22-member space truss using DSO

Group #	Area (in^2)
1	2.58
2	1.49
4	0.73
5	2.65
6	2.08
7	2.20
Weight (lb)	1031.15

7.5.4 Twenty-five-bar 3D transmission tower

A ground structure for 10-node, 25-bar transmission tower is shown in Figure 7.5. The elements of the tower are grouped in eight groups based on similar characteristics. Within each group, the elements have the same cross-sectional areas.

The groups are as follows – (1) A_1, (2) $A_2 \sim A_5$, (3) $A_6 \sim A_9$, (4) $A_{10} \sim A_{11}$, (5) $A_{12} \sim A_{13}$, (6) $A_{14} \sim A_{17}$, (7) $A_{18} \sim A_{21}$, and (8) $A_{22} \sim A_{25}$. The tower is subject to two independent loading conditions as given in Table 7.10. The maximum stress limit is 40,000 lb/in^2 for elements in tension. The stress limits for elements in compression are different for each group and are given in Table 7.11. The maximum displacement of each joint in any direction is limited to 0.35 in. The material density is 0.1 lb/in^3, and modulus of elasticity is 10,000 ksi.

The design problem is solved with continuous and discrete area variation. For continuous area variation, the cross-sectional areas are in the range $[0.005, 3]$ in^2. For the discrete area variation the area values are chosen from the following set. Area = (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.1, 2.2, 2.3, 2.4, 2.5, 2.6, 2.8, 3.0, 3.2, 3.4). All values are in square inches.

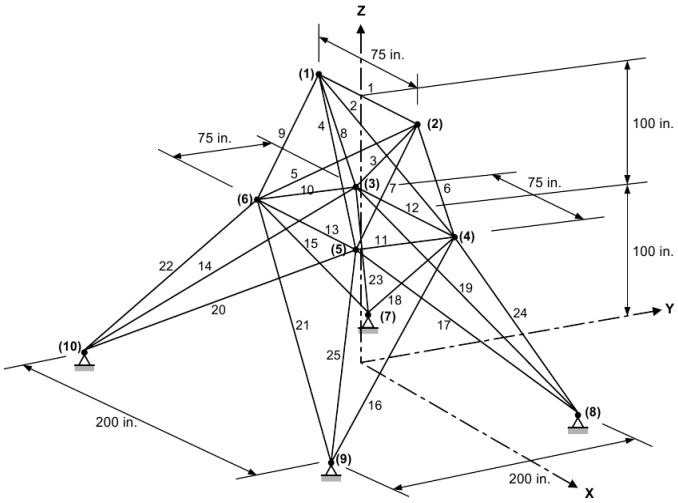


Figure 7.5: Ground structure for 10-Node, 25-Member 3-D Truss

In the topology optimization stage groups 1, 4 and 5 (and corresponding 5 elements 1, 10, 11, 12, and 13) are eliminated. For the transmission tower with continuous area variation, the average number of FEA evaluations in topology

Table 7.10: Load cases for 25-bar transmission tower

Load Case	Node #	F_x (lb)	F_y (lb)	F_z (lb)
1	1	1,000	10,000	-5,000
	2	0	10,000	-5,000
	3	500	0	0
	6	500	0	0
2	1	0	20,000	-5,000
	2	0	-20,000	-5,000

Table 7.11: Member stress limitations for 25-bar transmission tower

Group #	Area Variables	Compressive Stress Limitation (kpsi)	Tension Stress Limitation (kpsi)
1	A_1	35.092	40.0
2	$A_2 \sim A_5$	11.590	40.0
3	$A_6 \sim A_9$	17.305	40.0
4	$A_{10} \sim A_{11}$	35.092	40.0
5	$A_{12} \sim A_{13}$	35.092	40.0
6	$A_{14} \sim A_{17}$	6.759	40.0
7	$A_{18} \sim A_{21}$	6.959	40.0
8	$A_{22} \sim A_{25}$	11.082	40.0

optimization stage is 61 and for discrete area variation it is 53. The summary of the tower weights obtained using DSO and SDSO for continuous and discrete are variation are listed in Table 7.12. The minimum weight of the transmission tower obtained is 559.02 lb using DSO for continuous area variation and 557.04 lb using SDSO for discrete are variation. The average tower weights are smaller for SDSO as compared to DSO. The spatial surrogate models are built only in the last few generations and the benefits of SDSO are comparatively smaller in this example as compared to previous examples. The best design reported by Lee

and Geem [182] for the continuous area variation with the weight of 544.38 lb is obtained after 15,000 evaluations. Tower designs obtained with DSO and SDSO are with 1,061 and 1,053 FEA evaluations respectively. The results obtained by DSO and SDSO are within 4% of the best reported design using only 7% of corresponding function evaluations. The best designs for continuous and discrete area variation are listed in Table 7.13.

Table 7.12: Summary of 3-D transmission tower results using DSO and SDSO

	Continuous		Discrete	
	DSO	SDSO	DSO	SDSO
Best	559.02	563.67	571.4	557.04
Average	599.60	589.82	616.38	583.2
Worst	648.83	617.79	653.09	609.32

Table 7.13: Best designs for 3D transmission tower using DSO and SDSO

Group #	Continuous Area (in^2)	Discrete Area (in^2)
2	2.04	2.1
3	2.63	2.6
6	0.93	1.0
7	1.92	1.8
8	2.51	2.5
Weight (lb)	559.02	557.04

7.6 Summary

The design of discrete structures involves topology and sizing optimization. Topology optimization finds the optimum connectivity of the structural elements and

sizing optimization finds the sizes of the structural elements. Topology optimization requires special representation schemes to describe the connectivity using a fixed number of variables. The sizes of the structural elements are often not continuously varying but chosen from a set of discrete values. Since evolutionary algorithms can handle Boolean, integer and discrete variables, they have been used for simultaneous topology and sizing optimization. But the number of FEA evaluations required using EAs is usually quite large.

Discrete structures optimization algorithm based on ESO technique is proposed in this chapter. In DSO, the material is removed from the structural elements based on the least amount of strain energy. The use of strain energy instead of stress allows faster removal of material from the structure. In addition, a mutation operator is proposed to further speed up material removal, instead of prescribing a fixed rate. In DSO, topology optimization problem is solved first to identify the optimum topology. The second stage is sizing optimization to find the optimum sizes for structural elements. An EA is used to solve sizing optimization problem in second stage of DSO. An extension to DSO using spatial surrogates is also proposed as surrogate assisted DSO. In SDSO, spatial surrogate assisted EA is used for sizing optimization. The first stage of topology optimization in SDSO is the same as that of DSO.

The proposed algorithms are tested on 2-D and 3-D truss structures. The first stage results indicate that the optimum topologies are identified correctly for all the problems using very few FEA evaluations. For 2-D examples, the average number of FEA evaluations for topology optimization is 64–161 and for 3-D examples it is 20–61. Once the optimum topology is identified, the redundant structural elements are dropped and the subsequent sizing optimization problem is smaller than the original optimization problem. The designs obtained using

SDSO are much lighter than those obtained using DSO and fairly close to the best designs (within 1–7% for all the problem studied) reported in the literature. The best results reported for all the problems use tens of thousands of FEA evaluations, whereas the results obtained using DSO and SDSO are with 1,000 FEA evaluations for sizing and maximum of 174 evaluations for topology optimization. Thus a significant saving of 93–98% in the number of FEA evaluations is achieved by separating topology and sizing optimization problems, and using spatial surrogates to improve the convergence in sizing optimization.

Chapter 8

Conclusions

8.1 Research Summary and Outcomes

Evolutionary algorithms (EAs) are well suited as general global optimization methods owing to their ease of use and applicability to a wide range of single and multi-objective optimization problems. However, their use for engineering design can be difficult as evolutionary algorithms require evaluations of large numbers of candidate solutions and each evaluation in engineering design often requires a solution of computationally expensive simulation.

