NANYANG TECHNOLOGICAL UNIVERSITY

EVOLUTIONARY OPTIMIZATION FOR COMPUTATIONALLY EXPENSIVE PROBLEMS

EVOLUTIONARY OPTIMIZATION FOR COMPUTATIONALLY EXPENSIVE PROBLEMS

DUDY LIM

DUDY LIM

SCHOOL OF COMPUTER ENGINEERING

2009

2009

EVOLUTIONARY OPTIMIZATION FOR COMPUTATIONALLY EXPENSIVE PROBLEMS

DUDY LIM

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2009

Acknowledgments

I would like to express my gratitude to several individuals. First, I would like to thank **Dr. Ong Yew Soon**, my thesis advisor in the university. I also want to take this opportunity to express special thanks to **Dr. Jin Yaochu** and **Dr. Bernhard Sendhoff**, who have been my thesis co-advisors during the time I spent at Honda Research Institute Europe, Germany. Their ideas, supports, and encouragements are priceless for the completion of this thesis.

This thesis could never have been written without the support and assistance of all my colleagues in Parallel and Distributed Computing Centre (PDCC), Emerging Research Laboratory (ER Lab), Centre for Computational Intelligence (C2i), and School of Physical and Mathematical Sciences (SPMS).

Deep thanks go to my parents, brother, sister, and friends. Their generous and unflagging support, understanding, time, and love, have really guided me through many good and bad moments during these few years of research.

Finally, I want to truly acknowledge the role of Nanyang Technological University (NTU), especially School of Computer Engineering (SCE), who has laid the foundation for my education by giving me the opportunity to pursue higher degree in Singapore.

Dudy Lim

Table of Contents

Summary

Despite all the appealing features of Evolutionary Algorithms (EAs), thousands of calls to the analysis or simulation codes are often required to locate a near optimal solution. Two major solutions for this issue are: *1)* to use computationally less expensive surrogate models, and *2)* to use parallel and distributed computers.

In this thesis, model management frameworks utilizing a diverse set of surrogate models are proposed. The proposed Generalized Surrogate Memetic (GSM) framework aims to unify diverse set of data-fitting models synergistically in the evolutionary search. In particular, the GSM framework exploits both the positive and negative impacts of approximation errors in the surrogate models used. An extended management framework is also proposed for EAs using multi-scale models and demonstrated on two real-world examples. Experimental study performed using data-fitting and multi-scale models indicates that the proposed frameworks are capable of attaining reliable, high quality, and efficient performance under a limited computational budget.

In what follows, possibilities for further acceleration of the evolutionary optimization life cycle through parallelization are also considered. When applied to small-scale, dedicated, and homogeneous computing nodes, this seems to be a formidable solution. However, in a large-scale computing farm such as the Grid, reality proves otherwise. In a Grid computing environment, which emphasizes on the seamless sharing of computing resources across institutions, heterogeneity of resources is inevitable. In such situation, conventional parallelization without considering the heterogeneity of computing resources is likely to produce inefficient optimization. The latter part of this thesis summarizes our works on parallelizing evolutionary optimization in a heterogeneous Grid computing environment.

List of Abbreviations

ANN	Artificial Neural Network
API	Application Programming Interface
CA	Certificate Authority
CEM	Computational Electro Magnetics
CFD	Computational Fluid Dynamics
COGKIT	Commodity Grid Kit
CSM	Computational Structural Mechanics
DE	Differential Evolution
DFP	Davidon-Fletcher-Powell
DFT	Density Functional Theory
DOE	Design of Experiment
EA	Evolutionary Algorithm
EDA	Estimation of Distribution Algorithm
EGS	Evolutionary Gradient Search
EP	Evolutionary Programming
ES	Evolution Strategy
FEA	Finite Element Analysis
FMC	Finite Markov Chain
FSQP	Feasible Sequential Quadratic Programming
GA	Genetic Algorithm
GD	Generational Distance
GE-PEA	Grid-Enabled Parallel EA
GE-PEA-0	Original Grid-Enabled Parallel EA design prior to GE-PEA
GP	Gaussian Process

GP	Genetic Programming
GridFTP	Grid File Transfer Protocol
GridRPC	Grid Remote Procedure Call
GSI	Grid Security Infrastructure
GSM	Generalized Surrogate Memetic
GS-MOMA	Generalized Surrogate Multi-Objective Memetic Algorithm
GS-SOMA	Generalized Surrogate Single-Objective Memetic Algorithm
GS-SOMA-I	Generalized Surrogate Single-Objective Memetic Algorithm with ensemble model
GS-SOMA-II	Generalized Surrogate Single-Objective Memetic Algorithm with quadratic model
HR	Hypervolume Ratio
MA	Memetic Algorithm
MA-DFM	Memetic Algorithm with Dynamic Fidelity Models
MA-FFM	Memetic Algorithm with Fixed Fidelity Model
MDS	Monitoring and Discovery Service
MMA	Molecular Memetic Algorithm
MMA-0	Molecular Memetic Algorithm without any lower fidelity models
MMA-OSS	Molecular Memetic Algorithm with Ojamae-Shavitt-Singer Model
MMA-TTM	Molecular Memetic Algorithm with Thole-Type Model
MOEA	Multi-Objective Evolutionary Algorithm
MOGA	Multi-Objective Genetic Algorithm
MOMA	Multi-Objective Memetic Algorithm
MO	Multi-Objective
MOO	Multi-Objective Optimization
MS	Maximum Spread
NPGA	Niched-Pareto Genetic Algorithm
NPGA-II	Niched-Pareto Genetic Algorithm II
N-RMSE	Normalized Root Mean Square Error
NSGA	Non-dominated Sorting Genetic Algorithm
NSGA-II	Non-dominated Sorting Genetic Algorithm II

OSS	Ojamae-Shavitt-Singer
PAES	Pareto Archived ES
PEA	Parallel Evolutionary Algorithm
PF	Pareto Front
PR	Polynomial Regression
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
RMSE	Root Mean Square Error
RW	Roulette Wheel
SAEA	Surrogate-Assisted Evolutionary Algorithm
SAGA	Surrogate-Assisted Genetic Algorithm
SAMA	Surrogate-Assisted Memetic Algorithm
SOMA	Single-Objective Memetic Algorithm
SO	Single-Objective
SOO	Single Objective Optimization
SPEA	Strength Pareto Evolutionary Algorithm
SPEA-II	Strength Pareto Evolutionary Algorithm II
SQP	Sequential Quadratic Programming
SS-SOMA-AV	Single-Surrogate Single-Objective Memetic Algorithm averaged from SS-SOMA-GP, SS-SOMA PR, and SS-SOMA-RBF
SS-SOMA-GP	Single-Surrogate Single-Objective Memetic Algorithm with Gaussian Process
SS-SOMA-Perfect	Single-Surrogate Single-Objective Memetic Algorithm with Perfect model
SS-SOMA-PR	Single-Surrogate Single-Objective Memetic Algorithm with Polynomial Regression
SS-SOMA-RBF	Single-Surrogate Single-Objective Memetic Algorithm with Radial Basis Function
SUS	Stochastic Universal Sampling
SVM	Support Vector Machine
TTM	Thole-Type Model

VEGA	Vector Evaluated Genetic Algorithm
ZDT	Zitzler-Deb-Thiele

List of Figures

List of Tables

Chapter 1

Introduction

In science and engineering, the design optimization life cycle has been revolutionized by the introduction of massive computing power as well as advances in computational sciences and artificial intelligence. The advancement in hardware technology has brought about affordable and powerful computing platforms. Nevertheless, it is evident that the increasing complexity of arising problems in many real-world scenarios has led to the inevitable demand for even greater computational power. In design analysis and optimization processes where high-fidelity analysis codes are used, each function evaluation requiring the simulation of the high-fidelity analysis codes, such as Computational Structural Mechanics (CSM), Computational Fluid Dynamics (CFD) or Computational Electro Magnetics (CEM), costs minutes to hours of supercomputer time. Computational sciences and artificial intelligence, hence, are challenged to drive the engineering design optimization to both shortened design-cycle time and identification of creative new designs that are not only feasible, but also increasingly optimal with respect to some pre-defined criteria.

Conventional numerical optimization techniques are commonly used in engineering design process, instead of exhaustively enumerating the whole solution search space which is often computationally intractable. Typical conventional numerical optimization include steepest-descent, quadratic programming, pattern search, and linear approxima-

tion methods. One possible reason behind their wide adoption is the strong theoretical convergence assurance, at least to a local optimum. In addition, they perform extremely well in problems with convex landscapes. However, they are also subject to limitations. Most of these conventional optimization methods are deterministic, where a relatively good starting point must be carefully selected to obtain the global optimum, otherwise they only reach the local optimum. Moreover, the unavailability of accurate gradient information, noisy and multimodal landscapes may reduce their effectiveness in many real-world problems.

Fortunately, the last few decades have also been marked with many prominent advancements in the optimization study. Among those are a group of stochastic optimization algorithms inspired by Darwin's theory of evolution, collectively known as Evolutionary Algorithms (EAs). EAs have shown considerable success in locating the global optimum solution of optimization problems that may often be characterized by high dimensional, non-separable, multi-modal, constrained, and discontinuous/indifferentiable fitness landscape. However, thousands of calls to the analysis or simulation codes are often required to locate a near optimal solution in most conventional EAs, hence prohibiting their application on problems with time-consuming evaluations. The reasons for the high cost of evaluation and their effect on how many evaluations/generations can be afforded differ widely from one problem to another, as the following three examples from [139] may illustrate: *1)* When evolving controllers for a simulated robots, the fidelity of the physics simulator, the stochasticity in the system, and the desire to obtain robots that are robust to rare events may all play a part in making simulation times very long. *2)* When evolving a novel protein for a specific binding target by synthesis of proteins *in vitro* and their subsequent screening, thousands of proteins may be synthesized in parallel but each generation may take many hours to process, thus bringing about financial implications. *3)* When evolving a basic conceptual design for a new building, an architect

evaluating the designs would suffer from fatigue after several hours and eventually forced to stop.

To circumvent the aforementioned problems, approximation or surrogate models can be used to replace calls to the computationally expensive codes in the evolutionary search. Using surrogate models, the computational burden can be greatly reduced since the efforts involved in building the models and optimization using them are much lower than the standard approach of directly coupling the simulation codes with the optimizer. Nevertheless, this approach might not perform effectively and efficiently when the surrogate models are not properly managed. Some of the identified issues are:

- *Inaccuracy of the surrogate models.* It is one of many problems faced by most engineers and scientists due to lack of data or curse of dimensionality.

- *Search improvement through the use of imprecise surrogate models.* While focusing on improving accuracy, one might be ignorant of the more important aspect, *i.e.*, the fitness improvement which is possibly offered even by the imprecise surrogate models.

- *Suitability of the response surface approximation method to the problem in hand.* Depending on the complexity of a design problem, a single surrogate model that may have proven to be suitable on an instance might not work so well, or at all, on others.

- *Management of multi-scale computational models.* So far, majority of works concerning surrogate models in evolutionary computation have focused more on the use of data-fitting models. However, variable-fidelity multi-scale computational models often exist in many application domains which can also be utilized as cheaper fitness evaluator for more efficient evolutionary optimization.

- *Use of surrogate models in multi-objective evolutionary optimization.* While many real-world problems are naturally multi-objective, there is still little awareness on developing surrogate-assisted evolutionary frameworks which can be effortlessly scaled towards solving multi-objective problems.

As a result, there is an urgent need for the development of novel methodologies that efficiently and effectively manage the use of surrogate models in evolutionary optimization.

Meanwhile, EAs are also well-known for their ability to partition the population of individuals among multiple compute nodes seamlessly. Since the design optimization cycle time is directly proportional to the number of calls to the high-cost analysis codes, an intuitive way to reduce the total search time of evolutionary optimization algorithms is to parallelize the analysis of the design points. The benefits of combining the emerging Grid computing technology in the context of evolutionary design optimization are numerous. In particular, Grid provides the infrastructure to facilitate distributed computing and parallelism by tapping on vast compute power and a secure means of solving large-scale optimization problems. However, a very common problem in EA parallelization is that heterogeneity in the computing environment often exists, *e.g.*, CPU speed, network latency, memory capacity, and software platform. As such, any proposed parallelization frameworks must be designed to address the issue of variable simulation time due to the existing heterogeneity.

1.1 Research Objective

This thesis aims to address the different design issues arising from the management of surrogate models and parallel technologies for evolutionary optimization of computationally expensive problems. In particular, the research introduces and presents the methodologies and architectures for effective and efficient use of surrogate models and

parallel technologies in evolutionary optimization. Novel surrogate-assisted evolutionary schemes are subsequently proposed and developed for dealing with realistic optimization problems plagued with computationally expensive objective functions.

1.2 Core Contributions

Our contribution in the thesis begins with providing a comprehensive survey on existing state-of-the-art Surrogate-Assisted Evolutionary Algorithm (SAEA) optimization frameworks. From the survey, a taxonomy for SAEA is derived. Further, the effect of surrogate models on EA search convergence is analyzed mathematically.

In the research topic of SAEA using data-fitting models, a novel Generalized Surrogate Memetic (GSM) framework is proposed for both single and multi-objective optimization. Recent surrogate-assisted evolutionary frameworks have relied on the use of a variety of different modeling approaches to approximate the complex problem landscape. From these recent studies, one main research issue is with the choice of modeling scheme used, which has been found to affect the performance of evolutionary search significantly. Given that theoretical knowledge available for making a decision on an approximation model *a priori* is very much limited, a generalization of surrogate-assisted evolutionary frameworks for optimization of problems with objective(s) that are computationally expensive to evaluate, is proposed. The generalized evolutionary framework unifies diverse surrogate models synergistically in the evolutionary search. In particular, it focuses on attaining reliable search performance in the surrogate-assisted evolutionary framework by working on two major issues: *1*) to mitigate the *'curse of uncertainty'* robustly and, *2*) to benefit from the *'bless of uncertainty'*. The backbone of the generalized framework is a surrogate-assisted memetic algorithm that conducts simultaneous local searches using *ensemble* and *smoothing* surrogate models, with the aim of generating reliable fitness prediction and search improvements simultaneously. Empirical study on commonly used

single/multi-objective optimization benchmark problems and an aerodynamic airfoil design problem indicates that the generalized framework is capable of attaining reliable, high quality, and efficient performance under a limited computational budget.

In this thesis, evolutionary frameworks that leverage from the availability of multi-scale models are also proposed and studied on two real-world applications, particularly the water cluster structural optimization and the aerodynamic airfoil design. On the water problem, we demonstrate the management of two low-fidelity computational models in place of the high-fidelity first principle analysis code within the context of a Molecular Memetic Algorithm (MMA) for locating the global minimum energy structure. In MMA, it is shown that local optimization on the low-fidelity landscape brings the individual being optimized closer to the local optimum, hence benefitting the expensive high-fidelity evaluation conducted afterwards. Next, the airfoil problem demonstrates a classical example where data-centric multi-scale models can be created from a single computational model by using different residual tolerance. On such problems, we show that it is possible to enhance the evolutionary search using a Memetic Algorithm with Dynamic Fidelity Models (MA-DFM) framework. In particular, it decides at runtime, the appropriate fidelity level of the computational model, which is deemed to be computationally less expensive, to be used in place of the exact analysis code as the search progresses. Empirical study on both real-world problems reveals that evolutionary optimization utilizing multi-scale models serves as an efficient alternative approach for solving computationally expensive problems.

Last but not least, this thesis also presents the research work on parallelizing evolutionary optimization under a heterogeneous Grid computing environment to mitigate the issues relating to the intractable design life cycle of computationally expensive problems. In particular, a Grid-Enabled Parallel EA (GE-PEA) framework is proposed. The framework is developed using standard Grid technologies and has two distinctive features, *1*)

an extended Grid Remote Procedure Call (GridRPC) Application Programming Interface (API) to conceal the high complexity of Grid environment, and *2*) a metascheduler for seamless resource discovery and selection. To assess the practicality of the framework, theoretical analysis on the possible speed-up offered is presented. Empirical study on GE-PEA using a benchmark problem and a realistic aerodynamic airfoil optimization problem for diverse Grid environments having different communication protocols, cluster sizes, processing nodes, at geographically disparate locations also indicates that the proposed GE-PEA using Grid computing offers a credible framework for providing significant speed-up to evolutionary design optimization in science and engineering.

1.3 Organization of the Thesis

The remainder of this thesis is organized as follows:

- In Chapter 2, the fundamental notions in optimization, EAs, and relevant topics to this thesis, *i.e.*, Memetic Algorithms and Multi-Objective Evolutionary Algorithms are briefly described. Throughout this chapter, readers can grasp the background knowledge required for understanding further advanced topics in chapters 3-6.

- Chapter 3 presents a survey on major topics of this thesis, *i.e.*, Surrogate-assisted and parallel evolutionary computation. From a survey on SAEAs, a comprehensive taxonomy of the field with an accompanying analysis of the algorithms is proposed. In the rest of this chapter, a thorough review on parallel EAs and Grid computing technologies is also presented.

- Chapters 4-6 form the core of this thesis. Particularly, Chapter 4 presents the proposed optimization framework, labeled here as Generalized Surrogate Memetic framework, for solving both single and multi-objective problems. The empirical study on a set of benchmark problems and a real-world problem is also reported.

Chapter 5 presents the work on evolutionary optimization using multi-scale computational models, with application to two real-world applications. Particularly, a Molecular Memetic Algorithm for the water cluster structural optimization and Memetic Algorithm with Dynamic Fidelity Models framework for the aerodynamic airfoil design, are presented. Our work on parallel evolutionary optimization under a heterogeneous Grid computing environment is then presented in Chapter 6. This includes the theoretical analysis and empirical study on the proposed Grid-enabled Parallel EA.

- Finally, Chapter 7 concludes this thesis and outlines possible future research directions in the area of evolutionary optimization for computationally expensive problems.

Chapter 2

Background

This chapter begins by defining the notion of optimization, followed by a brief introduction to Evolutionary Algorithms (EAs). Subsequently, a brief overview of Memetic Algorithms (MAs) and Multi-Objective Evolutionary Algorithms (MOEAs) are also presented. This serves to equip the reader with some fundamental background information necessary for better understanding of the advanced topics in Chapters 3-6.