The research reported in this thesis is focused on improving the efficiency of evolutionary algorithms to obtain better designs for a fixed computational cost. One strategy is to use surrogate models in lieu of expensive simulations to evaluate the performance of a design. A generic framework, spatial surrogate modeling, is proposed for use of surrogate models within EA. This framework tries to address several shortcomings of the existing methods employed involving surrogate models. Multiple types of surrogate models are used instead of a single type to select the surrogate model that best approximates each of the objective

and the constraint functions. A spatial surrogate model mimics the behavior of a global surrogate model in the early stages of evolutionary search and a local surrogate model as the EA population converges. Spatial surrogate models are validated using a prediction accuracy criterion to ensure that the search based on the approximations is not misguided. Two EAs are proposed using spatial surrogate models for evaluation and evolution. In surrogate assisted evaluation (SAE-EA), the spatial surrogate models are periodically trained and used in place of the actual simulations. In surrogate assisted recombination (SAR-EA), the spatial surrogate models are used to perform a secondary evolutionary search and identify better offspring solutions within the broader EA scheme.

Most engineering problems are constrained design problems, but evolutionary algorithms do not have a native constraint handling mechanism. A simple approach commonly used is a penalty based formulation to convert a constrained optimization problem into an unconstrained optimization problem. This approach requires specification of penalty factors which are often problem dependent. The other approaches without use of penalty functions, favour feasible solutions over infeasible solutions and search the design space through feasible regions. The proposed infeasibility driven evolutionary algorithm (IDEA) retains a few infeasible solutions in the population and ranks them higher than feasible solutions to focus the search near the constraint boundaries through the infeasible as well as the feasible regions. Even though, only the feasible solutions are considered valid designs, the infeasible solutions in the population can be be used as trade-off solutions to study the effects of relaxing one or more constraints.

Two extensions to EA proposed in this research are tested on a number of benchmark problems and engineering examples. The results clearly indicate significant benefit of using spatial surrogate models. Various functions of up to 30

variables have been successfully approximated using the spatial surrogates with the maximum of 500 training solutions. Using SAR-EA, solutions very close to the optimum are found within 1,000 evaluations. For engineering examples, 5–20% improvements in the objective values are achieved using spatial surrogates over EA. The results of IDEA show faster convergence for single and multi-objective constrained optimization problems. Real life engineering design problems are constrained optimization problems and the advantages of IDEA are apparent from the results of the engineering examples documented in the thesis.

In the design of discrete structures, one of the goals is to identify the structural elements (and their connectivity) to form the optimum topology. Topology optimization requires capability to handle varying number of structural elements and EAs have been successfully used as they can handle discrete variables, which are often used to represent changes in the structure. As the spatial surrogate models cannot be used to represent functions of varying number of design variables, discrete structure optimization (DSO) is proposed to separate optimum topology identification and sizing optimization. Strain energy based material removal criterion is used to eliminate the elements that do not carry any load, thus identifying the optimum topology. Once the optimum topology is identified, sizing optimization problem has a fixed number of design variables and is solved using EA. Surrogate assisted DSO (SDSO), an extension to DSO uses spatial surrogate models to improve the convergence of sizing optimization. In the studies on 2-D and 3-D truss structures, truss designs obtained using DSO and SDSO have weights within 1–7% of the best reported in the literature and in the process require only 2–7% of the function evaluations, which is a significant saving in the computational cost.

The results obtained using spatial surrogates and infeasibility driven con-

straint handling are promising and demonstrate the potential of the proposed approaches to improve the efficiency of evolutionary algorithms for engineering design.

8.2 Achievements

To summarize, the achievements reported in the thesis are clustered around four concepts as follows.

1. A spatial surrogate modeling framework is proposed to replace expensive analysis with computationally cheap surrogate models in the context of evolutionary algorithms. Spatial surrogate models can have a single surrogate model or multiple spatially distributed surrogate models. In the case of multiple surrogate models, the number of surrogate models can be determined adaptively using a prediction accuracy based criterion. Spatial surrogate models are able to approximate various functions of up to 30 variables using RSM and RBF.

2. Two evolutionary algorithms are presented using the proposed spatial surrogate models.

   (a) The first is surrogate assisted evaluation (SAE-EA), where the spatial surrogates are used in lieu of expensive simulations to evaluate the objectives and the constraints. Using SAE-EA, the population migrates quickly to the region of better solutions, improving the convergence of evolutionary search.

   (b) The second is surrogate assisted recombination (SAR-EA), where the spatial surrogates are used in each generation to perform an evolu-

tionary search to quickly find the best solutions as per the surrogate models. The exploitation of surrogate models in SAR-EA can result in solutions very close to the optimum within a few function evaluations.

3. An infeasibility driven evolutionary algorithm (IDEA) with a novel constraint handling method is proposed and developed. In IDEA, the original m objective optimization problem is transformed into $m+1$ objective optimization problem, where the additional objective is the constraint violation measure. Two constraint violation measures are proposed.

   (a) A constraint violation count measure is used to solve a constrained optimization problem along with the unconstrained variant of the problem formulated by dropping all the constraints. With this approach the Pareto front for the modified unconstrained optimization problem is discovered.

   (b) A constraint relative rank measure is used to rank infeasible solutions with relatively less constraint violation higher. With this measure IDEA results in marginally infeasible solutions near the constraint boundaries. These solutions then can be used for trade-off studies.

4. Sizing and topology optimization methods based on evolutionary structural optimization are proposed and developed for the design of discrete structures. The proposed methods include –

   (a) Discrete structures optimization (DSO), where topology optimization and sizing optimization problems are solved separately to speed up the identification of the optimum topology.

   (b) Surrogate assisted discrete structures optimization (SDSO), which is

an extension of DSO using SAE-EA for sizing optimization.

Significant computational savings of 93-98% are obtained using SDSO to find the truss designs very close to the best reported in literature.

All the proposed algorithms (SAE-EA, SAR-EA, IDEA, DSO and SDSO) are implemented under a unified evolutionary algorithm framework. The unified framework has been developed in Matlab. The framework is coded in objective oriented style, consists of 16 classes and more than 5000 lines of Matlab code. The framework is currently being used by the multidisciplinary design optimization (MDO) group in UNSW@ADFA for fast craft design and design of flapping wings. The framework is also being used by the Hypersonics group at the University of Queensland for the design of a hypersonic vehicle. The concept of IDEA has been used to solve constrained dynamic optimization problems and optimization problems involving many objectives.

During the course of the research, a total of 22 articles have been published in various journals, books and conference proceedings (see List of Publications at the beginning of thesis).

8.3 Future Areas of Research

It is clear from the engineering examples, that for real life problem constraint handling is as important as improving the efficiency of evolutionary algorithms. It is logical, then, to extend the proposed constraint handling technique for spatial surrogate assisted EAs to gain a two-fold benefit.

In the spatial surrogate framework, when multiple spatially distributed surrogates are built, it is quite possible, that surrogate models on all clusters are

not valid (*i.e.* the prediction accuracy is not within the prescribed threshold). In such a case, currently, all the surrogate models are then discarded. There may be merit in utilizing the valid surrogate models as long as the evolutionary search is concentrated in the regions covered by the training data corresponding to the valid models.

The framework for spatial surrogate modeling and infeasibility driven constraint handling, though developed for evolutionary algorithms, are generic approaches. They can be adapted to other population based heuristic methods such as particle swarm methods and similar benefits can be derived as obtained for evolutionary algorithms.

References

[1] J. N. Siddall. *Optimal engineering design - principles and applications*. Marcel Dekker, Inc., New York, 1982.

[2] J. Golinski. Optimal synthesis problems solved by means of nonlinear programming and random methods. *Journal of Mechanisms*, 5(3):287–309, 1970.

[3] G. S. Hornby, A. Globus, D. S. Linden, and J. D. Lohn. Automated antenna design with evolutionary algorithms. In *Proceedings of the AIAA SPACE 2006 conference*, San Jose, CA, September 2006. AIAA 2006-7242.

[4] P. Hajela and E. Lee. Genetic algorithms in truss topological optimization. *International Journal of Solids and Structures*, 32(22):3341–3357, nov 1995.

[5] J. L. Henderson, Z. Gurdal, and A. C. Loos. Combined structural and manufacturing optimization of stiffened composite panels. *Journal of Aircraft*, 36(1):246–254, 1999.

[6] C. S. Krishnamoorthy. Structural optimization in practice: Potential applications of genetic algorithms. *Structural Engineering and Mechanics*, 11:151–170, 2001.

[7] G. Soremekun, Z. Gurdal, R. T Haftka, and L. T. Watson. Composite laminate design optimization by genetic algorithm with generalized elitist selection. *Computers & Structures*, 79:131–143, 2001.

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

[9] S. Nagendra, D. Jestin, Z. Gurdal, R. T. Haftka, and L. T. Watson. Improved genetic algorithm for the design of stiffened composite panels. *Computers & Structures*, 58(3):543–555, 1996.

[10] J. H. Kang and C. G. Kim. Minimum-weight design of compressively loaded composite plates and stiffened panels for postbuckling strength by genetic algorithm. *Composite Structures*, 69(2):239–246, 2005.

[11] F. Muyl, L. Dumas, and V. Herbert. Hybrid method for aerodynamic shape optimization in automotive industry. *Computers & Fluids*, 33(5-6):849–858, 2004.

[12] Y. S Lian and M. S Liou. Multi-objective optimization of transonic compressor blade using evolutionary algorithm. *Journal of Propulsion and Power*, 21(6):979–987, 2005.

[13] S. Obayashi, D. Sasaki, Y. Takeguchi, and N. Hirose. Multiobjective evolutionary computation for supersonic wing-shape optimization. *IEEE Transactions on Evolutionary Computation*, 4(2):182–187, 2000.

[14] D. Sasaki, M. Morikawa, S. Obayashi, and K. Nakahashi. Aerodynamic shape optimization of supersonic wings by adaptive range multiobjective genetic algorithms. In *Evolutionary Multi-Criterion Optimization*, volume 1993 of *Lecture Notes in Computer Science*, pages 639–652, Berlin, 2001. Springer-Verlag Berlin.

[15] R. Kicinger, T. Arciszewski, and K. De Jong. Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers & Structures*, 83(23-24):1943–1978, 2005.

[16] D. Dasgupta and Z. Michalewicz, editors. *Evolutionary Algorithms in Engineering Applications*. Springer-Verlag, Berlin, 1997.

[17] W. Annicchiarico, J. Periaux, M. Cerrolaza, and G. Winter, editors. *Evolutionary Algorithms and Intelligent Tools in Engineering Optimization*. Handbooks on Theory and Engineering Applications of Computational Methods. WIT Press, Southampton, UK, 2005.

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[19] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 1987.

[20] S. S. Rao. *Engineering Optimization*. John Wiley & Sons, New York, 1996.

[21] J. Nocedal and S. J. Wright. *Numerical Optimization*. Operations Research. Springer, 1999.

[22] R. Hooke and T. A. Jeeves. Direct search soluation of numerical and statistical problems. *J. ACM*, 8:212–229, 1961.

[23] J. A. Nelder and R. Mead. A simplex method for function minimization. *The computer journal*, 7(4):308, 1965.

[24] Colin R. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems*. Orient Longman, 1993.

[25] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[26] F. Glover and M. Laguna. *Tabu Search*. Kluwer, Norwell, MA, 1997.

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

[28] D. E. Goldberg. *Genetic Algorithms for Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[29] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial intelligence through simulated evolution.* John Wiley & Sons Inc, 1966.

[30] H. P. Schwefel. *Evolution and optimum seeking.* Wiley-Interscience, 1995.

[31] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[32] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29–41, February 1996.

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

[34] R. G. Reynolds. An introduction to cultural algorithms. In *Proceedings of the Third Annual Confernce on Evolutionary Programming*, pages 131–139, 1994.

[35] K. M. Passino. Biomimicry of bacteria foraging for distributed optimization and control. *IEEE Control Systems Magazine*, 22(3):52–67, June 2002.

[36] Z. W. Geem, J. H. Kim, and G. V. Loganathan. A new heuristic optimization algorithm: Harmony search. *Simulation*, 76(2):60–68, 2001.

[37] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148, 1995.

[38] K. Deb and M. Goyal. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, 26(4):30–45, 1996.

[39] D. A. Van Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm test suites. In *Proceedings of the 1999 ACM Symposium on Applied Computing*, pages 351–357, 1999.

[40] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb. A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation*, 12(3), 2008.

[41] J. F. M. Barhelemy and R. T. Haftka. Approximation concepts for optimum structural design a review. *Structural and Multidisciplinary Optimization*, 5(3):129–144, September 1993.

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

[43] R. E. Smith, B. A. Dike, and S. A. Stegmann. Fitness inheritance in genetic algorithms. In *Proceedings of ACM Symposium on Applied Computing*, pages 345–350. ACM, 1995.

[44] K. Sastry, D. E. Goldberg, and M. Pelikan. Don't evaluate, inherit. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, pages 551–558. Morgan Kaufmann, 2001.

[45] J.-H. Chen, D. E. Goldberg, S.-Y. Ho, and K. Sastry. Fitness inheritance in multi-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, pages 319–326. Morgan Kaufmann, 2002.

[46] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49, 1987.

[47] R. H. Myers and D. C. Montgomery. *Response Surface Methodology: Process and Product in Optimization Using Designed Experiments*. John Wiley & Sons, Inc. New York, NY, USA, 1995.

[48] G. G. Wang, Z. Dong, and P. Aitchison. Adaptive response surface method - A global optimization scheme for approximation-based design problems. *Engineering Optimization*, 33:707–734, 2001.

[49] G. G. Wang and T. W. Simpson. Fuzzy clustering based hierarchical metamodeling for space reduction and design optimization. *Journal of Engineering Optimization*, 36(3):313–335, 2004.

[50] Y. S. Lian and M. S Liou. Multiobjective optimization using coupled response surface model and evolutionary algorithm. *AIAA Journal*, 43(6):1316–1325, 2005.

[51] R. Rikards, H. Abramovich, K. Kalnins, and J. Auzins. Surrogate modeling in design optimization of stiffened composite shells. *Composite Structures*, 73:244–251, 2006.

[52] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1994.

[53] T. Morimoto, J. D. Baerdemaeker, and Y. Hashimoto. An intelligent approach for optimal control of fruit-storage process using neural networks and genetic algorithms. *Computers and Electronics in Agriculture*, 18(2-3):205–224, August 1997.

[54] J. Lee and P. Hajela. Parallel genetic algorithms implementation for multidisciplinary rotor blade design. *Journal of Aircraft*, 33(5):962–969, 1996.

[55] S.-T. Khu, D. Savic, Y. Liu, and H. Madsen. A fast evolutionary-based meta-modelling approach for the calibration of a rainfall–runoff model. In *Proceedings of 1st biennial meeting of the international environmental modelling and software society*, volume 1, pages 147–152, 2002.

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

[57] M. J. D. Powell. Radial basis functions for multivariate interpolation: a reivew. In J. C. Mason and M. G. Cox, editors, *Algorithms for approximation*, pages 143–167. Clarendon Press, 1987.

[58] M. Farina. A neural network based generalized response surface multiobjective evolutionary algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'02)*, volume 1, pages 956–961. IEEE Press, 2002.

[59] P. K. S. Nain and K. Deb. Computationally effective search and optimization procedure using coarse to fine approximation. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'03)*, volume 3, pages 2081–2088, Canberra, Australia, 2003.

[60] Y. Jin, T. Okabe, and B. Sendhoff. Neural network regularization and ensembling using multi-objective evolutionary algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'04)*, volume 1, pages 1–8, 2004.

[61] A. P. Giotis, M. Emmerich, B. Naujoks, K. C. Giannakoglou, and T. Bäck. Low-cost stochastic optimization for engineering applications. In K. C. Giannakoglou, D. Tsahalis, J. Periaux, K. Papailiou, and T. Fogarty, editors, *Proceedings of International Conference on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, Barcelona, Spain, 2001.

[62] Y.-S. Ong, P. B. Nair, and K. Y. Lum. Max-min surrogate-assisted evolutionary algorithm for robust design. *IEEE Transactions on Evolutionary Computation*, 10(4):392–404, 2006.

[63] Y.-S. Ong, A. J. Keane, and P. B. Nair. Surrogate-assisted coevolutionary search. In *Proceedings of the 9th International Conference on Neural Information Processing*, volume 3, pages 1140–1145, Singapore, 2002.