2.1 Optimization

Searching for optimality has been one of the most fundamental principles of life. Take for example, engineers working in aerospace industry strive to obtain a set of optimal decision variables resulting in aerodynamic designs while economists make every effort to maximize the profit with minimum cost. Having dealt with it in our everyday life, it is no surprise that optimization study is one of the oldest and still uprising sciences in this world. The term *'optimization'* can be literally defined as the process of finding the best solution (also termed as global optimum) for a given problem. Mathematically, optimization, for instance minimization of a d-dimensional problem, can be formulated as finding the decision variable, $\mathbf{x}^*$ which gives the minimum output for a given objective function $f(\mathbf{x}) : \mathbb{R}^d \rightarrow \mathbb{R}$, *i.e.*:

$$\forall \mathbf{x} : f(\mathbf{x}^*) \leq f(\mathbf{x}) \qquad \text{(Eq. 2.1)}$$

2.1.1 Fundamentals

In practice, optimization could be a complicated process due to the fact that most real-world optimization problems are composed of many components to be considered, as follows:

- **Decision variables.** The decision variables (or design vector/variables) refer to the values to be chosen in an optimization problem. They are often multi-dimensional, could be in continuous/real number and/or discrete domains, and bounded and/or unbounded in the search space. A vector of decision variables $\mathbf{x}$, is often represented by: $\mathbf{x} = [x_1, x_2, \ldots, x_d]^T$, where d is the problem dimensionality. In optimization studies, the term *'continuous/real-parameter optimization'* and *'combinatorial optimization'* are often used referring to the optimization involving real number and discrete search spaces, respectively.

- **Objective functions.** This is necessary to measure the quality of every candidate solution, which is represented by a set of decision variables. Mathematically, it is often simply represented by $f(\mathbf{x})$. In many real-world problems, difficulty could arise due to fact that there might be more than one objective, conflicting each other. This, however, is no longer of much concern due to recent developments in Multi-Objective Optimization (MOO) studies [123][33]. There also exist many real-world problems (*e.g.*, Travelling Salesman Problem), which could be categorized as *'NP-complete'*, where *'NP'* stands for non-deterministic polynomial, *i.e.*, there is currently no deterministic algorithm known to solve the problem in polynomial time. Further, when dynamic and uncertain environment is concerned, the objective functions could vary with time, embedded with approximation errors, or even require many extra evaluations for robustness purpose [25][144]. From practical point of view, a great concern also lies in the fact that most real-world problems

are plagued with computationally expensive objective functions, taking minutes to hours of supercomputing power to compute [142]. Hence, this raises a great challenge to design an effective and yet efficient optimization algorithm.

- **Constraints.** This refers to any restrictions imposed by the optimization problem and must be satisfied by a candidate solution to be acceptable. Generally, these constraints can be formulated as mathematical inequalities and equalities: $g_i(\mathbf{x}) \geq 0, i = 1, \ldots, k$ and $h_j(\mathbf{x}) = 0, j = 1, \ldots, l$ where k and l denote the number of inequality and equality constraints, respectively [122]. In other words, one might also interpret constraints as objective functions which only need to be satisfied rather than to be optimized. For instance, one of the most common constraints often applies in many optimization problems is the bound constraints on the decision variables, *i.e.*, $x_i^l \leq x_i \leq x_i^u$ where x_i^l and x_i^u are the lower and upper bound of x_i, respectively. Note that the computational budget allowed, such as the number of maximum function evaluations, CPU time, or wall-clock time, is also a form of optimization constraint though often not explicitly expressed.

Throughout this thesis, unless otherwise specified, computationally expensive[1] nonlinear programming problems under limited computational budget with bound constraints of the following form, is considered:

$$\begin{aligned} \text{minimize:} \quad & f(\mathbf{x}) \\ \text{subject to:} \quad & x_i^l \leq x_i \leq x_i^u, \end{aligned} \tag{Eq. 2.2}$$

[1]Note that throughout this thesis, the notion of *'computationally expensive'* problems refers to those problems which require many minutes to hours of supercomputer time to compute a single evaluation. Nevertheless, whenever necessary, computationally cheap benchmark problems (Appendix B.1) with various characteristics might be used, besides the computationally expensive real-world problems, for empirical study purpose.

or in multi-objective optimization:

$$\begin{aligned} \text{minimize:} \quad & f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_r(\mathbf{x}) \\ \text{subject to:} \quad & x_i^l \leq x_i \leq x_i^u, \end{aligned} \tag{Eq. 2.3}$$

where $i = 1, 2, \ldots, d$, d is the dimensionality of the search problem, r is the number of objective functions, and x_i^l, x_i^u are the lower and upper bounds of the i^{th} dimension of vector $\mathbf{x}$, respectively.

2.1.2 General Approaches

In the past decades, a plethora of optimization/search methods have been introduced in the literature, see for example [48][33][188][204]. Generally, most taxonomies found in the literature have classified the different methods based on *1*) whether stochastic/random elements subsist, *i.e.*, deterministic and stochastic(probabilistic), or *2*) whether derivative information is required, *i.e.*, direct and indirect search.

In deterministic optimization, no random elements are involved [188]. For instance, those under this category are deterministic hill-climbing, branch and bound, and steepest descent methods. On the other hand, stochastic optimization incorporates probabilistic (random) elements, besides the available information. Examples of stochastic optimization methods are Random Walk, Tabu Search, and Simulated Annealing. The main purpose of introducing randomness is due to the fact that deterministic methods are often flawed by the multimodal and noisy landscapes, and hence often can only arrive at local optima.

Direct and indirect searches classification, on the other hand, concentrates on the issue of whether derivatives of the objective functions are required. Direct search methods, also often known as the zeroth order methods, are often useful when derivatives, such as gradients or Hessians of the objective functions, are unavailable or unreliable [99]. Hence,

the main feature of direct search lies in the need for only the objective function values or even just the relative ranks. Some examples of methods under this category are Pattern Search, Lagrange Interpolation, and Simplex search. Indirect searches can be further divided into those using gradient information (first order methods) or hessian information (second order methods). In first order indirect search, only function values and first derivative vector are utilized [131]. It includes Steepest Descent, Conjugate Gradients, and Quasi-Newton methods. Second order indirect search methods, on the other hand, require the function values, its first derivative vector, and the second derivative matrix. Newton-Raphson method [100] is an example of this kind.

2.2 Evolutionary Algorithms

Inspired by the Darwinian's survival of the fittest principle from 1800s, the basic idea of EAs is to evolve a population of individuals, where each of the population members represents a candidate solution to a given problem. Descendants of the current population are generated by randomized processes to imitate the natural process of mutation and crossover (also often termed as recombination). A measure of quality or fitness value is then assigned to each of these individuals by means of a fitness function. Subsequently, a selection process is applied favoring better individuals to reproduce and proceed to next generation. The different artificial operators, *i.e.*, crossover, mutation, and selection, are generally termed as evolutionary operators. Basically, any algorithms with similar pattern to that depicted in Fig. 2.1, can be categorized under the umbrella of EAs.

According to the aforementioned discussions, it is obvious that EAs which depend heavily on stochastic evolutionary operators, belong to the stochastic optimization group. It is also worth noting that while EAs are often considered as direct search methods, since only the objective function values are required, [172] suggests an interesting proposition that EAs do have resemblance to gradient-based methods to certain extent. This is so

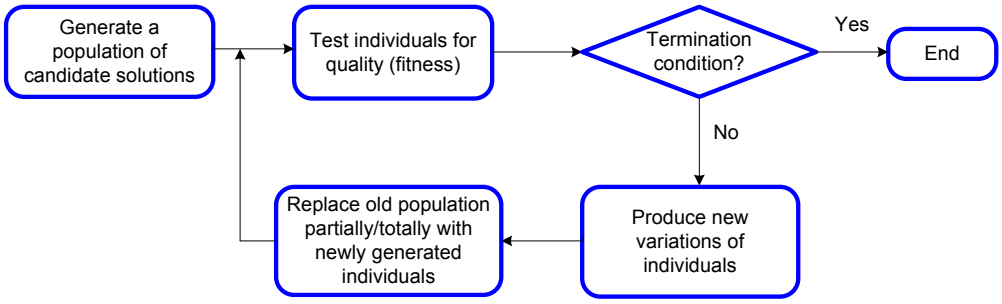


Figure 2.1: Workflow of Evolutionary Algorithms.

since EAs can be considered as approximating the gradient of an individual based on the fitness of generated offspring via evolutionary operators. Based on this proposition, an algorithm called Evolutionary Gradient Search (EGS) has also been reported in [172].

Since its emergence in 1960s, there have been different mainstreams of EAs founded by different groups of researchers. The first few dialects of EAs, are namely *1*) Genetic Algorithm (GA) [79][63] by Holland in the University of Michigan, Ann Arbor, *2*) Evolution Strategy (ES) [161][162][184] by Rechenberg and his colleagues in Berlin, and 3) Evolutionary Programming (EP) [51] by Fogel in San Diego, California. While they are similar in terms of the basic principle of evolution as explained above and depicted in Fig. 2.1, the differences are as follows:

- In GA, high emphasis is placed on the use of crossover as the main evolutionary operator. In the implementation, crossover is applied with high probability while mutation is only applied occasionally as its probability is set to a low value. As for the selection operator, typically, proportional selection which is probabilistic in nature is used. In its early versions, binary encoding is also often used which is no longer the case in contemporary GAs. A pseudocode of GA is presented in Algorithm 1.

- ES was originally developed for optimizing real-valued vectors. In the implementation of ES, recombination and normally distributed mutation operators for both the

Algorithm 1 Genetic Algorithm

1: **Initialization**: Generate and evaluate a population of design vectors, $P(t)$, for $t = 0$.
2: **while** termination condition is not satisfied **do**
3: Apply crossover and mutation operators on $P(t)$ to create the offspring population, $C(t)$.
4: Evaluate $C(t)$.
5: Apply selection operator on $P(t)$ and $C(t)$ to create a new parental population, $P(t+1)$.
6: $t = t + 1$
7: **end while**

design vectors and strategy parameters are employed. The latter is often termed as self-adaptation. The selection operator used is deterministic, either plus $(\mu + \lambda)$ or comma (μ, λ) strategy. The plus strategy differs from its comma counterpart as it includes the parental population into the selection, hence possibly preserves the top individuals in the population (elitism). More details are described in Algorithm 2.

Algorithm 2 Evolution Strategy

1: **Initialization**: Generate and evaluate a population of design vectors and their individual mutation strengths, $P(t)$, for $t = 0$, and population size μ.
2: **while** termination condition is not satisfied **do**
3: Apply crossover and mutation operators on the decision variables and mutation strengths of $P(t)$ to create the offspring population, $C(t)$ with size of λ.
4: Evaluate $C(t)$.
5: Apply selection operator on $C(t)$ (and $P(t)$ for $(\mu + \lambda)$) to to create a new parental population, $P(t + 1)$ of size μ.
6: $t = t + 1$
7: **end while**

- EP was originally developed for evolving finite-state machines, but have since been reported for optimizing real-valued vectors. It emphasizes the use of mutation without any crossover operators. The original EP is described in Algorithm 3. In its extended version, EP also uses normally distributed mutation on the design vectors and strategy parameters, and hence becoming harder to distinguish from its ES counterpart. One noticeable difference, however, is on the selection operator. Typically, tournament selection is used where each individual is evaluated against

q others chosen randomly. The μ individuals with the greatest number of wins are selected to be the parents of the next generation.

Algorithm 3 Evolutionary Programming

1: **Initialization**: Generate and evaluate a population of design vectors and their individual mutation strengths, $P(t)$, for $t = 0$, and population size μ.
2: **while** termination condition is not satisfied **do**
3: Apply mutation operator on the decision variables and mutation strengths of $P(t)$ to create the offspring population, $C(t)$ with size of μ.
4: Evaluate $C(t)$.
5: Apply selection operator on $C(t)$ and $P(t)$ $(\mu+\mu)$ to to create a new parental population, $P(t+1)$ of size μ.
6: $t = t + 1$
7: **end while**

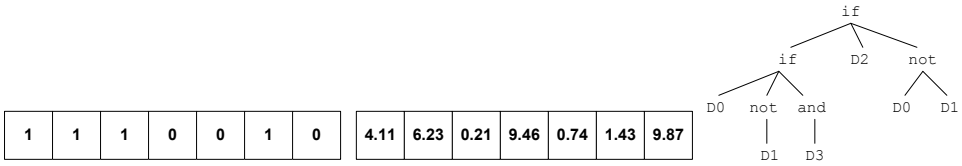
As the field continues to expand, a myriad of new paradigms such as Genetic Programming (GP) [106] by Koza, Messy GA [64] by Goldberg, Differential Evolution (DE) by Storn & Price [190], and various considerable hybrid between EAs and other research fields have emerged [82][137][104][132][133]. More details on the development of EAs can also be found in [19][50]. Often, the fields of evolutionary computing, neural networks, and fuzzy logic, are collectively referred as *'soft computing'*, or more broadly as *'computational intelligence'*.

2.2.1 Fundamentals

The first issue to consider when using EA for optimization is the representation of candidate solutions. After solutions are encoded into their representation, evolutionary operators are conducted to obtain varied solutions. Candidate solutions are then evaluated and put into selection pool for next generation. In this part of the thesis, a brief overview on the basic elements of an EA, *i.e.*: representation, crossover, mutation, and selection, is provided. Note that fitness/objective function represents the other core element of an EA. Since this has been discussed in Section 2.1.1, it will not be repeated here.

2.2.1.1 Representation

Besides evolutionary operators, the success of an EA is very much affected also by the representation or encoding used. A good representation scheme enables an instantiation of the solution for the underlying problem which could be completely realizable when put into practice [33]. Different representation schemes have evolved in the EA community. For instance, early GAs have mostly used binary string representation, ES and EP use a string of real-valued decision variables combined with their strategy parameters for self-adaptation purpose, while GP is commonly implemented using tree structures of computer programs. Examples of the binary, real, and tree representations of an indi-



2.2.a: Binary representation

2.2.b: Real representation

2.2.c: Tree representation

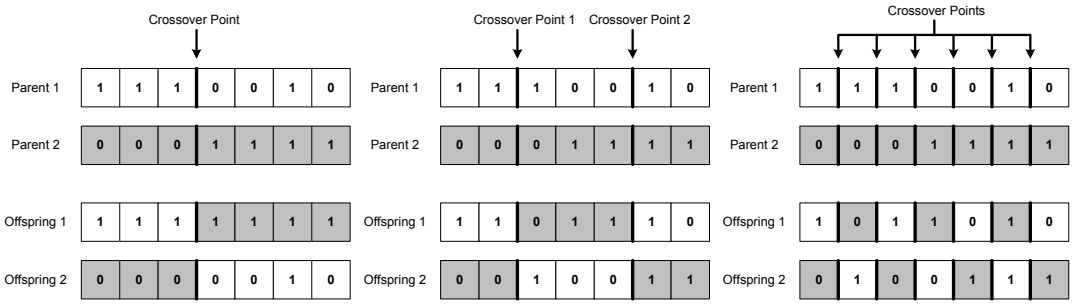
Figure 2.2: Different representation schemes used in EA.

vidual in EA are depicted in Fig. 2.2.a, 2.2.b, and 2.2.c, respectively. However, most of the representations used in todays applications have varied significantly to accommodate the different needs. For instance, many GAs are now implemented in real-valued decision variables, especially in engineering design problems as real-valued decision variables are quite common in this area. It is also worth noting that in practice, there are many problems which require a mix of different representations, *e.g.*, some decision variables are encoded as real-valued while others use integers.

2.2.1.2 Crossover

Crossover or recombination operator imitates a complex process, which occurs between pairs of chromosomes in the cell division phase of living beings. This operator randomly

chooses one or more locations (loci) of breakage and exchanges the subsequences before and after the loci between two individuals to create two new offspring. In GA, crossover is the main operator which is usually applied with a high probability. In ES, crossover is also commonly used though it is not the main operator as the method emphasizes more on the self-adaptation of the mutation strength. Meanwhile, in EP, crossover is not used. In GA, a random number, r_c is generated in the range between 0 and 1, and the individuals undergo crossover if $r_c \leq p_c$, where p_c is the user-defined crossover rate. Some well-known crossover types are: 1-point crossover, n-point crossover, and uniform



2.3.a: 1-point crossover 2.3.b: 2-point crossover 2.3.c: uniform crossover

Figure 2.3: Example of different types of crossover in EA.

crossover. While the first two crossover schemes pre-define the chromosomes' breakages (*i.e.*, either 1 or n), uniform crossover permits independent breakage for each allele. An illustration for each of the crossover operator is also depicted in Fig. 2.3. In uniform crossover, there is a maximum of $L-1$ break points for a string of length L. In further development, many researchers have come up with different crossover operators for real-valued design variables, such as geometrical, simplex, fitness-based scan, and diagonal multi-parent crossover [18].

2.2.1.3 Mutation

The mutation operator mimics nature's ability to adapt to changes in environmental conditions so as to improve the chances of survival. In GA, mutation is merely a secondary operator due to the low rate of occurrence used, but it serves as the core operator in both ES and EP. The mutation operation of binary-coded GA is illustrated in Fig. 2.4. Random number, $r_m \in [0, 1]$ is generated for each bit location, which undergoes mutation if $r_m \leq p_m$, where p_m is the user-defined mutation rate. In ES and EP, the

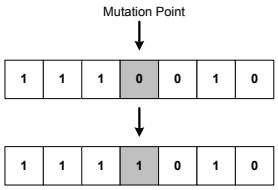


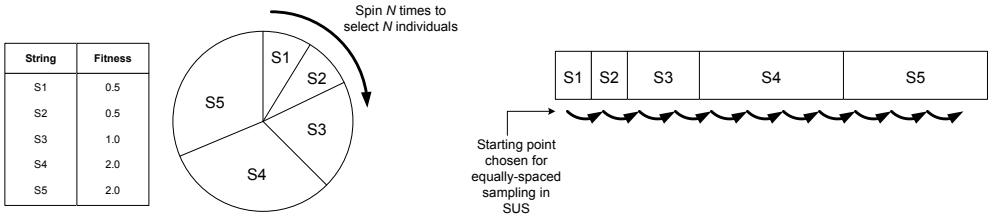
Figure 2.4: Example of bit-flip mutation in EA.

mutation rate is also encoded as part of the chromosomes and self-adapts by undergoing crossover and mutation, like all other decision variables. While the correct setting for p_m is problem-dependent, many researchers have recommendation for $p_m = 1/L$, where L is the chromosome length [18].