[64] T. Ray and W. Smith. Surrogate assisted evolutionary algorithm for multiobjective optimization. In *Proceedings of the 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Newport, Rhode Island, May 2006. AIAA-2006-2050.

[65] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4:409–436, 1989.

[66] A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In A. Eiben, Thomas Bäck, Marc Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, pages 87–96, 1998.

[67] A. Ratle. Kriging as a surrogate finess landscape in evolutionary optimization. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 15:37–49, 2001.

[68] D. Büche, N. N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms using fitness function models. In *Proceedings of GECCO Workshop on Learning, Adaptation and Approximation in Evolutionary Computation*, pages 166–169, 2003.

[69] M. Li, G. Li, and S. Azarm. A kriging metamodel assisted multi-objective genetic algorithm for design optimization. *Journal of Mechanical Design*, 130(3):031401, March 2008.

[70] M. A. El-Beltagy and A. J. Keane. Evolutionary optimization for computationally expensive problems using gaussian processes. In *Proceedings of the International Conference on Artificial Intelligence*, pages 708–714. CSREA, 2001.

[71] B. Wilson, D. Cappelleri, T. W. Simpson, and M. Frecker. Efficient pareto frontier exploration using surrogate approximations. *Optimization and Engineering*, 2(1):31–50, March 2001.

[72] M. Emmerich, A. P. Giotis, M. Özdemir, T. Bäck, and K. C. Giannakoglou. Metamodel-assisted evolution strategies. In *Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, pages 361–370. Springer, 2002.

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

[74] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[75] J. Knowles. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10(1):50–66, 2006.

[76] W. C. Carpenter and J.-F. Barthelemy. A comparison of polynomial approximation and artificial neural nets as response surface. *Structural and Multidisciplinary Optimization*, 5(3):166–174, September 1993.

[77] T. W. Simpson, T. M. Mauery, J. J. Korte, and F. Mistree. Comparison of response surface and kriging models for multidisciplinary design optimization. In *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis & Optimization*. AIAA, 1998. AIAA-98-4755.

[78] K. Rasheed, Xiao Ni, and S. Vattam. Comparison of methods for developing dynamic reduced models for design optimization. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 9(1):29–37, January 2005.

[79] D. C. Montgomery. *Design and Analysis of Experiments*. Wiley, 1976.

[80] A. B. Owen. Orthogonal arrays for computer experiments, integration and visualization. *Statistica Sinica*, 2(2):439–452, 1992.

[81] A. S. Hedayat, N. J. A. Sloane, and J. Stufken. *Orthogonal Arrays - Theory and Application*. Springer Series in Statistics. Springer, 1999.

[82] Michael D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, may 1979.

[83] Boxin Tang. Orthogonal array-based latin hypercubes. *Journal of the American Statistical Association*, 88(424):1392–1397, 1993.

[84] A. J. Booker, J. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, and M. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural and Multidisciplinary Optimization*, 17(1):1–13, February 1999.

[85] K. S. Won, T. Ray, and K. Tai. A framework for optimization using approximate functions. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'03)*, volume 3, pages 1520–1527, December 2003.

[86] Y. Jin and B. Sendhoff. Reducing fitness evaluations using clustering techniques and neural networks ensembles. In *Proceedings of the 2004 Genetic and Evolutionary Computing Conference GECCO*, pages 688–699, 2004.

[87] A. Ratle. Optimal sampling strategies for learning a fitness model. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'99)*, volume 3, pages 2078–2085, Washington D.C., July 1999.

[88] M. A. El-Beltagy, P. B. Nair, and A. J. Keane. Metamodeling techniques for evolutionary optimization of computationally expensive problems: promises and limitations. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, pages 196–203, Orlando, 1999. Morgan Kaufmann.

[89] M. Papadrakakis, N. D. Lagaros, and Y. Tsompanakis. Structural optimization using evolution strategies and neural networks. *Computer Methods in Applied Mechanics and Engineering*, 156(1-4), April 1998.

[90] A. Schmitz, E. Besnard, and E. Vivies. Reducing the cost of computational fluid dynamics optimization using multilayer perceptrons. In *Proceedings of the 2002 International Joint Conference on Neural Networks*, volume 2, pages 1877–1882. IEEE, 2002.

[91] L. Willmes, T. Bäck, Y. Jin, and B. Sendhoff. Comparing neural networks and kriging for fitness approximation in evolutionary optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation CEC'03*, volume 1, pages 663–670, 2003.

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

[93] Z. Zhou, Y.-S. Ong, and P. B. Nair. Hierarchical surrogate-assisted evolutionary optimization framework. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'04)*, pages 1586–1593. IEEE, 2004.

[94] Z. Zhou, Y.-S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum. Combining global and local surrogate models to accelerate evolutionary optimization. *Ieee Transactions on Systems Man and Cybernetics Part C-Applications and Reviews*, 37(1):66–76, 2007.

[95] Y. Tenne. A framework for memetic optimization using variable global and local surrogate models. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 13(8-9):781, 2009.

[96] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[97] S. Abney, R. E. Schapire, and Y. Singer. Boosting applied to tagging and PP attachment. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

[98] H. A. Abbass. Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'03)*, volume 3, pages 2074–2080, 2003.

[99] K. Hamza and K. Saitou. Vehicle crashworthiness design via a surrogate model ensemble and a coevolutionary genetic algorithm. In *Proceedings of IDETC/CIEASME 2005 International Design Engineering Technical Conference*, California, USA, September 2005.

[100] L. E. Zerpa, N. V. Queipo, S. Pintos, and J.-L. Salager. An optimization methodology of alkaline-surfactant-polymer flooding processes using field scale numerical simulation and multiple surrogates. *Journal of Petroleum Science and Engineering*, 47(3-4):197–208, June 2005.

[101] T. Goel, R. T. Haftka, W. Shyy, and N. V. Queipo. Ensemble of surrogates. *Structural and Multidisciplinary Optimization*, 33(3):199–216, March 2007.

[102] Z. Zhou, Y.-S. Ong, M. H. Lim, and B. S. Lee. Memetic algorithm using multi-surrogates for computationally expensive optimization problems. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 11(10):957–971, 2007.

[103] Y. Zhao, J. Gao, and X. Yang. A survey of neural network ensembles. In *International Conference on Neural Networks and Brain, 2005. ICNN&B*, volume 1, pages 438–442, 2005.

[104] D. Lim, Y.-S. Ong, Y. Jin, and B. Sendhoff. A study on metamodelling techniques, ensembles, and multi-surrogates in evolutionary computation. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1288–1295, 2007.

[105] Y. Jin, M. Olhofer, and B. Sendhoff. Managing approximate models in evolutionary aerodynamic design optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 592–599, May 2001.

[106] Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494, 2002.

[107] L. Bull. On model-based evolutionary computation. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 3(2):76–82, September 1999.

[108] S. D'Angelo and E. A. Minsci. Multi-objective evolutionary optimization of subsonic airfoils by kriging approximation and evolutionary control. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'05)*, volume 2, pages 1262–1267, 2005.

[109] P. B. Nair, A. J. Keane, and R. P. Shimpi. Combining approximation concepts with algorithm-based structural optimization procedures. In *Proceedings of the 39th AIAA/ASMEASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pages 1741–1751, 1998.

[110] Y. Jin, M. Olhofer, and B. Sendhoff. On evolutionary optimization with approximate fitness functions. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, pages 786–793, Las Vegas, Nevada, 2000.

[111] Y. Tenne and S. W. Armfield. Metamodel accuracy assessment in evolutionary optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation, 2008 (CEC 2008)*, pages 1505–1512, June 2008.

[112] K. Rasheed and H. Hirsh. Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, pages 628–635, Las Vegas, 2000. Morgan Kaufmann.

[113] A. Mutoh, S. Kato, T. Nakamura, and H. Itoh. Reducing execution time on genetic algorithms in real-world applications using fitness prediction. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 1, pages 552–559, Sydney, Australia, 2003. IEEE.

[114] M. Emmerich, K. C. Giannakoglou, and B. Naujoks. Single and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006.

[115] H. Ulmer, F. Streichert, and A. Zell. Evolution startegies assisted by gaussian processes with improved pre-selection criterion. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'03)*, volume 1, pages 692–699, 2003.