2.2.1.4 Selection

The selection operator chooses individuals in the population for reproduction where fitter individuals are given more chance to reproduce. Many variants of selection scheme have been proposed in the last decades. The canonical GA first introduces fitness proportionate selection scheme where the survival chance of an individual is defined by the ratio of its fitness value over the average fitness of the entire population. A typical implementation of the fitness proportionate selection scheme is using the *'Roulette Wheel'* (RW) method (see Fig. 2.5.a). Nevertheless, the RW method is often criticized for easily losing diversity since the selection is biased towards the larger portion of the wheel. The other

commonly used selection operator is *'Stochastic Universal Sampling'* (SUS), which does not exposed to the limitation faced by the RW method. In SUS, fitness values of the population are mapped linearly as depicted in Figure 2.5.b. Equally-spaced samples are taken in the linear map to form population of the next generation. Besides RW and SUS method, ranking selection [20] is also a well-known alternative selection operator, whose purpose is to prevent issues arising from non-scaled fitness values.



2.5.a: Roulette Wheel Selection 2.5.b: Stochastic Universal Selection

Figure 2.5: Examples of Roulette Wheel and Stochastic Universal Selection in EA.

2.2.2 Memetic Algorithms

Memetic Algorithms (MAs) are population-based meta-heuristic search methods that are inspired by Darwin's principle of natural evolution and Dawkin's notion of a meme defined as a unit of cultural evolution capable of local refinements [137]. While EAs are generally known to be capable of exploring and exploiting promising regions of the search space, they can take a relatively long time to locate the exact local optimum with high precision. In [195], it is reported that two competing goals, *i.e.*, *'exploration'* and *'exploitation'* govern the design of most global search methods. Exploration is necessary to ensure global reliability by searching through the whole search space and hence provides a reliable estimate of the global optimum. On the other hand, exploitation is also important to refine the best solutions found so far to their better quality counterparts, possibly located in the immediate neighborhood via local search. The local search

Algorithm 4 Memetic Algorithm (for SOO)

```
1: Initialization: Generate and evaluate a population of design vectors.
2: while termination condition is not satisfied do
3:    Apply crossover and/or mutation operator(s) to create the offspring population.
4:
5:    for each individual x in the offspring population do
6:       Apply local search to find an improved solution, x_opt.
7:       Perform replacement using Lamarckian/Baldwinian learning, i.e.,
8:       if Lamarckian Learning then
9:          if f(x_opt) < f(x) then
10:            x = x_opt, which is inclusive of
11:            f(x) = f(x_opt)
12:         end if
13:      else if Baldwinian Learning then
14:         if f(x_opt) < f(x) then
15:            f(x) = f(x_opt)
16:         end if
17:      end if
18:   end for
19:
20:   Select a new parental population
21: end while
```

method here includes any other optimization algorithm applied to refine the quality of an individual/chromosome. It could materialize in the form of deterministic gradient-based, stochastic, or other methods. The brief outline of a traditional MA for Single Objective Optimization (SOO) is provided in Algorithm 4.

From the pseudocode, it is noted that improved solution generated by the local search procedure in line 8, replaces the genotype and/or fitness of the original individual in the population. There are two basic replacement strategies that are commonly used in MAs [140]:

- *Lamarckian learning* forces the genotype to reflect the result of improvement in local search by placing the locally improved individual back into the population to compete for reproductive opportunities.

- *Baldwinian learning* only alters the fitness of the individuals and the improved

genotype is not encoded back into the population.

In this thesis, the focus of our research is on Lamarckian learning unless otherwise stated. It is worth noting that in [144], Baldwinian learning has been shown to be more suitable when considering the search for solutions which are robust to slight deviations in design variables and/or environmental parameters. Since, searching for such solutions is out of the scope of this thesis, we choose to apply Lamarckian learning which is rather suitable for finding global optimum solutions.

2.2.3 Multi-Objective Evolutionary Algorithms

Multi-criteria or Multi-Objective Optimization (MOO) problems arise naturally in many real-world situations. Take for instance, while automotive engineers might want to produce a vehicle equipped with many enhanced and expensive technologies, they must also minimize the production cost. Hence, in decision making, it is inevitable to seek for a compromise between the various objectives or goals. EAs as a class of population-based optimization algorithms suit the need for tackling this type of problems. This is so since in MOO, the goal is not to obtain a single global optimum solution as in its SOO counterpart, but to obtain a set of Pareto optimal solutions with the trade-off between different objectives. Using EAs, these solutions can be obtained within a single run.

Algorithm 5 Nondominated Sorting Genetic Algorithm II (NSGA-II)

1: **Initialization**: Generate and evaluate a population of design vectors, assign rank based on Pareto dominance.
2: **while** termination condition is not satisfied **do**
3: Apply evolutionary operator(s) to create the offspring population.
4: Assign domination rank on both parent and offspring populations.
5: Generate sets of non-dominated fronts.
6: Determine the crowding distance between individuals on each front.
7: Select individuals on the lower fronts (for convergence) and possess better crowding distance (for diversity), as new parental population.
8: **end while**

In MOO, a solution $\mathbf{x}^{(1)}$ is said to dominate solution $\mathbf{x}^{(2)}$ in the objective space, *i.e.*, $\mathbf{x}^{(1)} \preceq \mathbf{x}^{(2)}$ if the following two conditions hold:

- $\mathbf{x}^{(1)}$ is no worse than $\mathbf{x}^{(2)}$ on all objectives or $f_j(\mathbf{x}^{(1)}) \leq f_j(\mathbf{x}^{(2)})$ for all $j = 1, 2, \ldots, r$.
- $\mathbf{x}^{(1)}$ is strictly better than $\mathbf{x}^{(2)}$ on at least one objective, or $f_j(\mathbf{x}^{(1)}) < f_j(\mathbf{x}^{(2)})$ for at least one $j \in 1, 2, \ldots, r$

If set P is the entire feasible search space, the non-dominated set P^* is labeled as the *Pareto-optimal set*. Any two solutions in P^* must non-dominate each other, *i.e.*, $\mathbf{x}^{(1)} \sim \mathbf{x}^{(2)}$. On the other hand, Pareto front (PF^*) is the image of the Pareto-optimal set in objective space.

Among the well-known Multi-Objective EAs (MOEAs) proposed by various research groups are the Vector Evaluated GA (VEGA)[180] by Schaffer (1985), Multi-Objective GA (MOGA)[52] by Fonseca and Fleming (1993), Non-dominated Sorting GA (NSGA)[189] and NSGA-II[36] by Deb *et al.* (1994, 2000), Niched-Pareto GA (NPGA)[81] by Horn and Nafpliotis (1993), NPGA-II[46] by Erickson *et al.* (2001), Strength Pareto EA (SPEA)[215] and SPEA-II[213] by Zitzler *et al.* (1999, 2001), and Pareto Archived ES (PAES)[103] by Knowles and Corne (2000). From the literature of MOEAs, NSGA-II represents one of the most widely used and hence it is briefly outlined here as Algorithm 5. Note that this algorithm will be revisited in Chapter 4.

Chapter 3

Surrogate Modeling and Parallelization

This chapter presents our survey on two main topics related to this thesis, *i.e.*, *1*) surrogate modeling and *2*)parallelization, in the context of evolutionary computation.

3.1 Surrogate Modeling in Evolutionary Computation

Over the years, while there remains many emerging works on developing new optimization algorithms, there are numerous other efforts gearing towards improving the efficiency of existing algorithms. One of such research direction is on the use of surrogate models that are less computationally intensive. From the survey of literature, we note the significant number of studies on using surrogate models to assist optimization in recent years. Fig. 3.1 quantifies the number of publications on the topic of surrogate-assisted optimization in multi-disciplinary study, as listed in *Scopus*, which represents one of the largest database of research literatures. The data consists of ∼1078 publications from more than 160 researchers over 25 disciplines of study in the past 10 years, *i.e.*, 1998-2007. This constantly rising trend indicates the importance of surrogate-assisted optimization research and highlights its success in practice. In the context of surrogate-assisted evolutionary optimization, a similar trend is observed, where the number of publications

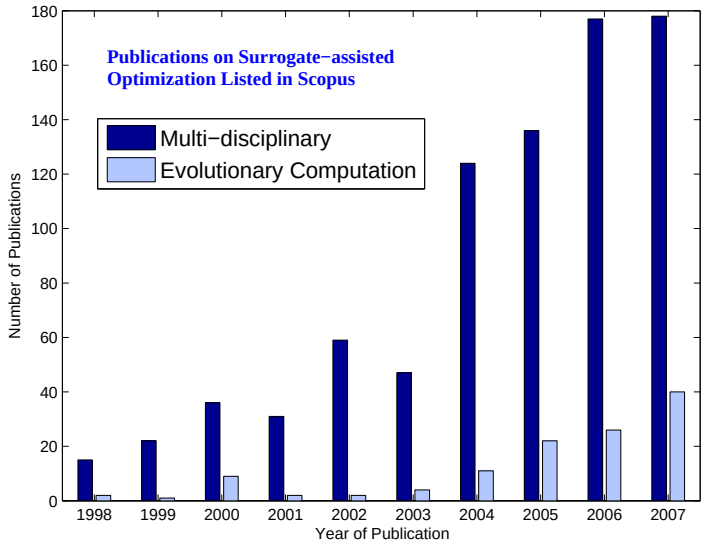


Figure 3.1: Publications on the topic of Surrogate-assisted optimization in the past 10 years (1998-2007).

specializing on surrogate-assisted evolutionary approaches, encompasses ∼20% from the total 1078 publications at the time of writing this thesis. In the rest of this chapter, various related aspects to SAEA (as summarized in Fig. 3.2), such as the motivations for using surrogate models, types of surrogate models used, surrogate model incorporation techniques, convergence property, and application/problem domains in which SAEAs have been reported, are discussed.

3.1.1 Motivations

The use of surrogate models in EA can be motivated by several reasons [86]:

- **The computation of the fitness function is time-consuming**. Many complex engineering design problems where high-fidelity models are used, each objective function evaluation requires the simulation of the high-fidelity models, such as Finite Element Analysis (FEA), Computational Fluid Dynamics (CFD) or Computational Electro Magnetics (CEM), *etc.*, may cost minutes to hours of supercomputer time.

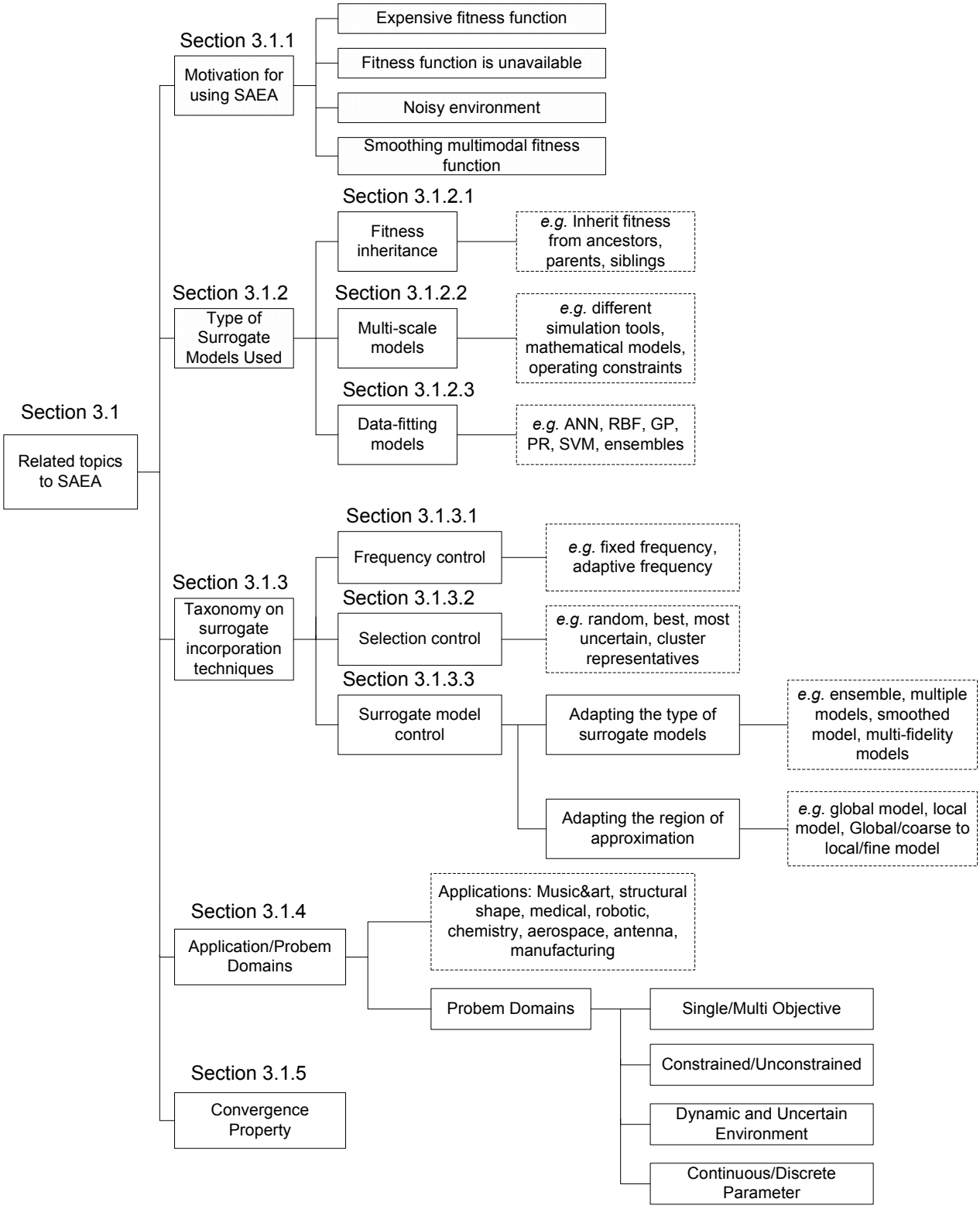


Figure 3.2: Various related topics to SAEA discussed in Section 3.1.

- **There is no explicit model for the fitness function**. This often occurs in evolutionary music and art designs where the fitness of an individual design totally depends on the human users which are susceptible to fatigue.

- **The environment is noisy**. In noisy environment, robust optimum solutions are often more desirable. However, this type of solutions come with a trade-off. Many extra evaluations are generally required to obtain the effective fitness in noisy environment, absorbing the possible perturbations experienced in real-life scenario.

- **The fitness landscape is highly multimodal**. Multimodality can often deceive optimization algorithms to arrive at local optimum instead of the desired global optimum. Some approximation models such as Polynomial Regression (PR) can be constructed to smooth the rugged landscape without much changes to the location of the global optimum.

3.1.2 Surrogate Models

In this section, a survey of the various approximation models commonly used in the context of SAEA is presented. In particular, they are categorized into three main categories, *i.e.*, *1*) fitness inheritance/estimation, and *2*) multi-scale computational models, and *3*) data-fitting surrogate models.

3.1.2.1 Fitness Inheritance/Estimation

This type of approximation obtains the fitness of an individual in the EA population using linear or non-linear combination of previously evaluated individuals, which can be the ancestors, parents, or even siblings. In [185][206][178], average or weighted fitness from the two parents are used to estimate the fitness of an offspring individual. Similarly, the fitness inheritance approach in [171] considers both parents to estimate the offspring's fitness via some arithmetic operations on their fitness and reliability measures. In [42],

representative siblings chosen from clustering are considered to derive the fitness of an offspring. In the context of EA optimization in noisy environment, [28] uses the fitness and variance average of parents to decide whether multiple samplings are required for determining the effective fitness of an individual. Meanwhile still in same context, [25] uses all previously evaluated individuals, *i.e.*, the ancestors are used for estimating the abundant fitness evaluations required in finding solutions which are robust to noise. In the context of Estimation of Distribution Algorithm (EDA), [150] and [179] consider all previously evaluated individuals to build the probabilistic models used for sampling new individuals. Similarly, [165] and [166] have also used weighted average fitness and nearest neighbours' fitness values, respectively, in Particle Swarm Optimization (PSO).

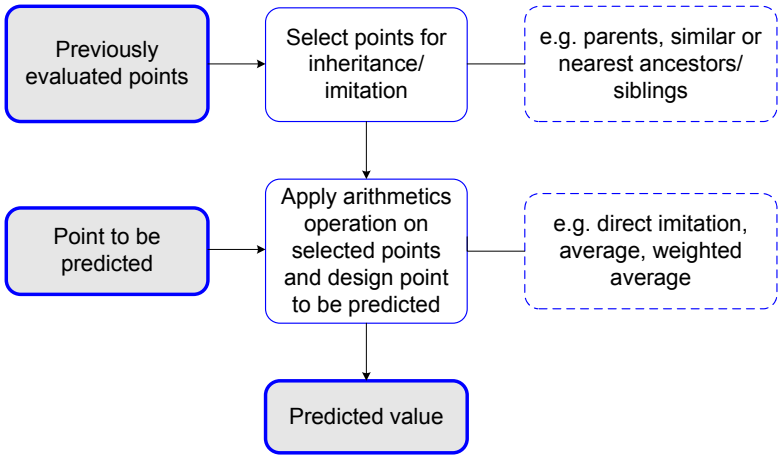


Figure 3.3: Fitness inheritance.

3.1.2.2 Multi-scale Computational Models

In science and engineering, computational models are commonly created to simulate and analyze a set of processes or phenomena observed in the physical system which are otherwise deemed to be too costly to construct. For instance, in water cluster optimization, there are low-fidelity empirical models such as OSS2 [135] or TTM2.1F [30], besides the first principle quantum mechanical computation such as Density Functional Theory

(DFT) [167]. The term *'fidelity'* here refers to the extent to which a model is capable to mimic the original physical system of interest. A common assumption is that higher fidelity models are generally more accurate at the expense of a higher computational cost. The complexity or level of details of a physical system may come in many forms. Some of those reported in the context of SAEA use lower fidelity models ranging from variable mathematical models [41], variable parametric formulations [39][160][147][97], to variable operating conditions [43].

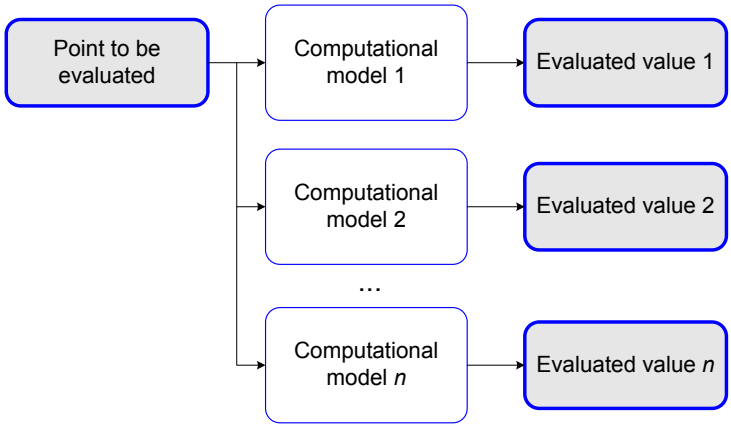


Figure 3.4: Multi-scale computational model.

3.1.2.3 Data-fitting Models

Another widely used approximation method in SAEA is the data fitting techniques, such as Artificial Neural Networks (ANN), Kriging/Gaussian Process (GP), Polynomial Regression (PR), Radial Basis Function (RBF), and Support Vector Machine (SVM). Since all these techniques actually model the physical models, they are also often known as *'metamodels'*. The common idea behind this type of surrogate model is to train or compute a set of model parameters such that error observed in the training and/or testing dataset is minimized. A detailed description of three metamodels used in later chapters, *i.e.*, Kriging/GP, PR, and RBF, is also presented in Appendix A. Reported SAEA works using the various data fitting techniques in the literature can be listed as follows:

- **ANN**: [94][47][92][126][80][88][58][68][37][71];
- **Kriging/GP**: [158][159][45][197][27][208][210][105][102][44][209][207][187];
- **RBF**: [142][208][141][144][209][207];
- **PR**: [110][111][208][210][146][209][207][112];
- **SVM**: [7][198][115];
- **Multiple models**: [207];
- **Ensemble models**: [93][73];

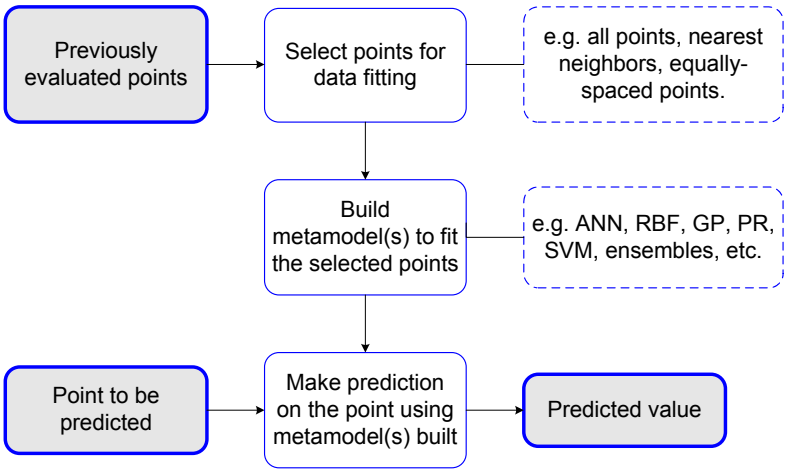


Figure 3.5: Data-fitting model.

3.1.3 Surrogate Models Management Frameworks

The use of surrogate models often brings about approximation errors, which can subsequently lead to false convergence in optimization. Hence, the challenge in most SAEA is to maintain convergence of the underlying EA to near-optimum solution while minimizing the computational budget as much as possible. Taking this cue, most SAEAs

introduced can be categorized based on the mechanisms by which the fitness approximation is incorporated, into: *1) frequency control strategies* , *2) selection control strategies*, and *3) model control strategies.* It is worth noting that in [92], the term *'evolution control'* is introduced referring to the different model management strategies that adapt the frequency and/or the selection of individuals to be evaluated based on the exact function or surrogate model.

3.1.3.1 Frequency Control

To decide on the frequency of performing exact function evaluations, some researchers have considered a fixed portion of individuals and/or generation count that would undergo exact function evaluation, such as in [25], [158], [29], and [92]. Meanwhile, [68] considered the approximation error and correlation measure to determine the quality of the surrogate model, which in turn is used to define the frequency of approximated individuals in the next generation. As far as frequency control is concerned, it is also worth noting the notions of *'pre-selection'* [197][198][68] and *'informed operator'* [157][43]. In both schemes, λ out of θ individuals are identified based on prediction of surrogate models (where λ is the population size and $\lambda < \theta$).

3.1.3.2 Selection control

The frequency adaptation in SAEA often comes in tandem with some rules for selecting which individuals are to be exact evaluated. The simplest of them is by random selection [92]. Others choose the most promising individuals in the surrogate model to be exact-evaluated. However, note that the notion of *'most promising'* may have diverse interpretations. In [127][111][157][45][92][85][197][198][98], individuals having the best approximated fitness are chosen to be exact-evaluated. Others working along this line considered the most uncertain individuals as also promising [25][47][197][102][187]. The aim of re-evaluating individuals with the most uncertain approximation values using

the exact fitness function is to reduce the uncertainty in the respective region. At the same time, there is also higher probability that this individual is of good quality since confidence on the current prediction is low. In [101][42][93][68], greater diversity in the selected individuals is expected by only exact evaluating the cluster centres.

3.1.3.3 Surrogate Model Control

Besides frequency and selection control, many works in this area have also approached the SAEA efficiency issue through the type of surrogate model used. Note that surrogate model control may be in the form of using variable-fidelity models and/or different region of approximation. Most of the schemes which employ multi-scale models in the context of parallel island GA [39] and [97], use differing surrogate models in the different islands. Meanwhile, as far as the region of approximation is concerned, many works have also considered local models. For instance, [138][142][163][97][141][144][207][71] use only local approximation models instead of the conventional way of using single global approximation for all the fitness predictions. Others in [126][7][147][208][37][209] considered using a hierarchical architecture from a global/coarse-grained to local/fine-grained as the search progresses. Some interesting works in [58] and [186] adopted an inverse modeling approach where the Pareto front (in the context of multi-objective optimization), instead of the decision search space is modeled. Note that though most of the techniques grouped into this category emphasize on a particular surrogate model control, often they are fused with some other forms of frequency/selection/surrogate model controls.

3.1.4 Application and Problem Domains

From a survey of the literature, it is interesting to note that SAEAs have enjoyed great successes in many application domains, and found their contributions in new problem domains including multi-objective, constrained, dynamic environment, and discrete problems. Some of the applications where SAEAs have been reported range from manu-

facturing [202], structural engineering [148][149][187], medical [183][182][181], aerospace [39][152][68][45][60][61][91][92][136][142][144][157], robotic [116], chemistry [42][43], antenna [147], to music and art [23][94][95]. Meanwhile, the diverse problem domains in which SAEA has been reported include constrained problems [158][85][164][44][71], multi-objective problems [47][85][98][58][44][96][105][102][126][186][166], discrete parameter problems [150], and dynamic & uncertain environment [25][176][144][146].

3.1.5 Convergence Analysis

In this subsection, the convergence property of the SAEAs is discussed. Hitherto, there are several empirical or experiment-based proofs reported for such SAEAs. In [185], it is reported that evolutionary search converges well even when only 10% of the population are evaluated using the exact function evaluations. Similarly, in [92], it is observed that EA can achieve correct convergence if more than 50% of the function evaluations are carried out using the exact fitness function. However, to the best of our knowledge, there has not been any universal framework that theoretically justifies those empirical findings. To begin, the plethora of existing SAEAs reviewed in the previous section are generalized into their generic framework as described in Algorithm 6. The main idea in this framework is to interleave expensive function evaluations and surrogate model evaluations, with some pre-defined or adaptively controlled proportion, in both global and local search (memetic) phases. It is obvious that as the proportion of exact function evaluations approaches 100%, the resultant algorithm is close to the original underlying EA.

Here, a theoretical analysis on the convergence property of general SAEAs is presented using the Finite Markov Chain (FMC) theory by extending from previous efforts on the convergence analysis of genetic algorithms in [40][169][192][137]. In FMC theory, a random process is represented by the transitions from/to a finite number of states,

Algorithm 6 A generic SAEA Framework

```
1: Initialization: Generate a database Γ, containing a population of designs with their exact
   fitness values.
2: while computational budget is not exhausted do
3:    Generate offspring population using evolutionary operators (selection, crossover, muta-
      tion).
4:    Build global/local surrogate model(s) using points from database Γ.
5:    Evaluate offspring population using surrogate model(s) and/or exact function evaluations.
6:    Archive all exact evaluations into database Γ.
7:    if Memetic Search then
8:       Perform local refinement employing surrogate model(s) and exact function evaluations.
9:       Archive all exact evaluations into database Γ.
10:      Replace offspring with refined individuals.
11:   end if
12: end while
```

$S = \{s_1, s_2, \ldots, s_n\}$. The transitions are represented by P, a $(n \times n)$-matrix with elements $P_{i,j} : i, j = \{1, \ldots, n\}$, where P_{ij} denotes the probability of the process moving from state i to j and n is the number of all possible states. For clarity, some important definitions are outlined as follows:

Definition 3.1 *The random process is said to be time-homogeneous if the transition probability P_{ij} is independent of time, i.e., $P_{ij}^{t_1} = P_{ij}^{t_2}$.*

Definition 3.2 *A square matrix $M = \{m_{ij}\}_{n\times n}$ is said to be positive ($M > 0$) if $m_{ij} > 0$ for all $i, j \in \{1, \ldots, n\}$.*

Definition 3.3 *A square matrix $M = \{m_{ij}\}_{n\times n}$ is said to be stochastic if $\sum_{j=1}^{n} m_{ij} = 1$ for all $i, j \in \{1, \ldots, n\}$.*

Definition 3.4 *The j^{th} column in a square matrix $M = \{m_{ij}\}_{n\times n}$ is said to be a positive column if $m_{ij} > 0$ for all $i \in \{1, \ldots, n\}$.*

Definition 3.5 *A Markov chain is irreducible if for all $s_i, s_j \in S$, $s_i \longleftrightarrow s_j$. In other words, for any $s_i, s_j \in S$, there is a t such that $P_{ij}^t > 0$. Otherwise, the chain is said to be reducible. Finite irreducible chains are always recurrent.*

Definition 3.6 *An irreducible chain is said to be aperiodic if for all pairs of states $i, j \in S$, there exists an integer $T < \infty$, such that for all $t > T$, the probability $P_{ij}^t > 0$.*

Without loss of generality, GA is used as an instance of EA throughout the analysis. First, the states of a Markov chain are defined as S, where each state in S represents the collection of individuals in an GA population at a certain time t. Take for instance, $S_1 = \{1001, 0100, 1100, \ldots, O_s\}$, where O_j denotes the j^{th} offspring of the population with size of s. Since the chromosomes are encoded with a finite resolution, there are a finite number of possible states in S.

The probabilistic changes in an population due to the evolutionary operators in a typical GA may then be modeled using stochastic matrices P_c, P_m, and P_s denoting the crossover, mutation, and selection operators, respectively. In Memetic Algorithm (MA), besides the GA operators, a local refinement procedure is also performed on each individual. Here, the local refinement procedure is modeled as the transition matrix P_l. In total, the GA or MA can then be modeled by a single transition matrix P as $P_{GA} = P_c P_m P_s$ or $P_{MA} = P_c P_m P_l P_s$, respectively. In surrogate-assisted GAs (SAGAs), the existence of approximated fitness generates additional bias to the selection process, represented by P_s. Similarly, for surrogate-assisted MAs (SAMAs) which employ local refinements based on the approximated fitness, this may result in a varying P_l.

Lemma 3.1 *The transition matrix P_s and P_l in SAGA and SAMA, respectively can be time-homogeneous after some period T.*

Proof: When different surrogate models are used, their varying accuracies might affect the search. However, with the availability of proven model management frameworks that converge to a local optimum, the effect of varying accuracies in the surrogate models used is minimized. For instance, the trust-region framework [15][194][32] may be employed to

ensure that local searches will eventually reach their respective local optimum. The trust-region framework assumes the availability of first order ($f'(\mathbf{x})$) and zero order ($f(\mathbf{x})$) accuracy in the surrogate models to ensure convergence to a local optimum, [32] and [194] have independently shown that the use of inexact gradient information is sufficient for such convergence property. As the search progresses, the cardinality of database archived, $\mid \Gamma \mid$ grows and thus the accuracies of the surrogate models. Hence, at time $\tau > T$, it happens that:

$$\lim_{t \to \tau} Prob(|f(\mathbf{x}) - \hat{f}(\mathbf{x})| \approx 0 \mid \Gamma_t) = 1 \qquad \text{(Eq. 3.1)}$$

A credible model management framework supported by a mild assumption on the model accuracy hence ensures that the global/local searches starting from a point $\mathbf{x}$ will always reach its respective local optimum $\mathbf{x}_{opt}$ in the surrogate landscape at time $\tau > T$, or mathematically:

$$\lim_{t \to \tau} Prob(\arg\min \hat{f}(\mathbf{x}^{(i)}) = \mathbf{x}_{opt}^{(i)} \mid \hat{f}(\mathbf{x}) \approx f(\mathbf{x})) = 1, \ \forall i = \{1, \ldots, s\} \qquad \text{(Eq. 3.2)}$$

where s is the population size. Consequently, this infers the time-homogeneity property of P_s and P_l in both SAGA and SAMA. $\square$

Lemma 3.2 *Let P_c, P_m, P_l, and P_s be stochastic matrices, where P_m is positive. Then $P_{SAGA} = P_c P_m P_s$ in SAGA and $P_{SAMA} = P_c P_m P_l P_s$ in SAMA have at least one positive column.*

Proof: Since P_m is positive, it is always true that $A = P_c P_m$ is also positive. To proceed, it can be safely assumed that $P_{s_{ii}} > 0$ for all $i \in \{1, 2, \ldots, n\}$ since there is always a possibility that selection does not alter the population state [169]. Then, it is ensured that $P_{SAGA} = AP_s$ is also a positive matrix.

In SAMA, suppose $S_l = \{S_1, S_2, \ldots, S_z\} \subset S$, are the states in which all members of the population are local optima, *i.e.*:

$$S_l = \{S_i \mid \mathbf{x}^{(j)} \in S_i \wedge f'(\mathbf{x}^{(j)}) = 0\},\ \forall i = \{1, \ldots, z\},\ \forall j = \{1, \ldots, s\} \qquad \text{(Eq. 3.3)}$$

where z is the number of states with only local optima, s is the population size, and $f'(\mathbf{x})$ is the first derivative of $f(\mathbf{x})$. It is obvious that the global optimum is among the z local optima. Since all states in S_l will only loop back to itself as the result of local search as they consist only local optima, P_l will have $P_{l_{ii}} = 1$ for all $i \in \{1, 2, \ldots, z\}$. Having such conditions in P_l ensures that $B = AP_l$ contains z positive columns, corresponding to the states in S_l. Since $P_{s_{ii}}$ is always positive for all $i \in \{1, 2, \ldots, n\}$, it is ensured that $P = BP_s$ will have z positive columns. □

Theorem 3.1 *SAGA has the global convergence property.*

Proof: From Lemma 3.2, $P_{SAGA} = P_c P_m P_s$ is strictly positive. Consequently, the entire state space is a closed (ergodic) set because the Markov chain is irreducible and aperiodic. Hence, the Markov chain must be composed of only positive recurrent states. Suppose $S_{opt} \subset S$, S_{opt} being the set of states containing the global optima, $\mathbf{x}_{opt}$. Because P_{SAGA} is irreducible, aperiodic, and positive recurrent, as $t \to \infty$, the probability that all possible population states, including S_{opt}, will be visited at least once, approaches 1. Hence, regardless of the initial distribution, it is apparent that $\lim_{t\to\infty} Prob\{\mathbf{x}_{opt} \in S_t\} = 1$. Further, since the fittest solution is always tracked in practice, it extends from [169] that the search converges globally using an elitist selection mechanism. □

Theorem 3.2 *SAMA has the global convergence property.*

Proof: On the other hand, $P_{SAMA} = P_c P_m P_l P_s$ is positive in z columns, corresponding to the states in S_l (Lemma 3.2). Since the global optimum is a subset of all local optima, it

is always true that the global optimum can be found in some states S_{opt}, where $S_{opt} \subset S_l$. Having a positive column in the transition matrix P_{SAMA} implies that all states in S_{opt} will be reachable from any other states as $t \rightarrow \infty$. Consequently, this implies that $\lim_{t\rightarrow\infty} Prob\{\mathbf{x}_{opt} \in S_t\} = 1$. Based on the same rationale on the best solution tracking (Theorem 3.1), the search converges globally using an elitist selection mechanism. □

3.2 Parallel and Distributed Computing

As a population-based method, one well-known strength of EA is the ability to partition the population of individuals among multiple computing nodes. Doing so allows sublinear speed-up in computation and even super-linear speed-up [12]. In this section, a brief overview on two important topics, *i.e.*, parallel EA and Grid computing, is presented to provide a necessary understanding on our works in latter chapters.

3.2.1 Parallel Evolutionary Algorithms

In the literature, there exist different classifications on Parallel Evolutionary Algorithms (PEAs). Though most of them consider parallel GA only, the classification can be generalized for other existing EAs. In [10], the different parallelization techniques are divided into two major groups, *i.e.*, the *standard* and *decomposition* approaches. The standard approach is rather straightforward since its goal is solely to gain computational speed-up. Hence, it only parallelizes the concurrent computations of the sequential algorithm resulting in no changes in the algorithmic behavior. On the other hand, the decomposition approach divides the population and/or problem search space into smaller partitions hence changes in algorithmic behavior are expected. Similarly, [12] also considers such classification, using the terms *panmictic* and *structured* EAs. In [134], the different parallel GAs are classified based on their implementation details into 8 classes, *i.e.*: *1) Synchronous and asynchronous master-slave*, *2) Static subpopulations with migration*, *3)*

Static overlapping subpopulations, *4) Dynamic overlapping subpopulations*, *5) Massively parallel GAs*, *6) Parallel steady-state GAs*, *7) Parallel messy GAs*, and *8) Hybrid of other methods*.