[116] K. S. Anderson and Y. Hsu. Genetic crossover strategy using an approximation concept. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*, volume 1, 1999.

[117] K. Abboud and M. Schoenauer. Surrogate deterministic mutation: Preliminary results. In *Artificial Evolution*, volume 2310 of *Lecture Notes in Computer Science*, pages 919–954. Springer, 2002.

[118] P. Moscato and C. Cotta. A gentle introduction to memetic algorithms. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 105–144. Kluwer Academic Press, 2003.

[119] P. Moscato and C. Cotta. Memetic algorithms. In T. González, editor, *Handbook of Approximation Algorithms and Metaheuristics*. Chapman & Hall/CRC Press, 2007.

[120] K.-H. Liang, X. Yao, and C. Newton. Combining landscape approximation and local search in global optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation*, volume 2, pages 1514–1520, 1999.

[121] K.-H. Liang, X. Yao, and C. Newton. Evolutionary search of approximated N-Dimensional landscape. *International Journal of Knowledge-based Intelligent Engineering Systems*, 4(3):172–183, 2000.

[122] K. W. C. Ku, M. W. Mak, and W.-C. Siu. A study of the lamarckian evolution of recurrent neural networks. *IEEE Transactions on Evolutionary Computation*, 4(1):31–42, 2000.

[123] Y.-S. Ong, P. B. Nair, and A. J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 41(4):687–696, 2003.

[124] Y.-S. Ong, K. Y. Lum, P. B. Nair, D. M. Shi, and Z. K. Zhang. Global convergence unconstrained and bound constrained Surrogate-Assisted evolutionary search in aerodynamic shape design. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'03)*, volume 3, pages 1856–1863, Canberra, Australia, 2003.

[125] Y.-S. Ong, Z. Zhou, and D. Lim. Curse and blessing of uncertainty in evolutionary algorithms using approximation. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC-2006)*, pages 2928–2935, 2006.

[126] J. MacQueen. Some methods for classification and analysis of multivariate ob servations. In *Proc. Fifth Berkeley Symp. on Math. Statist. and Prob.*, volume 1, pages 281–297, 1967.

[127] H. B. Nielsen. Dace - a matlab kriging toolbox.

[128] C. A. C. Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245–1287, January 2002.

[129] A. Homaifar, C. X. Qi, and S. H. Lai. Constrained optimization via genetic algorithms. *Simulation*, 62(4):242–253, 1994.

[130] A. Kuri-Morales and C. V. Quezada. A Universal Eclectic Genetic Algorithm for Constrained Optimization. In *Proceedings 6th European Congress on Intelligent Techniques & Soft Computing, EUFIT'98*, pages 518–522, Aachen, Germany, September 1998. Verlag Mainz.

[131] J. A. Joines and C. R. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In David Fogel, editor, *Proceedings of the first IEEE Conference on Evolutionary Computation (CEC)*, volume 2, pages 579–584, Orlando, Florida, 1994.

[132] Z. Michalewicz and N. F. Attia. Evolutionary optimization of constrained problems. In *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pages 98–108. World Scientific, 1994.

[133] Z. Michalewicz. Genetic Algorithms, Numerical Optimization, and Constraints. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA-95)*, pages 151–158, San Mateo, California, July 1995. University of Pittsburgh, Morgan Kaufmann Publishers.

[134] J. C. Bean. Genetics and random keys for sequencing and optimization. Technical Report TR 92-43, Department of Industrial and Operations Engineering, The University of Michigan, 1992.

[135] A. B. Hadj-Alouane and J. C. Bean. A Genetic Algorithm for the Multiple-Choice Integer Program. *Operations Research*, 45(1):92–101, 1997.

[136] R. Farmani and J. A. Wright. Self-adaptive fitness formulation for constrained optimization. *IEEE Transactions on Evolutionary Computation*, 7(5):445–455, October 2003.

[137] Y. Wang, Z. Cai, Y. Zhou, and Z. Fan. Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. *Structural and Multidisciplinary Optimization*, 37(4):395–413, January 2009.

[138] F. Hoffmeister and J. Sprave. Problem-independent handling of constraints by use of metric penalty functions. In Lawrence J. Fogel, Peter J. Angeline, and Thomas Bäck, editors, *Proceedings of the Fifth Annual Conference on Evolutionary Programming (EP'96)*, pages 289–294, San Diego, California, February 1996. The MIT Press.

[139] T. P. Runarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Compution*, 4(3):284–294, September 2000.

[140] H. Xiao and J. W. Zu. A new constrained multiobjective optimization algorithm based on artificial immune systems. In *Proceedings of the 2007 IEEE International Conference on Mechatronics and Automation*, Harbin, China, 2007.

[141] L. D. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[142] M. Schoenauer and Z. Michalewicz. Evolutionary computation at the edge of feasibility. In *Parallel Problem Solving from Nature - PPSN IV*, volume 1141 of *Lecture Notes in Computer Science*, pages 245–254, 1996.

[143] S. Koziel and Z. Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

[144] Z. Michalewicz and J. Xiao. Evaluation of Paths in Evolutionary Planner/Navigator. In *Proceedings of the 1995 International Workshop on Biologically Inspired Evolutionary Systems*, pages 45–52, Tokyo, Japan, May 1995.

[145] Z. Michalewicz and G. Nazhiyath. Genocop III: A co-evolutionary algorithm for numerical optimization with nonlinear constraints. In *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, volume 2, pages 647–651, 1995.

[146] J. Xiao, Z. Michalewicz, and L. Zhang. Evolutionary Planner/Navigator: Operator Performance and Self-Tuning. In *Proceedings of the 3rd IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, May 1996.

[147] J. Xiao, Z. Michalewicz, and K. Trojanowski. Adaptive Evolutionary Planner/Navigator for Mobile Robots. *IEEE Transactions on Evolutionary Computation*, 1(1):18–28, 1997.

[148] D. Powell and M. M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, pages 424–431, San Mateo, California, July 1993. Morgan Kaufmann Publishers.

[149] K. Deb. An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2):311–338, 2000.

[150] C. A. C. Coello. Treating constraints as objectives for single-objective evolutionary optimization. *Engineering Optimization*, 32(3):275–308, January 2000.

[151] P. D. Surry and N. J. Radcliffe. The COMOGA method: Constrained optimisation by multi-objective genetic algorithms. *Control and Cybernetics*, 26(3):391–412, 1997.

[152] E. Camponogara and S. N. Talukdar. A genetic algorithm for constrained and multiobjective optimization. In J. T. Alander, editor, *3rd Nordic Workshop on Genetic Algorithms and Their Applications*, pages 49–62, 1997.

[153] C. A. C. Coello and E. Mezura-Montes. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, 16(3):193–203, July 2002.

[154] D. A. G. Vieira, R. L. S. Adriano, L. Krahenbuhl, and J. A. Vasconcelos. Handing constraints as objectives in a multiobjective genetic based algorithm. *Journal of Microwaves and Optoelectronics*, 2(6):50–58, December 2002.

[155] D. A. G. Vieira, R. L. S. Adriano, J. A. Vasconcelos, and L. Krahenbuhl. Treating constraints as objectives in multiobjective optimization problems using niched pareto genetic algorithm. *IEEE Transactions on Magnetics*, 40(2):1188–1191, March 2004.

[156] S. B. Hamida and M. Schoenauer. An adaptive algorithm for constrained optimization problems. In *Parallel Problem Solving from Nature - PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, pages 529–538, 2000.

[157] S. B. Hamida and M. Schoenauer. ASCHEA: new results using adaptive segregational constraint handling. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'02)*, volume 1, pages 884–889, May 2002.

[158] R. Hinterding and Z. Michalewicz. Your brains and my beauty: parent matching for constrained optimisation. In *Proceedings of the IEEE International Conference on Evolutionary Computation (CEC'98)*, pages 810–815, May 1998.

[159] E. Mezura-Montes and C. A. C. Coello. A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, 9(1):1–17, 2 2005.

[160] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons Pvt. Ltd., 2001.

[161] C. A. C. Coello. Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems. *Computers in Industry*, 41(2):113–127, January 2000.

[162] K. C. Giannakoglou. Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *Progress in Aerospace Sciences*, 38(1):43–76, 2002.