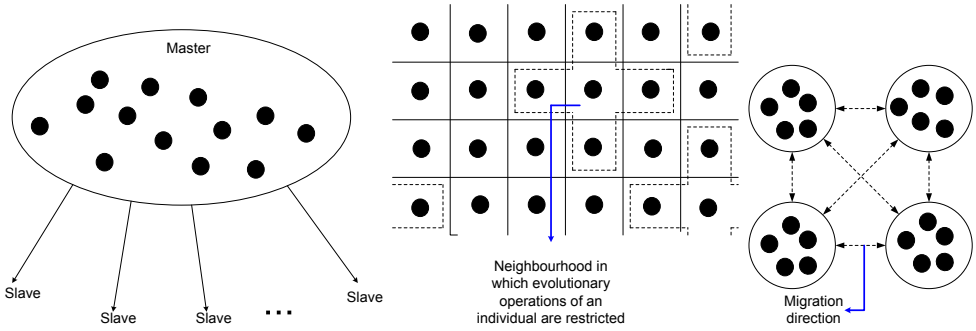
On the other hand, [31] categorizes the different parallelization techniques of PEA into four different models as depicted in Fig. 3.6-3.7, which are described as follows:

- **Master-slave model** [49][26][78][8][74]. In this model, it is assumed that there is only a single panmictic population. However, unlike the canonical EA, evaluations of individuals are distributed by scheduling fractions of the population among the processing computing nodes. Such a model has the advantage for ease of implementation and does not alter the algorithmic behavior of a canonical EA.

- **Fine-grained or Cellular model** [66][125][118][67][177] [14]. Fine-grained PEA consists of only a single population, which is spatially structured. For instance, using a Von Neumann neighbourhood structure, selection and mating are restricted to the four direct neighbours. Nevertheless, groups overlap such that good individuals may disseminate across the entire population.

- **Multi-population/deme or island model** [69][151] [193][34][22][65]. This model consists of several subpopulations that exchange individuals occasionally. This exchange of individuals is called migration and is controlled by several parameters, *i.e.*, the migration interval, frequency, and individual selection criteria for migration and replacement.

- **Hierarchical/hybrid model** [70][113][13][14] It is also possible to combine two of the aforementioned approaches to form a hierarchical or hybrid model of PEA. The most common hierarchical model involves an island model at the top and master-slave, cellular, or another island model in each subpopulation.

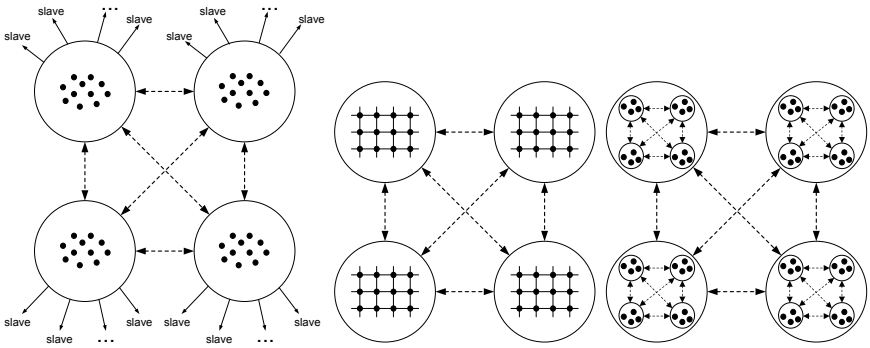


3.6.a: Master-slave model

3.6.b: Cellular model

3.6.c: Island model

Figure 3.6: Three basic models of Parallel Evolutionary Algorithms.



3.7.a: Island + Master-slave model

3.7.b: Island + Cellular model

3.7.c: Island + Island model

Figure 3.7: Hierarchical models of Parallel Evolutionary Algorithms.

3.2.2 Grid Computing

Advancements in technology have brought about the availability of more powerful processing power of today's computer. Moore's law [124] stated that the number of transistors that can be inexpensively placed on an integrated circuit is increasing exponentially, doubling approximately every two years. Even so, the problems encountered in real-life, especially in science and engineering, also have the tendency to become more complex and hence computationally intensive. The recent development in multi-core hardware platform has been showing great promises to cope with this issue. However, the availability of many such powerful computers might be still cost-prohibitive for a single organization. A more viable solution is one of the latest trend in distributed computing community, *i.e.*, the Grid computing technology [54][21][56][55].

The term Grid computing refers to a group of independent, geographically disparated, and possibly heterogeneous computer clusters, each consists single or multiple computing nodes, offering computational power or storage for authorized users. The Grid technology has made it possible for different project groups to harness computing resources that span across laboratories or even organizations efficiently, in contrast to the old days of having inadequate number of dedicated computer resources capable only for solving small-scale problems. In addition, the sharing of computational resources may also relax the constraint on limited commercial licenses a project group may access seamlessly for analyzing the designs at once. This is primarily due to the high costs associated with the site licenses of commercial analysis packages, for example, Ansys [1], Fluent [2], and Sysnoise [6].

A typical Grid computing environment with different functional units is depicted in Fig. 3.8. Generally, most of such functionalities has been incorporated in Grid middlewares, such as Globus Toolkit as the *de facto* middleware in this field. The processes involved are described as follows:

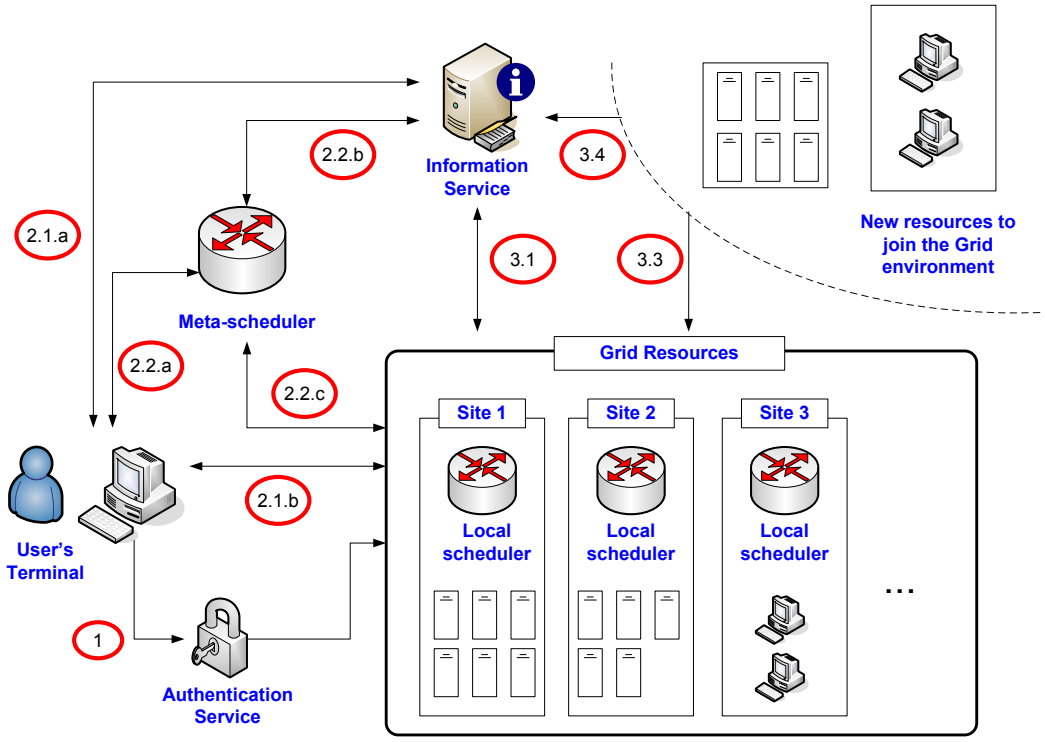


Figure 3.8: A typical Grid computing environment.

(i) To access the available Grid resources, a user is required to authenticate him/herself to the Grid. A common procedure is that only a single sign-on is required for an authorized user to access all resources living in the Grid. For instance, in the X.509 authentication protocol, before the first access to the Grid, a user must acquire an X.509 certificate and create a public/private keypair. While the private key is safeguarded, the public key is then sent to the Certificate Authority (CA) as a certificate request. Upon approval, a signed certificate will be issued for the user, which contains the user's public key and X.509 identity encrypted with the CA's private key. Note that every resource in the Grid should have a copy of the CA's certificate containing its public key, hence can authenticate the user by decrypting the user's certificate.

(ii) Only after authenticated, the user is allowed to use the computing resources. There

could be different scenarios for job submission depending on the Grid configuration and user's needs, as follows:

2.1 User can request the information service for a list of available computing resources, to which jobs can be submitted. In Figure 3.8, this step is shown by 2.1.a and 2.1.b. This scenario might be an option for some reasons: *1)* the environment do not specifically have a meta-scheduler or brokerage service, or *2)* user purposely wants to bypass the meta-scheduler to reduce overhead or to use own scheduling algorithm.

2.2 An alternative path is to make use of the meta-scheduling service. Depending on the meta-scheduler's implementation, there could be 2 possible ways this scenario can be carried out: *1)* the meta-scheduler acts as a proxy to the user to submit jobs, as shown in processes 2.2.a, 2.2.b, and 2.2.c, or *2)* the meta-scheduler only provides the job schedule, and jobs are still submitted directly by user. These are illustrated by processes 2.2.a, 2.2.b, and 2.1.b.

(iii) Besides the above, it is also important for the information service to maintain the most up-to-date information. Hence, any changes on the existing resources such as addition/removal of compute nodes, and new resources to join the Grid should be updated (refer to 3.1, 3.2, and 3.3). In practice, this functionality might be realized by Globus Monitoring and Discovery Service (MDS) [5], Ganglia [120], *etc.*

Chapter 4

Evolutionary Computation Using Data-fitting Approximation Models

In spite of the extensive research efforts on the topic of surrogate-assisted EAs, existing surrogate-assisted evolutionary frameworks remain open for further improvement. Jin *et al.* in [90] have shown that existing surrogate-assisted evolutionary frameworks proposed are often flawed by the introduction of false optima since the parametric approximation technique used may not be capable of modeling the problem landscapes accurately, thus producing unreliable search. Generally, the *'curse of dimensionality'* creates significant difficulties in the construction of accurate surrogate models for fitness prediction. On the other hand, it is worth keeping in mind that approximation error in the surrogate model does not always harm. A surrogate model capable of smoothing the multi-modal or noisy landscape of the complex problem may contribute more beneficially to the evolutionary search than one that models the original fitness function accurately. For instance, the study in [145] has emphasized the importance of predicting search improvement as opposed to the usual practice of improving only the quality of the surrogate in the context of evolutionary optimization.

Based on these recent works, it is also worth highlighting the influence of the approximation method used on the performance of any surrogate-assisted evolutionary search. The greatest barrier to further progress is that, with so many approximation techniques

available in the literature, it is almost impossible to know which is most relevant for modeling the problem landscape or generating reliable fitness predictions when one has only limited knowledge of its fitness space before the search starts. Moreover, approximation techniques by themselves may model differently on different problem landscapes. Depending on the complexity of a design problem, a single approximation model that may have proven to be successful on an instance might not work so well, or at all, on others. In the field of multidisciplinary optimization, such observations have also been reported [173][205][119][155][175][174][62][9][98]. In those works, this issue is commonly handled by performing multiple optimization runs, each on different surrogate model or ensemble model. In [173][205][174], a set of surrogate models consisting Kriging, PR, RBF, and weighted average ensemble is used to demonstrate that multiple surrogates can improve robustness of optimization at minimal cost. Similarly, [119] uses PR and RBF surrogate models in the context of multi-objective optimization and shows that each of the models perform better at different region of the Pareto front. Others in [155][175][62][9] resolve this issue by introducing various ensemble model building techniques. It is shown from these works that ensemble models generally outperform most of the individual surrogates.

Given the restricted theoretical knowledge available, the present chapter introduces a generalized framework for unifying diverse surrogate models synergistically in the evolutionary search. In contrast to the existing efforts, the focus here is on predicting search improvement in the context of optimization as opposed to solely on improving the prediction quality of the approximation. In particular, the problem to attain a reliable search improvement in surrogate-assisted evolutionary framework is generalized into two major goals: *1)* to mitigate the *'curse of uncertainty'* and, *2)* to benefit from the *'bless of uncertainty'*. The *'curse of uncertainty'*[1] refers to the negative consequences introduced by the approximation error of the surrogate models used. On the other hand, *'bless of*

[1]In the present context, the definition of 'uncertainty' refers to the approximation errors in the fitness function due to the use of surrogate models based on the definitions given in [87].

uncertainty' refers to the benefits attained by the use of surrogate models. Particularly, surrogate models that are capable of generating reliable fitness predictions on diverse problems of different landscapes to mitigate the *'curse of uncertainty'* on one hand, and on the other hand surrogate models that are capable of smoothing rugged fitness landscapes to prevent the search from getting stuck in local optima [145], are sought. Previous works by Liang *et al.* [110][111] have also confirmed that smoothed landscape of rugged fitness landscape can lead the search to optimum solutions easier than using the exact fitness landscape.

4.1 Impacts of Approximation Errors

In this section, the effects of uncertainty introduced by inaccurate approximation models on search performance of SAEAs are briefly discussed. Without loss of generality, here computationally expensive minimization problems under limited computational budget with bound constraints as illustrated in Equation Eq. 2.3 are considered.

Note that when approximation is applied on one or more objectives, there are two commonly adopted strategies, *i.e.*, *1)* one approximation model per objective function, and *2)* one approximation model for an aggregated(linear or nonlinear combination) objective function, $f_{aggr}(\mathbf{x})$, which is defined as:

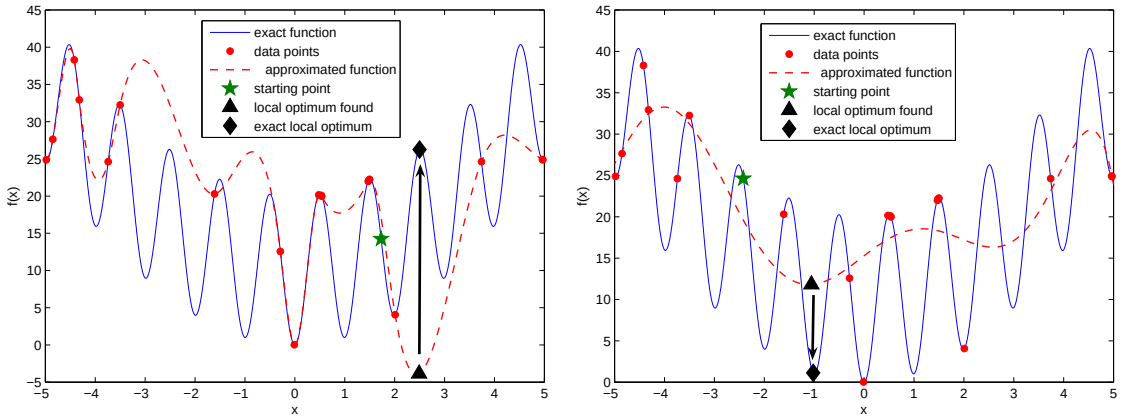
$$f_{aggr}(\mathbf{x}) = \sum_{i=1}^{r} w_i f_i(\mathbf{x}) \qquad \text{(Eq. 4.1)}$$

where r is the number of objectives, w_i is the weight for the i^{th} objective. In this chapter, the second strategy is considered. Since in single-objective context, $f_{aggr}(\mathbf{x}) = f(\mathbf{x}) = f_1(\mathbf{x})$, the term $f(\mathbf{x})$ might be used interchangeably to $f_{aggr}(\mathbf{x})$ for brevity purpose when only single-objective context is considered.

If $f_{aggr}(\mathbf{x})$ denotes the original fitness function and the approximated function is $\hat{f}_{aggr}(\mathbf{x})$, the approximation errors at any solution vector $\mathbf{x}$ is $e(\mathbf{x})$, *i.e.*, the uncertainty

introduced by the surrogate at $\mathbf{x}$, may then be defined as:

$$e(\mathbf{x}) = |f_{aggr}(\mathbf{x}) - \hat{f}_{aggr}(\mathbf{x})| \qquad \text{(Eq. 4.2)}$$



4.1.a: 'Curse of uncertainty' in single-objective SAEA. Approximated function in the figure is obtained using spline interpolation technique.

4.1.b: 'Bless of uncertainty' in single-objective SAEA. Approximated function in the figure is obtained using a low order Polynomial Regression.

Figure 4.1: Curse and bless of uncertainty in single-objective SAEA.

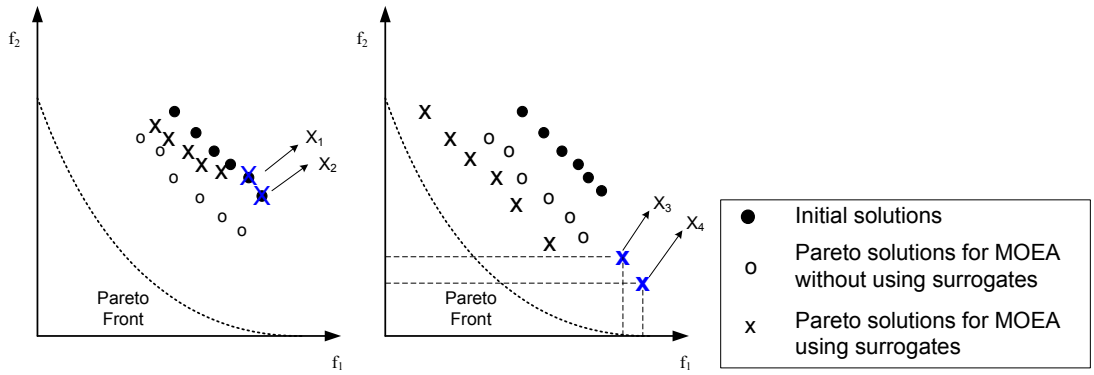
Here, the negative and positive impacts introduced by the approximation inaccuracies of the surrogates on SAEA search [145] are highlighted. The negative impact or otherwise known as the '*curse of uncertainty*' on SAEA search can be briefly defined as the phenomenon where the inaccuracy of the surrogates used results in the SAEA search to stall or converge to false optimum. To illustrate the '*curse*' effect, we refer to Fig. 4.1.a where the SAEA is likely to converge to the false optimum of the spline interpolation model due to inaccuracy. On the other hand, the positive impact, *i.e.*, the '*bless of uncertainty*' in SAEA materializes when the use of surrogate brings about greater search improvements over the use of original exact objective/fitness function. For instance, the surrogate can help to traverse the search across valleys and hills of local optima by smoothing the ruggedness/multi-modality of the problem landscape. To illustrate the blessing effect,

we refer to the example in Fig. 4.1.b, where a low order polynomial regression scheme is used to approximate the exact objective function. Due to the smoothing effect of the polynomial surrogate, the search leads to an improved solution that is unlikely to be attained even if the exact objective function is used. Hence, the '*bless of uncertainty*' brings about possible acceleration in the search. Besides a faster convergence, recent study in [143] revealed that the '*bless of uncertainty*' in SAEA also exists in the form of improving evolutionary search diversity through the use of surrogate models.