[163] A. Jameson, W. Schmidt, and E. Turkel. 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.

[164] S. Rajeev and C. S. Krishnamoorthy. Discrete optimization of structures using genetic algorithms. *Journal of Structural Engineering*, 118(5):1233–1250, May 1992.

[165] C. A. C. Coello, M. Rudnick, and A. D. Christiansen. Using genetic algorithms for optimal design of trusses. In *Proceedings of Sixth Interanational Conference on Tools with Artificial Intelligence*, pages 88–94, New Orleans, LA, USA, nov 1994.

[166] S.-J. Wu and P.-T. Chow. Steady-state genetic algorithms for discrete optimization of trusses. *Computers & Structures*, 56(6):979–991, 1995.

[167] K. Deb and S. Gulati. Design of truss-structures for minimum weight using genetic algorithms. *Finite Elements in Analysis and Design*, 37(5):447–465, may 2001.

[168] A. Kaveh and V. Kalatjari. Topology optimization of trusses using genetic algorithm, force method and graph theory. *International Journal for Numerical Methods in Engineering*, 58(5):771–791, 2003.

[169] A. Kaveh and M. Shahrouzi. Simultaneous topology and size optimization of structures by genetic algorithm using minimal length chromosome. *Engineering Computations*, 23(6):644–674, 2006.

[170] S. Rajeev and C. S. Krishnamoorthy. Genetic Algorithms-Based methodologies for design optimization of trusses. *Journal of Structural Engineering*, 123(3):350–358, March 1997.

[171] W. Tang, L. Tong, and Y. Gu. Improved genetic algorithm for design optimization of truss structures with sizing, shape and topology variables. *International Journal for Numerical Methods in Engineering*, 62(13):1737–1762, 2005.

[172] M. R. Ghasemi, E. Hinton, and R. D. Wood. Optimization of trusses using genetic algorithms for discrete and continuous variables. *Engineering Computations*, 16(3):272–301, 1999.

[173] K. Deb, S. Gulati, and S. Chakrabarti. Optimal Truss-Structure design using Real-Coded genetic algorithms. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 479–486, University of Wisconsin, Madison, Wisconsin, USA, 1998. Morgan Kaufmann.

[174] N. Turkkan. Discrete optimization of structures using a floating point genetic algorithm. In *Proceedings of Annual Conference of the Canadian Society for Civil Engineering*, Canada, 2003.

[175] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):493–530, 1989.

[176] Y. M. Xie and G. P. Steven. A simple evolutionary procedure for structural optimization. *Computational Structures*, 49(5):885–896, 1993.

[177] G. P. Steven, O. M. Querin, and Mike Xie. Evolutionary structural optimisation (ESO) for combined topology and size optimisation of discrete structures. *Computer Methods in Applied Mechanics and Engineering*, 188(4):743–754, 2000.

[178] D. N. Chu, Y. M. Xie, A. Hira, and G. P. Steven. On various aspects of evolutionary structural optimization for problems with stiffness constraints. *Finite Elements in Analysis and Design*, 24(4):197–212, 1997.

[179] P. Tanskanen. A multiobjective and fixed elements based modification of the evolutionary structural optimization method. *Computer Methods in Applied Mechanics and Engineering*, 196(1-3):76–90, 2006.

[180] W. Gutkowski, J. Bauer, and J. Zawidzka. An effective method for discrete structural optimization. *Engineering Computations*, 17(4):417–426, 2000.

[181] M. Pyrz and J. Zawidzka. Optimal discrete truss design using improved sequential and genetic algorithm. *Engineering Computations*, 18(8):1078–1090, 2001.

[182] K. S. Lee and Z. W. Geem. A new structural optimization method based on the harmony search algorithm. *Computers & Structures*, 82(9-10):781–798, 2004.

[183] X. Yao and Y. Liu. Fast evolutionary strategies. In Peter J. Angeline, Robert G. Reynolds, John R. McDonnell, and Russ Eberhart, editors, *Evolutionary Programming VI*, pages 151–161, Berlin, 1997. Springer.

[184] Z. Michalewicz and M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.

[185] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[186] K. Deb, A. Pratap, and T. Meyarivan. Constrained test problems for multi-objective evolutionary optimization. In *Evolutionary Multi-Criterion Optimization*, volume 1993 of *Lecture Notes in Computer Science*, pages 284–298, 2001.

Appendix A

f-series optimization problems

The f-series of problems are the unconstrained single objective problems [183]. Thirteen problems (f01–f13) in the f-series are considered in the thesis. Problems f01–f07 are unimodal with a single global minimum. Functions f08–f13 are multi-modal with a large number of local minima. The number of local minima increases exponentially with the problem dimension. All the problems are scalable in the number of design variables. In this study, 30 design variables are considered for all the problems.

A.1 Sphere Model

$$f_1(\mathbf{x}) = \sum_{i=1}^{30} x_i^2$$

$$-100 \leq x_i \leq 100, \quad i = 1, 2, \ldots, 30 \qquad \min(f_1) = f_1(0, \ldots, 0) = 0$$

A.2 Schwefel's Problem 2.22

$$f_2(\mathbf{x}) = \sum_{i=1}^{30} |x_i| + \prod_{i=1}^{30} |x_i|$$

$$-10 \leq x_i \leq 10, \quad i = 1, 2, \ldots, 30 \qquad \min(f_2) = f_2(0, \ldots, 0) = 0$$

A.3 Schwefel's Problem 1.2

$$f_3(\mathbf{x}) = \sum_{i=1}^{30} \left(\sum_{j=1}^{i} x_j \right)^2$$

$$-100 \leq x_i \leq 100, \quad i = 1, 2, \ldots, 30 \qquad \min(f_3) = f_3(0, \ldots, 0) = 0$$

A.4 Schwefel's Problem 2.21

$$f_4(\mathbf{x}) = \max_i \{|x_i|,\ 1 \leq i \leq 30\}$$

$$-100 \leq x_i \leq 100, \quad i = 1, 2, \ldots, 30 \qquad \min(f_4) = f_4(0, \ldots, 0) = 0$$

A.5 Generalized Rosenbrock's Function

$$f_5(\mathbf{x}) = \sum_{i=1}^{29} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$$

$$-30 \leq x_i \leq 30, \quad i = 1, 2, \ldots, 30 \qquad \min(f_5) = f_5(1, \ldots, 1) = 0$$

A.6 Step Function

$$f_6(\mathbf{x}) = \sum_{i=1}^{30} \left(\lfloor x_i + 0.5 \rfloor \right)^2$$

$$-100 \leq x_i \leq 100, \quad i = 1, 2, \ldots, 30 \qquad \min(f_6) = f_6(0, \ldots, 0) = 0$$

A.7 Quartic Function with Noise

$$f_7(\mathbf{x}) = \sum_{i=1}^{30} i x_i^4 + random[0, 1)$$

$$-1.28 \leq x_i \leq 1.28, \quad i = 1, 2, \ldots, 30 \qquad min(f_7) = f_7(0, \ldots, 0) = 0$$

A.8 Generalized Schwefel's Problem 2.26

$$f_8(\mathbf{x}) = -\sum_{i=1}^{30} \left(x_i \sin\left(\sqrt{|x_i|}\right)\right)$$

$$-500 \leq x_i \leq 500, \quad i = 1, 2, \ldots, 30$$

$$\min(f_8) = f_8(420.9687, \ldots, 420.9687) = -12569.5$$

A.9 Generalized Rastrigin's Function

$$f_9(\mathbf{x}) = \sum_{i=1}^{30} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$$

$$-5.12 \leq x_i \leq 5.12, \quad i = 1, 2, \ldots, 30 \qquad \min(f_9) = f_9(0, \ldots, 0) = 0$$

A.10 Ackley's Function

$$f_{10}(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30} x_i^2}\right) - \exp\left(\frac{1}{30}\sum_{i=1}^{30}\cos(2\pi x_i)\right) + 20 + e$$

$$-32 \leq x_i \leq 32, \quad i = 1, 2, \ldots, 30 \qquad \min(f_{10}) = f_{10}(0, \ldots, 0) = 0$$

A.11 Generalized Griewank Function

$$f_{11}(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

$$-600 \leq x_i \leq 600, \quad i = 1, 2, \ldots, 30 \qquad \min(f_{11}) = f_{11}(0, \ldots, 0) = 0$$