Next, to illustrate '*curse and bless of uncertainty*' in the context of multi-objective optimization, we refer to the examples in Figs. 4.2.a and 4.2.b. Fig. 4.2.a depicts the effect of '*curse of uncertainty*' in MOEA search due to the presence of inaccurate surrogate models. In Fig. 4.2.a, the surrogate-assisted MOEA search is observed to be evolving towards poor non-dominated solutions in comparison to that based on the exact fitness functions. Moreover, those labeled as $\mathbf{x}_1$ and $\mathbf{x}_2$ in Fig. 4.2.a suggest that some solutions might stall, while others fail to converge optimally. On the other hand, Fig. 4.2.b illustrates the presence of '*bless of uncertainty*' where the errors in the surrogate used is observed to improve the MO evolutionary search in both convergence and diversity measures. Particularly, some improved solutions of the surrogate-assisted search is shown to converge faster than a search without using surrogates, while others such as $\mathbf{x}_3$ and $\mathbf{x}_4$ are newly found non-dominated solutions which improve the search diversity.

4.2 Generalizing Surrogate-Assisted Evolutionary Algorithms

In this section, a generalization of surrogate-assisted evolutionary frameworks for optimization of problems with objectives that are computationally expensive to evaluate, is presented. The generalized framework illustrated here for unifying diverse approximation



4.2.a: 'Curse of uncertainty' in multi objective EA using surrogates.

4.2.b: 'Bless of uncertainty' in multi objective EA using surrogates.

Figure 4.2: Curse and Bless of Uncertainty in Multi-Objective EA using Surrogates.

concept synergistically is a surrogate-assisted memetic algorithm that conducts simultaneous local searches on separate *ensemble* and *smoothing* surrogate models. Note that the rationale behind using a memetic framework over a traditional evolutionary framework is multi-fold [137][140]. First, we aim to exploit MAs' capability of locating the local and global optima efficiently. Second, a memetic model of adaptation exhibits the plasticity of individuals that a pure genetic model fails to capture. Further, by limiting the use of surrogate models within the local search procedures, the global convergence property of EAs can be ensured.

In the generalized framework, we introduce first the idea of employing online local ensemble surrogate models constructed from diverse approximation concepts using data points that lie in the vicinity of an initial guess. The surrogate models are used to replace the expensive function evaluations performed in the local search phase. The improved solution generated by the local search procedure then replaces the genotype and/or fitness of the original individual.

4.2.1 Ensemble Model

To mitigate the *'curse of uncertainty'* due to the effect of using imperfect fitness approximation, surrogate models that are capable of generating reliable fitness predictions on diverse problems, are sought. In particular, since it is almost impossible to know in advance which approximation technique best suits the optimization problem in hand, a synergy of diverse approximation methods is considered through the use of ensemble models to generate reliable accurate predictions across problems of differing functional landscapes [93][114][155], as opposed to single surrogate models created by specific approximation scheme that may not be appropriate for the problem in hand. In what follows, online local weighted average ensembles are considered. For instance, in the single-objective context, the predicted ensemble output of $f(\mathbf{x})$ using n surrogate models is formulated as:

$$\begin{aligned} \hat{f}_{ens}(\mathbf{x}) &= \sum_{i=1}^{n} c_i \hat{f}_i(\mathbf{x}), \\ \sum_{i=1}^{n} c_i &= 1, \end{aligned} \qquad \text{(Eq. 4.3)}$$

where $\hat{f}_{ens}(\mathbf{x})$ and $\hat{f}_i(\mathbf{x})$ are the fitness prediction made by the ensemble and i^{th} surrogate model, respectively. The same formulation applies in the multi-objective context where $f_{aggr}(\mathbf{x})$ is considered. c_i is the weight coefficient associated with the i^{th} surrogate model. A model can be assigned a larger weight if it is found or deemed to be more accurate. Hence, the weighting function becomes:

$$c_i = \frac{\sum_{j=1, j\neq i}^{n} \varepsilon_j}{(n-1)\sum_{j=1}^{n} \varepsilon_j}, \qquad \text{(Eq. 4.4)}$$

where ε is the error measurement for the j^{th} surrogate model. Here, the root mean square error ($rmse$) is used as the error measurement. The $rmse$ of each surrogate model is then of the form:

$$rmse = \sqrt{\frac{\sum_{i=1}^{m} e_i^2(\mathbf{x})}{m}}, \qquad \text{(Eq. 4.5)}$$

where m is the number of data samples compared, $e(\mathbf{x})$ is the error of prediction for data point $\mathbf{x}$, as shown in Equation (Eq. 4.2). Some good expositions on other ensemble model building techniques can also be found in [155][175][62][9][114]. While the use of ensembles generally results in more accurate models, one obvious shortcoming is on the extra CPU cost on building multiple models to be aggregated. However, in the context of computationally expensive problems which might cost many minutes to hours of supercomputer time, this issue is almost negligible.

4.2.2 Landscape Smoothing Model

Meanwhile, to benefit from the *'bless of uncertainty'*, smoothing techniques including global convex underestimation, tunneling and filling methods are some appropriate alternatives that may be used [153]. Given a functional landscape, smoothing methods transform the function into one with noticeably fewer minima, thus speeding up the evolutionary search. In the generalized framework, global convex underestimation is used for successive smoothing of the problem landscape within the local search phase which is realized through low-order polynomial regression (PR). Besides the generalization property of PR models on rugged landscape, the low computational costs incurred makes them very efficient as online surrogate models. Note that the PR model may be used in both ensemble and the smoothing models, hence only a one time model building cost is involved.

4.2.3 Single-Objective Optimization

In this subsection, the generalized surrogate memetic framework for single-objective optimization is described. A brief outline of the generalized surrogate single-objective memetic algorithm (GS-SOMA) is presented in Algorithm 7. Note that the difference between the GS-SOMA and a traditional MA is highlighted in Algorithm 4 and lies in the local search phase.

Algorithm 7 Generalized Surrogate Single-Objective Memetic Algorithm (GS-SOMA)

initialization: Generate and evaluate a database containing a population of designs, archive all exact evaluations into the database.
while computational budget is not exhausted **do**
  if generation count $<$ database building phase (G_{db}) **then**
    Evolve the population using exact fitness function evaluations, archive all exact evaluations into the database.
  else
    Apply evolutionary operators to create a new population.

    /$*$ $*$ $*$ $*$ **Local Search Phase** $*$ $*$ $*$ $*$/

    for each individual $\mathbf{x}$ in the population **do**
      Find m nearest points to $\mathbf{x}$ in database as training points for surrogate models.
      Build model-1: M_1, as an ensemble of all M'_j for $j = 1, \ldots, n$ where n is the number of surrogate models used.
      Build model-2: M_2, which is a low-order PR model.
      Apply local search in M_1 to arrive at $\mathbf{x}^{(1)}_{opt}$, and M_2 to arrive at $\mathbf{x}^{(2)}_{opt}$.
      Replace $\mathbf{x}$ with the locally improved solution, *i.e.*,
      if $f(\mathbf{x}^{(1)}_{opt}) < f(\mathbf{x}^{(2)}_{opt})$ **then**
        $\mathbf{x} = \mathbf{x}^{(1)}_{opt}$
      else
        $\mathbf{x} = \mathbf{x}^{(2)}_{opt}$
      end if
      Archive all new exact function evaluations into the database.
    end for

    /$*$ $*$ **End of Local Search Phase** $*$ $*$/

  end if
end while

GS-SOMA begins with the initialization of a population of design points. During the database building phase, the search operates like a traditional evolutionary algorithm based on the original exact fitness function for some initial G_{db} generations. Up to this stage, no form of surrogates are used, and all exact fitness function evaluations made are archived in a central database. If a previously acquired database is already available, the database building phase can be ignored by setting $G_{db} = 0$. Subsequently, the algorithm proceeds into the local search phase. For each individual $\mathbf{x}$, n online surrogates that model the fitness function are created dynamically using m training data points, which lie in the vicinity of $\mathbf{x}$, extracted from the archived database of previously evaluated design points. From the n surrogates, an ensemble model is built. From here, two separate local searches are conducted on *1)* M_1, the ensemble of n surrogate models, and *2)* M_2, a low-order PR model. If improved solutions are achieved, GS-SOMA proceeds with the individual replacement scheme. Since the Lamarckian learning is adopted here, the genotype/phenotype of the initial individual is then replaced by the higher quality solutions among the two that are locally improved based on M_1 and M_2, *i.e.*, $\mathbf{x}_{opt}^{(1)}$ or $\mathbf{x}_{opt}^{(2)}$. Note that since local searches are meant to obtain improved solutions, it is always the case that $f(\mathbf{x}_{opt}^{(1)}) \leq f(\mathbf{x})$ and $f(\mathbf{x}_{opt}^{(2)}) \leq f(\mathbf{x})$. Having such condition, comparison only needs to be performed between $f(\mathbf{x}_{opt}^{(1)})$ and $f(\mathbf{x}_{opt}^{(2)})$. It is worth noting that any exact function evaluations encountered during the search are archived into the database. The search cycle is then repeated until the allowed maximum computational budget is exhausted.

4.2.4 Multi-Objective Optimization

Next, we describe the Generalized Surrogate Memetic framework in the context of MOO. The brief outline of a typical Multi-Objective Memetic Algorithm (MOMA) using weighting (scalarization) technique [83][84][104] is illustrated in Algorithm 8. In contrast, the

studied GSM framework for multi-objective optimization (GS-MOMA) is outlined in Algorithm 9. Note that the key differences of the two algorithms lie in the local search phase and selection pool forming highlighted.

Algorithm 8 Multi-Objective Memetic Algorithm

1: **initialization**: Generate and evaluate a population of design vectors.
2: **while** computational budget is not exhausted **do**
3: Apply MO evolutionary operators to create a new population.
4:
5: /∗∗∗∗ **Local Search Phase** ∗∗∗∗/
6:
7: **for** each individual $\mathbf{x}$ in the population **do**
8: Generate a random weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_r)$, $\sum_{i=1}^{r} w_i = 1$ where r is the number of objectives.
9: Apply local search in $f_{aggr} = \sum_{i=1}^{r} w_i f_i(\mathbf{x})$ to find an improved solution, $\mathbf{x}_{opt}$.
10: Perform Lamarckian learning, *i.e.*,
11: **if** $f_{aggr}(\mathbf{x}_{opt}) < f_{aggr}(\mathbf{x})$ **then**
12: $\mathbf{x} = \mathbf{x}_{opt}$
13: **end if**
14: **end for**
15:
16: /∗∗ **End of Local Search Phase** ∗∗/
17:
18: **end while**

GS-MOMA begins with the population initialization phase and evolutionary search based on the exact fitness function for a number of early generations, G_{db}, before entering the local search phase. Similar to GS-SOMA, if a previously acquired database is already available, $G_{db} = 0$ can be applied to bypass the database building phase. In the local search phase, independent local searches are conducted on *1)* M_1, the ensemble of n surrogate models, and *2)* M_2, the smoothing low-order PR model on each individual of the generated offspring population. It is worth noting that any exact function evaluations encountered during the search are archived into the database. For the sake of brevity, the core distinguishing feature of GS-MOMA is highlighted in line 17 of Algorithm 9, *i.e.*, the existence of the *Replace&Archive* procedure.

Algorithm 9 Generalized Surrogate Multi-objective Memetic Algorithm (GS-MOMA)

initialization: Generate and evaluate an initial population with N_{pop} individuals, archive all exact evaluations into a database.
while computational budget is not exhausted **do**
  if generation count $<$ database building phase (G_{db}) **then**
    Evolve the population using exact fitness function evaluations, archive all exact evaluations into the database.
  else
    Generate the offspring population, P_o using evolutionary operators from current population, P_c.

    /$*$ $*$ $*$ $*$ **Local Search Phase** $*$ $*$ $*$ $*$/

    Initialize the learning archive, A_l to empty state.
    for each individual $\mathbf{x}$ in the offspring population **do**
      Generate a random weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_r)$, $\sum_{i=1}^{r} w_i = 1$ where r is the number of objectives.
      Find m nearest points to $\mathbf{x}$ in database as training points for surrogate models.
      Build model-1: M_1, as an ensemble of all M'_j for $j = 1, \ldots, n$ where n is the number of surrogate models used, of $f_{aggr} = \sum_{i=1}^{r} w_i f_i(\mathbf{x})$
      Build model-2: M_2, which is a low-order PR model, of $f_{aggr} = \sum_{i=1}^{r} w_i f_i(\mathbf{x})$
      Apply local search in M_1 to arrive at $\mathbf{x}_{opt}^{(1)}$, and M_2 to arrive at $\mathbf{x}_{opt}^{(2)}$
      Replace&Archive($\mathbf{x}$, $\mathbf{x}_{opt}^{(1)}$, $\mathbf{x}_{opt}^{(2)}$, A_l)
    end for

    /$*$ $*$ **End of Local Search Phase** $*$ $*$/

    /$*$ $*$ $*$ $*$ **Selection pool forming** $*$ $*$ $*$ $*$/

    Form selection pool, $P_s = P_c \bigcup P_o \bigcup A_l$.

    /$*$ $*$ **End of selection pool forming** $*$ $*$/

  end if
end while

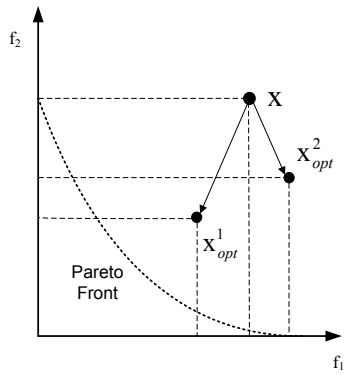
Algorithm 10 Procedure *Replace&Archive*($\mathbf{x}$, $\mathbf{x}_{opt}^{(1)}$, $\mathbf{x}_{opt}^{(2)}$, A_l)

if $\mathbf{x}_{opt}^{(1)} \preceq \mathbf{x}$ **then**
  $\mathbf{x} = \mathbf{x}_{opt}^{(1)}$
  if $\mathbf{x}_{opt}^{(2)} \preceq \mathbf{x}_{opt}^{(1)}$ **then**
    $\mathbf{x} = \mathbf{x}_{opt}^{(2)}$
  else if $\mathbf{x}_{opt}^{(2)} \sim \mathbf{x}_{opt}^{(1)}$ **then**
    Archive $\mathbf{x}_{opt}^{(2)}$ in A_l
  end if
else if $\mathbf{x}_{opt}^{(2)} \preceq \mathbf{x}$ **then**
  $\mathbf{x} = \mathbf{x}_{opt}^{(2)}$
  if $\mathbf{x}_{opt}^{(2)} \sim \mathbf{x}_{opt}^{(1)}$ **then**
    Archive $\mathbf{x}_{opt}^{(1)}$ in A_l
  end if
else if $(\mathbf{x}_{opt}^{(1)} \sim \mathbf{x}) \wedge (\mathbf{x}_{opt}^{(2)} == \mathbf{x})$ **then**
  Archive $\mathbf{x}_{opt}^{(1)}$ in A_l
else if $(\mathbf{x}_{opt}^{(2)} \sim \mathbf{x}) \wedge (\mathbf{x}_{opt}^{(1)} == \mathbf{x})$ **then**
  Archive $\mathbf{x}_{opt}^{(2)}$ in A_l
else if $(\mathbf{x}_{opt}^{(1)} \sim \mathbf{x}) \wedge (\mathbf{x}_{opt}^{(2)} \sim \mathbf{x})$ **then**
  if $(\mathbf{x}_{opt}^{(1)} \preceq \mathbf{x}_{opt}^{(2)}) \parallel (\mathbf{x}_{opt}^{(1)} == \mathbf{x}_{opt}^{(2)})$ **then**
    Archive $\mathbf{x}_{opt}^{(1)}$ in A_l
  else if $\mathbf{x}_{opt}^{(2)} \preceq \mathbf{x}_{opt}^{(1)}$ **then**
    Archive $\mathbf{x}_{opt}^{(2)}$ in A_l
  else
    Archive $\mathbf{x}_{opt}^{(1)}$ and $\mathbf{x}_{opt}^{(2)}$ in A_l
  end if
end if

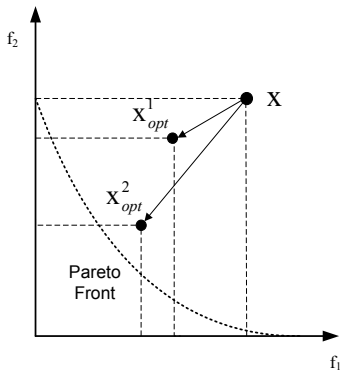
The *Replace&Archive* procedure performs replacements based on domination between the original offspring and the two local optima found. The original offspring will only be replaced by one dominating optimum found. All other local optima are then archived into the learning archive, A_l. Similar to the one in its single-objective counterpart, the only acceptable result in GS-MOMA's local searches is either $\mathbf{x}_{opt} \preceq \mathbf{x}$ or $\mathbf{x}_{opt} \sim \mathbf{x}$. Otherwise, there is no improvement to the original offspring, and hence we get $\mathbf{x}_{opt} == \mathbf{x}$. The different local search outcomes and corresponding actions taken by the scheme are summarized in Table 4.1, and can be obtained from the procedure provided in Algorithm 10. From there, it is noted that there are six different possible actions taken by GS-MOMA, *i.e.*,:

- One replacement is performed (*e.g.*, Fig. 4.3a).
- Two subsequent replacements are performed (*e.g.*, Fig. 4.3b).
- One replacement and one archiving are performed (*e.g.*, Fig. 4.3c).
- One archiving is performed (*e.g.*, Fig. 4.3d).
- Two archivings are performed (*e.g.*, Fig. 4.3e).
- Neither replacement nor archiving is performed (*e.g.*, Fig. 4.3f).

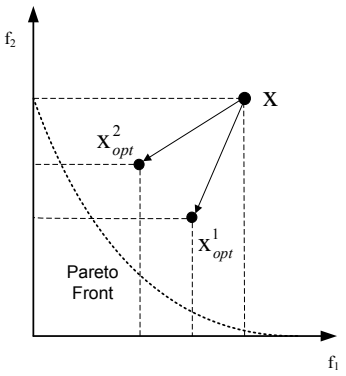
At the end of each GS-MOMA generation, A_l is combined with the current parent population, P_c, and the offspring population, P_o to form the entire pool of individuals, P_s that will then undergo the MOEA selection mechanism, *i.e.*, $P_s = P_c \bigcup P_o \bigcup A_l$. From here, the process described repeats until the maximum computational budget of the GS-MOMA is exhausted.



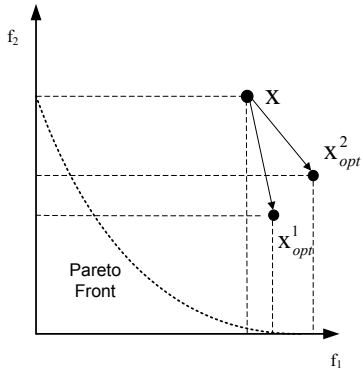
4.3.a: An example of the case where only one replacement is performed by GS-MOMA. $(\mathbf{x}_{opt}^{(1)} \preceq \mathbf{x}) \wedge (\mathbf{x}_{opt}^{(1)} \preceq \mathbf{x}_{opt}^{(2)}) \wedge (\mathbf{x} \sim \mathbf{x}_{opt}^{(2)})$. $\mathbf{x}_{opt}^{(1)}$ replaces $\mathbf{x}$.



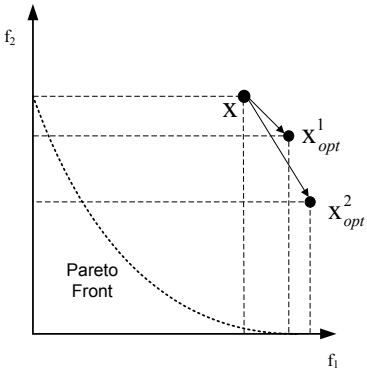
4.3.b: An example of the case where two subsequent replacements are performed by GS-MOMA. $(\mathbf{x}_{opt}^{(1)} \preceq \mathbf{x}) \wedge (\mathbf{x}_{opt}^{(2)} \preceq \mathbf{x}_{opt}^{(1)})$. $\mathbf{x}_{opt}^{(1)}$ replaces $\mathbf{x}$, followed by $\mathbf{x}_{opt}^{(2)}$ replaces $\mathbf{x}$.



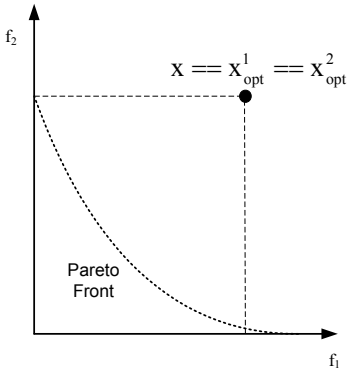
4.3.c: An example of the case where one replacement and one archiving are performed by GS-MOMA. $(\mathbf{x}_{opt}^{(1)} \preceq \mathbf{x}) \wedge (\mathbf{x}_{opt}^{(2)} \preceq \mathbf{x}) \wedge (\mathbf{x}_{opt}^{(1)} \sim \mathbf{x}_{opt}^{(2)})$. $\mathbf{x}_{opt}^{(1)}$ replaces $\mathbf{x}$, $\mathbf{x}_{opt}^{(2)}$ is archived in A_l.



4.3.d: An example of the case where only one archiving is performed by GS-MOMA. $(\mathbf{x} \sim \mathbf{x}_{opt}^{(1)}) \wedge (\mathbf{x} \sim \mathbf{x}_{opt}^{(2)}) \wedge (\mathbf{x}_{opt}^{(1)} \preceq \mathbf{x}_{opt}^{(2)})$. $\mathbf{x}_{opt}^{(1)}$ is archived in A_l.



4.3.e: An example of the case where two archivings are performed by GS-MOMA. $(\mathbf{x} \sim \mathbf{x}_{opt}^{(1)}) \wedge (\mathbf{x} \sim \mathbf{x}_{opt}^{(2)}) \wedge (\mathbf{x}_{opt}^{(1)} \sim \mathbf{x}_{opt}^{(2)})$. Both $\mathbf{x}_{opt}^{(1)}$ and $\mathbf{x}_{opt}^{(2)}$ are archived in A_l.



4.3.f: An example of the case where neither replacement nor archiving is performed. No new optimum found.

Figure 4.3: Examples of the six different actions taken by the *Replace&Archive* scheme in GS-MOMA for corresponding results of local searches.

Table 4.1: Actions taken by the *Replace&Archive* scheme in GS-MOMA for corresponding results of local searches. Note that irrelevant cases have been excluded for brevity.

$\mathbf{x}_{opt}^{(1)}$ vs $\mathbf{x}$	$\mathbf{x}_{opt}^{(2)}$ vs $\mathbf{x}$	$\mathbf{x}_{opt}^{(1)}$ vs $\mathbf{x}_{opt}^{(2)}$	Actions taken by GS-MOMA
$\preceq$	$\preceq$	$\preceq$	$\mathbf{x} = \mathbf{x}_{opt}^{(1)}$
$\preceq$	$\preceq$	$\succ$	$\mathbf{x} = \mathbf{x}_{opt}^{(2)}$
$\preceq$	$\preceq$	$\sim$	$\mathbf{x} = \mathbf{x}_{opt}^{(1)}$, archive $\mathbf{x}_{opt}^{(2)}$
$\preceq$	$\preceq$	$==$	$\mathbf{x} = \mathbf{x}_{opt}^{(1)}$
$\preceq$	$==$	$\preceq$	$\mathbf{x} = \mathbf{x}_{opt}^{(1)}$
$\preceq$	$\sim$	$\preceq$	$\mathbf{x} = \mathbf{x}_{opt}^{(1)}$
$\preceq$	$\sim$	$\sim$	$\mathbf{x} = \mathbf{x}_{opt}^{(1)}$, archive $\mathbf{x}_{opt}^{(2)}$
$==$	$\preceq$	$\succ$	$\mathbf{x} = \mathbf{x}_{opt}^{(2)}$
$==$	$==$	$==$	No changes
$==$	$\sim$	$\sim$	Archive $\mathbf{x}_{opt}^{(2)}$
$\sim$	$\preceq$	$\succ$	$\mathbf{x} = \mathbf{x}_{opt}^{(2)}$
$\sim$	$\preceq$	$\sim$	$\mathbf{x} = \mathbf{x}_{opt}^{(2)}$, archive $\mathbf{x}_{opt}^{(1)}$
$\sim$	$==$	$\sim$	Archive $\mathbf{x}_{opt}^{(1)}$
$\sim$	$\sim$	$\preceq$	Archive $\mathbf{x}_{opt}^{(1)}$
$\sim$	$\sim$	$\succ$	Archive $\mathbf{x}_{opt}^{(2)}$
$\sim$	$\sim$	$\sim$	Archive $\mathbf{x}_{opt}^{(1)}$ and $\mathbf{x}_{opt}^{(2)}$
$\sim$	$\sim$	$==$	Archive $\mathbf{x}_{opt}^{(1)}$

4.2.5 Local Search Scheme

In the GSM framework for SO/MOO, a trust-region-regulated search strategy is utilized to ensure convergence to some local optimum or the global optimum of the exact computationally expensive fitness function [15][194][32][168], even though surrogate models are deployed in the local search. Note that the original trust-region framework [15] requires the first order(gradient) and zero order(function value) accuracy of the surrogate models, to ensure convergence to the local optima. However, in [32][194], it has also been shown that the use of inexact gradient information is sufficient for such convergence property.

For each individual in the GS-SO/MOMA population, the local refinement/search proceeds with a sequence of trust-region subproblems of the form

$$\begin{aligned} \text{minimize}: \quad & \hat{f}^{(k)}(\mathbf{x}_c^{(k)} + \mathbf{s}), \\ \text{subject to}: \quad & \|\mathbf{s}\| \leq \Psi^{(k)}, \end{aligned} \tag{Eq. 4.6}$$

where $k = 0, 1, 2, \ldots, k_{max}$, $\hat{f}(\mathbf{x})$ is the approximation function corresponding to the objective function $f(\mathbf{x})$. Meanwhile, $\mathbf{x}_c^{(k)}$, $\mathbf{s}$, and $\Psi^{(k)}$ represent the initial guess (current

best solution) at iteration k, an arbitrary step, and the trust-region radius at iteration k, respectively. In our experiments, the Sequential Quadratic Programming (SQP) [107] is used to minimize the sequence of subproblems in the approximated landscape.

During the local search, the initial trust-region radius Ψ is initialized based on the minimum and maximum values of the m design points used to construct the surrogate model (refer to line 11 of Algorithm 7 and line 13 of Algorithm 9). The trust-region radius for iteration k, *i.e.*, $\Psi^{(k)}$ is updated based on a measure which indicates the accuracy of the surrogate model at the k^{th} local optimum, $\mathbf{x}_{opt}^{(k)}$. This measure, $\rho^{(k)}$, provides a measure of the actual versus predicted change in the exact fitness function values at the k^{th} local optimum and is calculated as:

$$\rho^{(k)} = \frac{f(\mathbf{x}_c^{(k)}) - f(\mathbf{x}_{opt}^{(k)})}{\hat{f}(\mathbf{x}_c^{(k)}) - \hat{f}(\mathbf{x}_{opt}^{(k)})}. \qquad \text{(Eq. 4.7)}$$

The value of $\rho^{(k)}$ is then used to update the trust-region radius as follows [15]:

$$\begin{aligned} \Psi^{(k+1)} &= C_1\Psi^{(k)}, \quad \text{if } \rho^{(k)} \leq C_2, \\ &= \Psi^{(k)}, \quad \text{if } C_2 < \rho^{(k)} \leq C_3, \\ &= C_4\Psi^{(k)}, \quad \text{if } \rho^{(k)} > C_3, \end{aligned} \qquad \text{(Eq. 4.8)}$$

where C_1, C_2, C_3, and C_4 are constants. Typically, $C_1 \in (0, 1)$ and $C_4 \geq 1$ for the scheme to work efficiently. From the literatures of trust-region framework [15][142][210][209], suitable values are $C_1 = 0.25$, $C_2 = 0.25$, $C_3 = 0.75$, and $C_4 = 2$, if $||\mathbf{x}_{opt}^{(k)} - \mathbf{x}_c^{(k)}||_\infty = \Psi^{(k)}$ or $C_4 = 1$, if $||\mathbf{x}_{opt}^{(k)} - \mathbf{x}_c^{(k)}||_\infty < \Psi^{(k)}$.

The trust-region radius for the next iteration, $\Psi^{(k+1)}$, is reduced if the accuracy of the surrogate, measured by $\rho^{(k)}$ is low. On the other hand, $\Psi^{(k)}$ is doubled if the surrogate is found to be accurate and the k^{th} local optimum, $\mathbf{x}_{opt}^{(k)}$, lies on the trust-region bounds. Otherwise, the trust-region radius remains unchanged.

The initial guess of the optimum at iteration $k+1$ becomes

$$\begin{aligned} \mathbf{x}_c^{(k+1)} &= \mathbf{x}_{opt}^{(k)}, \quad \text{if } \rho^{(k)} > 0, \\ &= \mathbf{x}_c^{(k)}, \quad \text{if } \rho^{(k)} \leq 0. \end{aligned} \tag{Eq. 4.9}$$

The trust-region expansion and contraction for an individual terminates when the termination condition is satisfied. For instance, this termination condition could be when the trust-region radius Ψ approaches ε, where ε represents some small trust-region radius, or when a maximum number of iteration k_{term} is reached.

4.3 Empirical Study

In this section, an empirical study on the GSM framework for solving single and multi-objective optimization problems is presented. All experimental codes are written in C++ language. In the present study, we consider a diverse set of exact interpolating and generalizing approximation techniques for constructing the local surrogate models, *i.e.*, M_1 and M_2. These include the interpolating Kriging/Gaussian process (GP), interpolating linear spline radial basis function (RBF) and 2^{nd} order polynomial regression (PR). Greater details on GP, PR, and RBF can be found in Appendix A. To begin with, the user-specified parameters of the GSM framework are discussed as follows. Apart from the parameters of the underlying SO/MOEA, the generalized framework has three additional user-specified parameters: m, G_{db} and k_{term}.

Since model accuracy is highly dependent on the sufficiency of the m data points used for model building, the size of nearest neighboring points used is defined by $d + (d+1)(d+2)/2$, where d is the dimensionality of the optimization problem. From these m data points, as many as $(d+1)(d+2)/2$ among them[2] are chosen uniformly as the

[2]This amount corresponds to the minimum number of data points required for building quadratic regression models.

training data for building the surrogates, the remaining data points then form the set for validating the prediction quality of the surrogate. To choose the nearest points from the database, comparison based on Euclidean distance is used. In our implementation, the database size is also limited by only keeping the latest $10m$ points archived since it is generally the case that latest evaluated design points are more relevant to the current population. It is worth noting we do not particularly limit any detail implementation regarding the way to select the m points in this work. Clustering technique might also be considered for reducing the complexity in the case where database size is large. The computational cost of model building incurs a maximum of a few seconds for each model building, on a P-IV 1.6GHz 1GB RAM. For instance, it takes approximately 30 seconds, < 1 second, and 5 seconds to build a/an GP, PR and RBF model on a 30D problem, respectively. Thus, the computational cost for model building is regarded as negligible compared to the computational cost for a fitness evaluation of a real-world complex problem, which takes many minutes to hours.

Parameter G_{db}, on the other hand, defines the period of the database building phase (refer to line 4-5 in Algorithms 7 and 9) before the core operation of the GSM framework begins to take effect. Hence G_{db} can be adapted for different optimization problems according to the fulfillment on the requirement of parameter m. Therefore, the lower bound of G_{db} is defined by the period to acquire a minimum of m data points for construction of reliable surrogate models.

Theoretically, the trust-region local search scheme generally terminates when the trust-region radius, Ψ approaches ε, where ε represents some very small value for termination condition (refer to Section 4.2.5). Nevertheless, for practical reason, under limited computational budget, it is more appropriate to derive an appropriate value for k_{term} as the termination condition in the trust-region local search. In what follows, we present a theoretical bound for k_{term}:

$$\Psi_{min}^{(1)} (C_1)^{(k_{min})} \leq \varepsilon \quad \text{(Eq. 4.10)}$$

$$\Rightarrow (C_1)^{(k_{min})} \leq \frac{\varepsilon}{\Psi_{min}^{(1)}} \quad \text{(Eq. 4.11)}$$

$$\Rightarrow k_{min} \log C_1 \leq \log \frac{\varepsilon}{\Psi_{min}^{(1)}} \quad \text{(Eq. 4.12)}$$

Since $C_1 \in (0,1) \rightarrow \log C_1 < 0$, we arrive at:

$$\Rightarrow k_{min} \geq \left(\log \left(\frac{\varepsilon}{\Psi_{min}^{(1)}} \right) \right) / (\log C_1) \quad \text{(Eq. 4.13)}$$

$$\Rightarrow k_{min} \geq \log_{C_1} \left(\frac{\varepsilon}{\Psi_{min}^{(1)}} \right). \quad \text{(Eq. 4.14)}$$

Similarly, the maximum number of trust-region iterations in the local search, *i.e.*, k_{max}, is estimated by:

$$k_{max} < N_{succ}^{max} + N_{succ}^{max} \log_{C_1} \left(\frac{\varepsilon}{\Psi_{max}^{(1)}} \right) \quad \text{(Eq. 4.15)}$$

$$\Rightarrow k_{max} < N_{succ}^{max} \left(1 + \log_{C_1} \left(\frac{\varepsilon}{\Psi_{max}^{(1)}} \right) \right). \quad \text{(Eq. 4.16)}$$

Note that N_{succ}^{max} is the maximum number of successful iterations, while $\Psi_{min}^{(1)}$ and $\Psi_{max}^{(1)}$ are the lower and upper bounds of the initial trust-region radius. In effect, the bounds for k_{term} as the termination condition can be derived as:

$$\log_{C_1} \left(\frac{\varepsilon}{\Psi_{min}^{(1)}} \right) \leq k_{term} < N_{succ}^{max} \left(1 + \log_{C_1} \left(\frac{\varepsilon}{\Psi_{max}^{(1)}} \right) \right). \quad \text{(Eq. 4.17)}$$

In the trust-region-regulated local search, $\Psi^{(1)}$ depends on the local region of interest where the initial m nearest neighbors are located. Hence it is not possible to define this term precisely for any new optimization problem. For instance, if $\Psi_{min}^{(1)} \approx 10\varepsilon$ and $C_1 = 0.25$, we arrive at:

$$k_{term} \geq \frac{\log 0.1}{\log 0.25},$$

$$k_{term} \geq 1.66. \quad \text{(Eq. 4.18)}$$

As opposed to using $k_{term} = 1$ which translates to a single iteration local search, a minimum value of $k_{term} \geq 2$ is more practical to allow the mechanisms of the trust-region-regulated local search to take effect.

4.3.1 Single-Objective Optimization

Empirical study on the GS-SOMA is performed using ten benchmark problems (F1-F10) reported in [38][191] and summarized here in Table B.1. More detailed description of the problems is also provided in Appendix B.1. They consist of problems with diverse properties in terms of separability, multi-modality, and continuity.