A.12 Generalized Penalized Functions

$$f_{12}(\mathbf{x}) = \frac{\pi}{30}\Bigg\{10\sin^2(\pi y_1) + \sum_{i=1}^{29}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right] + (y_n-1)^2\Bigg\} + \sum_{i=1}^{30}u(x_i,10,100,4)$$

$$-50 \le x_i \le 50, \quad i = 1,2,\ldots,30 \qquad \min(f_{12}) = f_{12}(1,\ldots,1) = 0$$

$$f_{13}(\mathbf{x}) = 0.1\Bigg\{\sin^2(3\pi x_1) + \sum_{i=1}^{29}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right] + (x_{30}-1)^2[1+\sin^2(2\pi x_{30})]\Bigg\} + \sum_{i=1}^{30}u(x_i,5,100,4)$$

$$-50 \le x_i \le 50, \quad i = 1,2,\ldots,30 \qquad \min(f_{13}) = f_{13}(1,\ldots,1) = 0$$

where

$$u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, & x_i > a, \\ 0, & -a \le x_i \le a, \\ k(-x_i-a)^m, & x_i \le -a. \end{cases}$$

$$y_i = 1 + \frac{1}{4}(x_i+1)$$

Appendix B

g-series optimization problems

The g-series of problems consists of constrained single objective problems [184, 143]. The g-series has a total of eleven problems out of which eight problems without equality constraints are studied.

B.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 $f(x^*) = -15$.

B.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 $f(x^*) = 0.803619$.

B.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).

B.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.

B.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 $(g1, g2, g3, g4, g5, g6)$.

B.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.

B.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).

B.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).

$$g_1(x) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$$

where $0 \leq x_i \leq 10$ $(i = 1, 2, 3)$ and $p, q, r = 1, 2, \ldots, 9$.

Appendix C

Zitlzer-Deb-Thiele's (ZDT) test problems

Zitzler *et al.* [185] proposed six test problems for bi-objective unconstrained optimization. The problem ZDT5 is defined for binary representation and is not considered. The test problems have the form,

$$\begin{aligned} &\text{Minimize} \quad f_1(\mathbf{x}), \\ &\text{Minimize} \quad f_2(\mathbf{x}) = g(\mathbf{x})h(f_1(\mathbf{x}), g(\mathbf{x})). \end{aligned}$$

Function definitions for the three functions $f_1(\mathbf{x})$, $g(\mathbf{x})$ and $h(\mathbf{x})$ are varied for the test problems. The Pareto optimal front for all the problems (ZDT1 to ZDT4, ZDT6) corresponds to $g(\mathbf{x}) = 1$. (See [160] for detailed discussion on these test problems.)

C.1 ZDT1

ZDT1 is a 30-variable ($n = 30$) problem with convex Pareto optimal front as shown in Fig. C.1. The functions are defined as follows:

$$\begin{aligned} f_1(\mathbf{x}) &= x_1, \\ g(\mathbf{x}) &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i, \\ h(f_1, g) &= 1 - \sqrt{f_1/g}. \end{aligned}$$

All the variables lie in the range [0,1]. The Pareto optimal front corresponds to $0 \leq x_1 \leq 1$ and $x_i = 0$ for $i = 2, 3, \ldots, 30$.

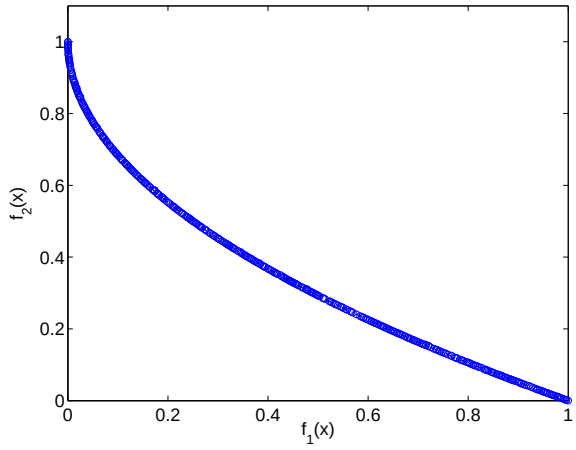


Figure C.1: Pareto optimal front for ZDT1

C.2 ZDT2

ZDT2 is a 30-variable ($n = 30$) problem with nonconvex Pareto optimal front as shown in Fig. C.2. The functions are defined as follows:

$$\begin{aligned} f_1(\mathbf{x}) &= x_1, \\ g(\mathbf{x}) &= 1 + \frac{9}{n-1} \sum_{i=2}^{n} x_i, \\ h(f_1, g) &= 1 - (f_1/g)^2. \end{aligned}$$

All the variables lie in the range [0,1]. The Pareto optimal front corresponds to $0 \leq x_1 \leq 1$ and $x_i = 0$ for $i = 2, 3, \ldots, 30$.

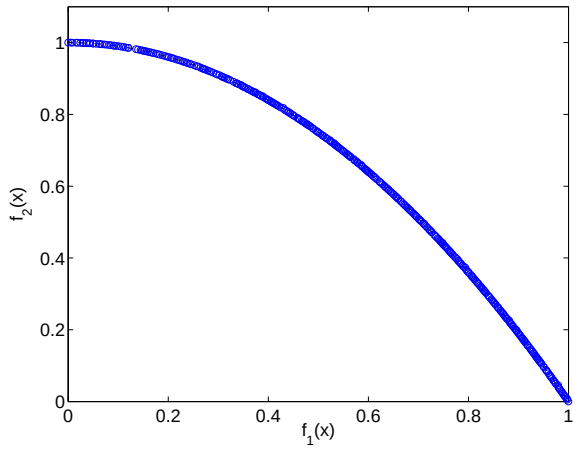


Figure C.2: Pareto optimal front for ZDT2

C.3 ZDT3

ZDT3 is a 30-variable ($n = 30$) problem with disconnected Pareto optimal fronts as shown in Fig. C.3. The functions are defined as follows:

$$\begin{aligned} f_1(\mathbf{x}) &= x_1, \\ g(\mathbf{x}) &= 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i, \\ h(f_1, g) &= 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1). \end{aligned}$$

All the variables lie in the range [0,1]. The Pareto optimal front corresponds to $x_i = 0$ for $i = 2, 3, \ldots, 30$. Not all points corresponding to $0 \leq x_1 \leq 1$ lie on the Pareto front.

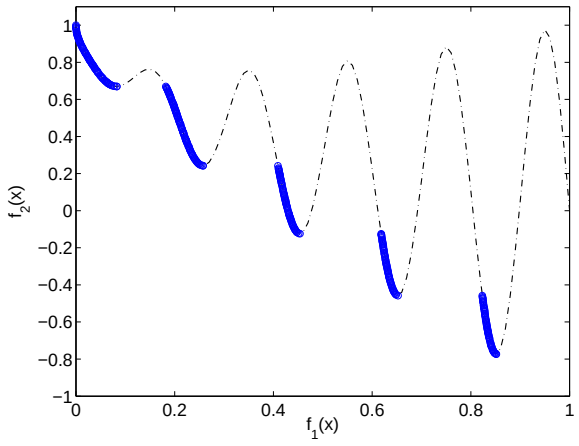


Figure C.3: Pareto optimal front for ZDT3

C.4 ZDT4

ZDT4 is a 10-variable ($n = 10$) problem with convex Pareto optimal front as shown in Fig. C.4. The functions are defined as follows:

$$\begin{aligned} f_1(\mathbf{x}) &= x_1, \\ g(\mathbf{x}) &= 1 + 10(n-1) + \sum_{i=2}^{n} (x_i^2 - 10\cos(4\pi x_i)), \\ h(f_1, g) &= 1 - \sqrt{f_1/g}. \end{aligned}$$

The variable x_1 lies in the range [0,1] and the other variables lie in the range [-5,5]. The problem has many local Pareto optimal fronts, each corresponding to $0 \leq x_1 \leq 1$ and $x_i = 0.5m$ for $i = 2, 3, \ldots, 10$, where m is in the range [-10,10]. The global Pareto optimal front corresponds to $m = 0$.

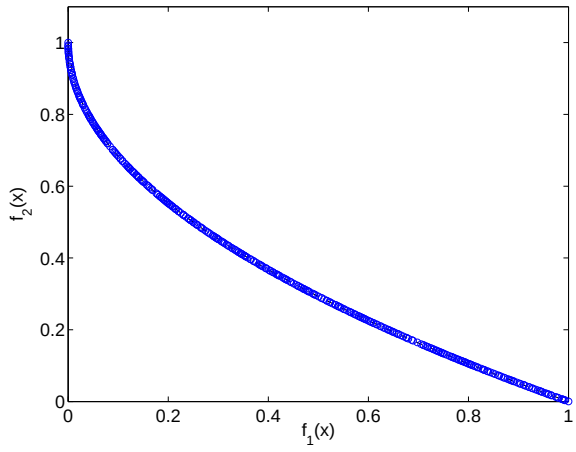


Figure C.4: Pareto optimal front for ZDT4

C.5 ZDT6

ZDT6 is a 10-variable ($n = 10$) problem having a nonconvex Pareto optimal front as shown in Fig. C.5. The functions are defined as follows:

$$f_1(\mathbf{x}) = 1 - \exp(-4x_1)\sin^6(6\pi x_1),$$
$$g(\mathbf{x}) = 1 + 9\left[\frac{1}{9}\sum_{i=2}^{n} x_i\right]^{0.25},$$
$$h(f_1, g) = 1 - (f_1/g)^2.$$

All variables lie in the range [0,1]. The Pareto optimal front corresponds to $0 \leq x_1 \leq 1$ and $x_i = 0$ for $i = 2, 3, \ldots, 10$.

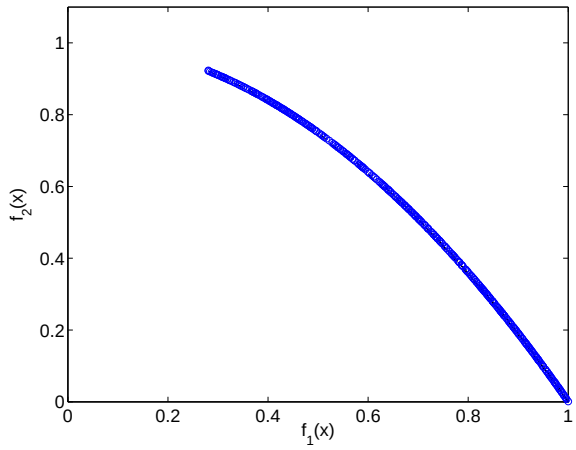


Figure C.5: Pareto optimal front for ZDT6

Appendix D

CTP problems

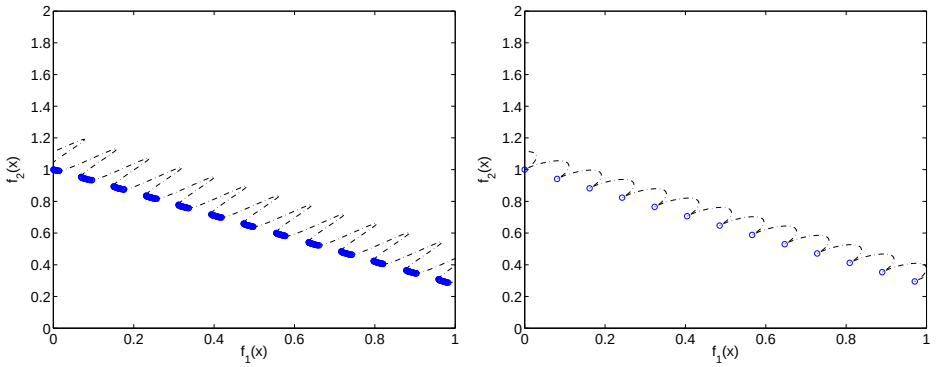
Deb *et al.* [186] proposed constrained biobjective test problems (CTP). The test functions CTP2-CTP7 have a single constraint and CTP8 has two constraints. The mathematical formulation of CTP2-CTP8 is given in Eq. D.1. Both the constraints of CTP8 are of the same form as the single constraint in other CTP problems.

$$\begin{aligned}
&\text{Min. } f_1(\mathbf{x}) = x_1, \\
&\text{Min. } f_2(\mathbf{x}) = C(\mathbf{x}) \left(1 - \sqrt{f_1(\mathbf{x})/C(\mathbf{x})}\right), \\
&\quad g(\mathbf{x}) = \cos(\theta)\left(f_2(\mathbf{x}) - e\right) - \sin(\theta) f_1(\mathbf{x}) \geq \\
&\qquad\qquad a|\sin\left(b\pi\left(\sin(\theta)\left(f_2(\mathbf{x}) - e\right) + \cos(\theta) f_1(\mathbf{x})\right)^c\right)|^d, \\
&\quad C(\mathbf{x}) = 1 + \sum_{i=2}^{10} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right),
\end{aligned} \tag{D.1}$$

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 D.1.

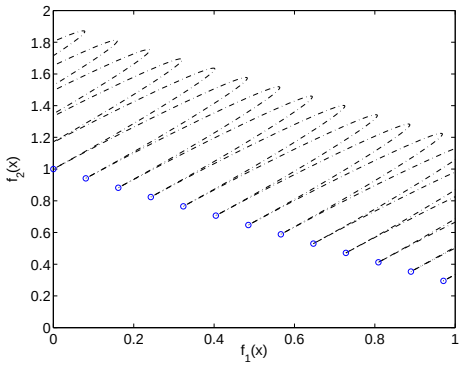
Table D.1: Parameters for the Test Problems CTP2 to CTP8

	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

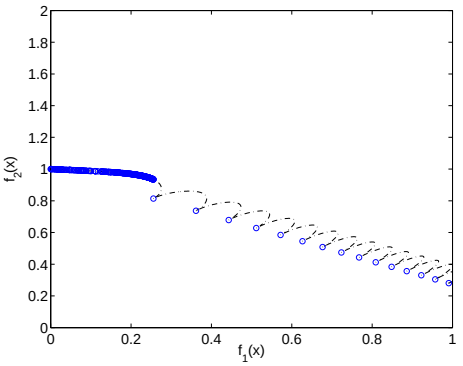


(a) CTP2

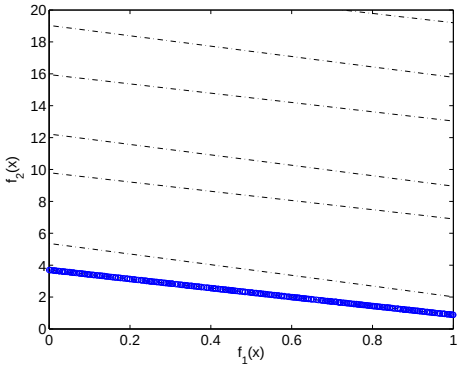
(b) CTP3



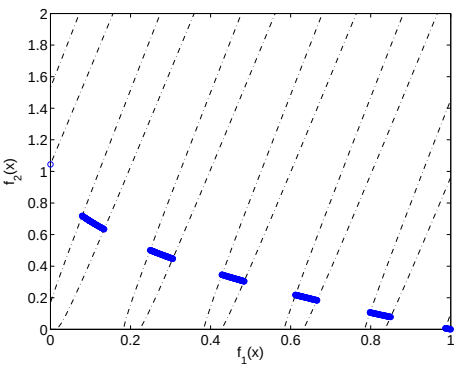
(c) CTP4



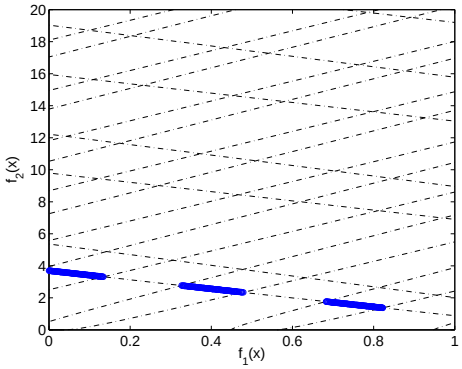
(d) CTP5



(e) CTP6



(f) CTP7



(g) CTP8

Figure D.1: Pareto optimal fronts for CTP