Table 4.2: The benchmark problems used (F1-F10) for the empirical study of single-objective optimization.

Benchmark Problem	**Description**	**Global Optimum** $f(x^*)$
F1	Ackley	0.0
F2	Griewank	0.0
F3	Rosenbrock	0.0
F4	Shifted Rotated Rastrigin (F10 in [191])	-330.0
F5	Shifted Rotated Weierstrass (F11 in [191])	90.0
F6	Shifted Expanded Griewank plus Rosenbrock (F13 in [191])	-130.0
F7	Hybrid Composition Function (F15 in [191])	120.0
F8	Rotated Hybrid Composition Function (F16 in [191])	120.0
F9	Rotated Hybrid Composition Function with Narrow Basin Global Optimum (F19 in [191])	10.0
F10	Non-continuous Rotated Hybrid Composition Function (F23 in [191])	360.0

Table 4.3: Definition of the Single-Objective MAs (SOMAs) compared.

Algorithms	**Definition**
GA	No surrogate is used
SS-SOMA-GP	Single surrogate SOMA with M_1: GP
SS-SOMA-PR	Single surrogate SOMA with M_1: PR
SS-SOMA-RBF	Single surrogate SOMA with M_1: RBF
SS-SOMA-Perfect	Single surrogate SOMA with M_1: Perfect model
GS-SOMA	Generalized surrogate SOMA with M_1: *weighted-average* ensemble of GP, PR, and RBF M_2: PR

In this chapter, all the benchmark problems are configured with a dimensionality of $d = 30$ for SOO. Performance comparisons are then made between the GA, SS-SOMA-

Table 4.4: Setting of experiments for GA, SS-SOMA, SS-SOMA-Perfect, and GS-SOMA.

Parameters Setting	
Population size (N_{pop})	100
Crossover probability (P_{cross})	0.9
Mutation probability (P_{mut})	0.1
Maximum number of exact evaluations	8000
Evolutionary operators	uniform crossover & mutation, elitism and ranking selection
Number of trust region iteration(k_{term}) for SS-SOMA and GS-SOMA	3
Database building phase (G_{db}) for SS-SOMA and GS-SOMA (in number of generations)	20
Number of independent runs	20

GP, SS-SOMA-PR, SS-SOMA-RBF, SS-SOMA-Perfect, and GS-SOMA (refer to Table 4.3 for the definition of the algorithms investigated here). Note that to facilitate a fair comparison, the surrogate memetic variants are built on top of the same GA used in the study. Hence, this ensures that any improvements observed is a direct contribution of the surrogate framework considered. SS-SOMA-*XXX* refers to the different surrogate-assisted single-objective MA variants, each with a unique approximation method used to generate the surrogate model. For instance, *XXX* in SS-SOMA-*XXX* refers to GP, PR, or RBF. On the other hand, SS-SOMA-Perfect refers to an SS-SOMA that employs an imaginary approximation technique that generates error-free surrogates[3], *i.e.*, $rmse = 0$. Hence the notion of curse or blessing of uncertainty does not exist in the SS-SOMA-Perfect search. As such, any SS-SOMA-*XXX* that under/out-performs SS-SOMA-Perfect is clearly attributed to the effects of curse and blessing of uncertainty, respectively. Last but not least, GS-SOMA refers to the Generalized Surrogate framework for single-objective optimization. The common parameter settings of the algorithms used in the present experimental study are summarized in Table 4.4.

[3]An error-free surrogate model can be realized by using exact fitness function in the portion of SS-SOMA where a surrogate model should be used, but the incurred fitness evaluation is counted only as many as in SS-SOMA.

4.3.1.1 Experimental Results

In Tables 4.5-4.14, the detailed results from 20 independent runs of SS-SOMAs, GS-SOMA, and GA are presented. The GS-SOMA and best performing SS-SOMA are highlighted in the tables. Note that none of the SS-SOMAs always dominates in performance on all ten benchmark problems. This makes good sense since the performance of any surrogate-assisted evolutionary search would depend on the match between the characteristics of the problem landscape and approximation scheme used. For instance, in the tables, it is shown that SS-SOMA-PR serves to be best suited for F1, F5, and F9 since it outperforms all other algorithms on these problems. Similarly, this also applies to SS-SOMA-GP which excels on F3. On the other hand, SS-SOMA-RBF, though not superior, performs relatively well on F3, F4, F7, and F8. Moreover, it is worth noting that the SS-SOMAs are observed to have performed significantly poor on several occasions. For instance, SS-SOMA-PR fares badly on F3, F4, F7, and F8. The same is true for SS-SOMA-GP on F1, F4-F8, and F10, and SS-SOMA-RBF on F1, F2, F5, F6, F9, and F10.

Table 4.5: Solution quality at the end of 8000 exact function evaluations for F1 using GA, SS-SOMA-GP, SS-SOMA-PR, SS-SOMA-RBF, and GS-SOMA.

Optimization	Statistical Values				
Algorithm	Mean	Std. Dev.	Median	Best	Worst
GA	1.24e+01	9.50e-01	1.23e+01	1.12e+01	1.42e+01
SS-SOMA-GP	6.43e+00	9.73e-01	3.98e+00	2.87e+00	1.56e+01
SS-SOMA-PR	1.39e+00	1.93e-01	1.36e+00	1.14e+00	1.75e+00
SS-SOMA-RBF	4.91e+00	7.57e-01	4.86e+00	3.78e+00	6.09e+00
GS-SOMA	3.58e+00	5.09e-01	3.67e+00	2.87e+00	4.28e+00

Table 4.6: Solution quality at the end of 8000 exact function evaluations for F2 using GA, SS-SOMA-GP, SS-SOMA-PR, SS-SOMA-RBF, and GS-SOMA.

Optimization	Statistical Values				
Algorithm	Mean	Std. Dev.	Median	Best	Worst
GA	4.58e+01	8.61e+00	4.67e+01	2.15e+01	6.19e+01
SS-SOMA-GP	1.79e+01	8.58e+00	1.07e+01	5.15e-09	3.00e+01
SS-SOMA-PR	1.18e-02	2.78e-02	4.29e-08	7.48E-10	1.19e-01
SS-SOMA-RBF	7.49e-01	8.98e-02	7.51e-01	6.02e-01	8.72e-01
GS-SOMA	2.2e-03	4.60e-03	8.95e-09	1.40E-10	1.54e-02

Table 4.7: Solution quality at the end of 8000 exact function evaluations for F3 using GA, SS-SOMA-GP, SS-SOMA-PR, SS-SOMA-RBF, and GS-SOMA.

Optimization Algorithm	Statistical Values				
	Mean	Std. Dev.	Median	Best	Worst
GA	4.10e+02	1.01e+02	3.85e+02	2.33e+02	5.73e+02
SS-SOMA-GP	2.99e+01	7.73e-01	3.00e+01	2.87e+01	3.11e+01
SS-SOMA-PR	6.73e+01	2.55e+01	5.62e+01	3.72e+01	1.04e+02
SS-SOMA-RBF	4.90e+01	2.92e+01	3.97e+01	2.92e+01	1.57e+02
GS-SOMA	4.63e+01	2.92e+01	3.02e+01	2.83e+01	1.26e+02

Table 4.8: Solution quality at the end of 8000 exact function evaluations for F4 using GA, SS-SOMA-GP, SS-SOMA-PR, SS-SOMA-RBF, and GS-SOMA.

Optimization Algorithm	Statistical Values				
	Mean	Std. Dev.	Median	Best	Worst
GA	-5.46e+01	3.01e+01	-5.48e+01	-1.11e+02	5.19e-01
SS-SOMA-GP	-1.19e+02	1.87e+01	-1.17e+02	-1.50e+02	-8.71e+01
SS-SOMA-PR	-1.19e+02	1.23e+01	-1.21e+02	-1.43e+02	-9.01e+01
SS-SOMA-RBF	-1.65e+02	1.86e+01	-1.66e+02	-1.91e+02	-1.36e+02
GS-SOMA	-1.26e+02	1.60e+01	-1.23e+02	-1.64e+02	-9.97e+01

Table 4.9: Solution quality at the end of 8000 exact function evaluations for F5 using GA, SS-SOMA-GP, SS-SOMA-PR, SS-SOMA-RBF, and GS-SOMA.

Optimization Algorithm	Statistical Values				
	Mean	Std. Dev.	Median	Best	Worst
GA	1.26e+02	2.85e+00	1.26e+02	1.20e+02	1.32e+02
SS-SOMA-GP	1.19e+02	4.29e+00	1.20e+02	1.12e+02	1.25e+02
SS-SOMA-PR	5.67e+01	3.79e+00	1.16e+02	1.13e+02	1.25e+02
SS-SOMA-RBF	1.21e+02	2.61e+00	1.21e+02	1.18e+02	1.24e+02
GS-SOMA	1.19e+02	3.05e+00	1.19e+02	1.14e+02	1.24e+02

Table 4.10: Solution quality at the end of 8000 exact function evaluations for F6 using GA, SS-SOMA-GP, SS-SOMA-PR, SS-SOMA-RBF, and GS-SOMA.

Optimization Algorithm	Statistical Values				
	Mean	Std. Dev.	Median	Best	Worst
GA	-9.57e+01	9.43e+00	-9.79e+01	-1.06e+02	-7.28e+01
SS-SOMA-GP	-1.02e+02	2.99e+00	-1.03e+02	-1.05e+02	-9.74e+02
SS-SOMA-PR	-1.06e+02	2.45e+00	-1.07e+02	-1.09e+02	-1.02e+02
SS-SOMA-RBF	-1.03e+02	2.43e+00	-1.03e+02	-1.07e+02	-9.96e+01
GS-SOMA	-1.12e+02	1.05e+00	-1.23e+02	-1.13e+02	-1.11e+02

Table 4.11: Solution quality at the end of 8000 exact function evaluations for F7 using GA, SS-SOMA-GP, SS-SOMA-PR, SS-SOMA-RBF, and GS-SOMA.

Optimization Algorithm	Statistical Values				
	Mean	Std. Dev.	Median	Best	Worst
GA	7.29e+02	5.92e+01	7.27e+02	6.43e+02	8.21e+02
SS-SOMA-GP	6.81e+02	7.23e+01	6.95e+02	6.02e+02	8.23e+02
SS-SOMA-PR	6.42e+02	5.80e+01	6.34e+02	5.73e+02	7.09e+02
SS-SOMA-RBF	6.27e+02	7.93e+01	5.99e+02	5.95e+02	8.49e+02
GS-SOMA	6.07e+02	3.06e+01	6.00e+02	5.79e+02	6.59e+02

Table 4.12: Solution quality at the end of 8000 exact function evaluations for F8 using GA, SS-SOMA-GP, SS-SOMA-PR, SS-SOMA-RBF, and GS-SOMA.

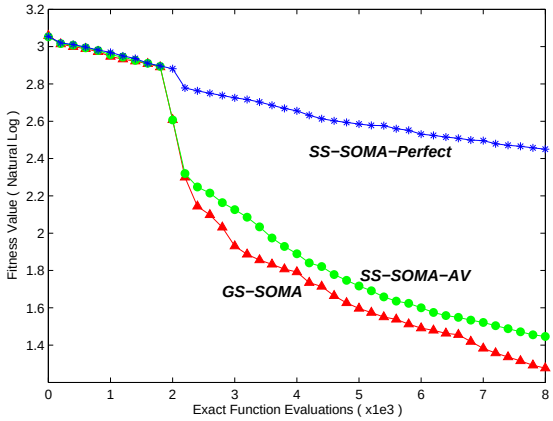
Optimization Algorithm	Statistical Values				
	Mean	Std. Dev.	Median	Best	Worst
GA	4.83e+02	6.3e+01	4.62e+02	4.19e+02	6.06e+02
SS-SOMA-GP	4.52e+02	9.66e+01	4.35e+02	3.40e+02	5.63e+02
SS-SOMA-PR	3.94e+02	4.41e+01	3.75e+02	3.43e+02	4.52e+02
SS-SOMA-RBF	3.79e+02	3.3e+01	3.69e+02	3.51e+02	4.41e+02
GS-SOMA	3.25e+02	1.17e+02	2.86e+02	2.32e+02	5.54e+02

Table 4.13: Solution quality at the end of 8000 exact function evaluations for F9 using GA, SS-SOMA-GP, SS-SOMA-PR, SS-SOMA-RBF, and GS-SOMA.

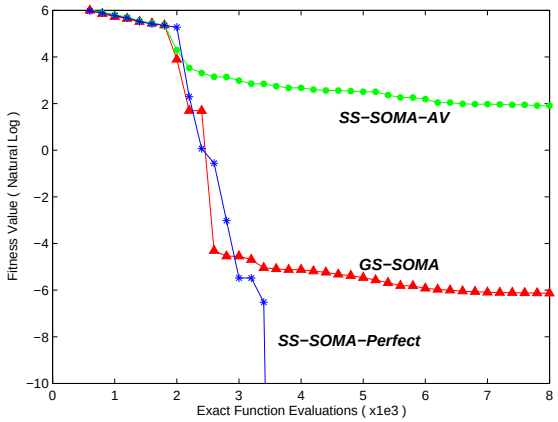
Optimization Algorithm	Statistical Values				
	Mean	Std. Dev.	Median	Best	Worst
GA	1.02e+03	2.35e+01	1.02e+03	9.86e+02	1.08e+03
SS-SOMA-GP	9.42e+02	1.71e+01	9.37e+02	9.25e+02	9.81e+02
SS-SOMA-PR	9.32e+02	8.26e+00	9.31e+02	9.22e+02	9.48e+02
SS-SOMA-RBF	9.81e+02	1.43e+01	9.80e+02	9.67e+02	1.00e+03
GS-SOMA	9.42e+02	1.75e+01	9.37e+02	9.30e+02	9.86e+02

Table 4.14: Solution quality at the end of 8000 exact function evaluations for F10 using GA, SS-SOMA-GP, SS-SOMA-PR, SS-SOMA-RBF, and GS-SOMA.

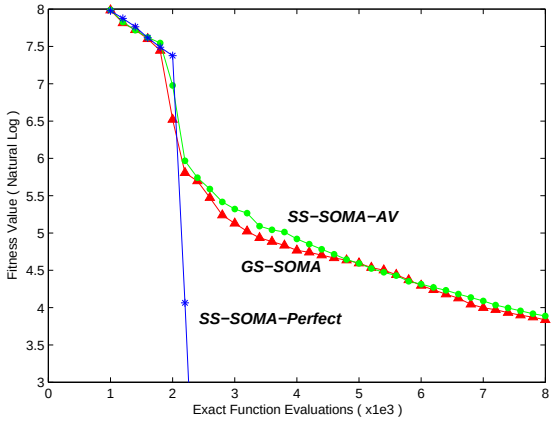
Optimization Algorithm	Statistical Values				
	Mean	Std. Dev.	Median	Best	Worst
GA	1.51e+03	5.52e+01	1.52e+03	1.40e+03	1.58e+03
SS-SOMA-GP	1.26e+03	1.88e+02	1.22e+03	1.03e+03	1.54e+03
SS-SOMA-PR	1.07e+03	1.07e+02	1.04e+03	9.42e+02	1.29e+03
SS-SOMA-RBF	1.12e+03	1.16e+02	1.15e+03	9.59e+02	1.28e+03
GS-SOMA	1.01e+03	7.85e+01	9.53e+02	9.09e+02	1.51e+03



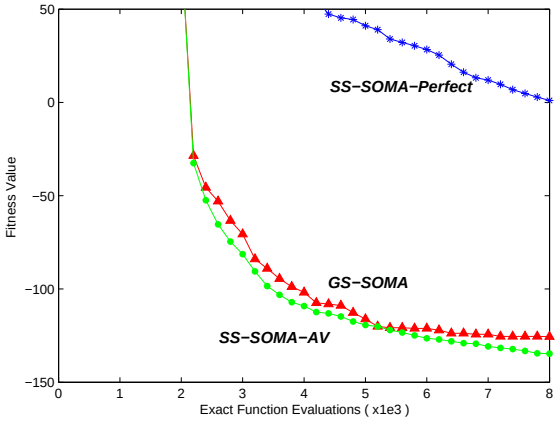
4.4.a: Convergence trends of F1 obtained from GS-SOMA, SS-SOMA-Perfect, and SS-SOMA-AV.



4.4.b: Convergence trends of F2 obtained from GS-SOMA, SS-SOMA-Perfect, and SS-SOMA-AV.

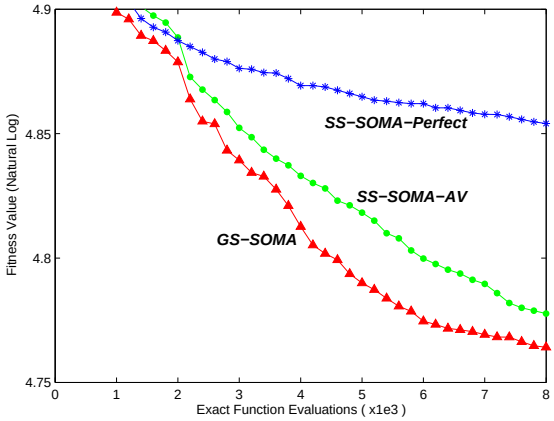


4.4.c: Convergence trends of F3 obtained from GS-SOMA, SS-SOMA-Perfect, and SS-SOMA-AV.

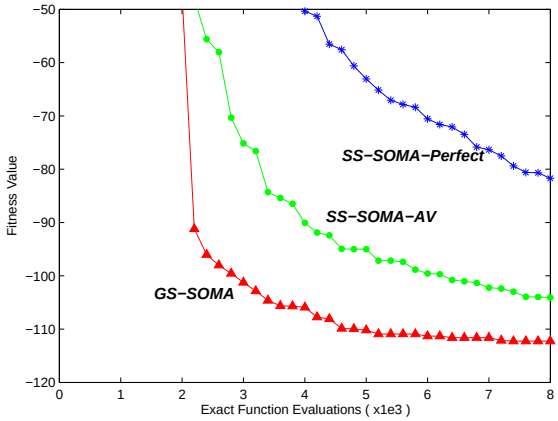


4.4.d: Convergence trends of F4 obtained from GS-SOMA, SS-SOMA-Perfect, and SS-SOMA-AV.

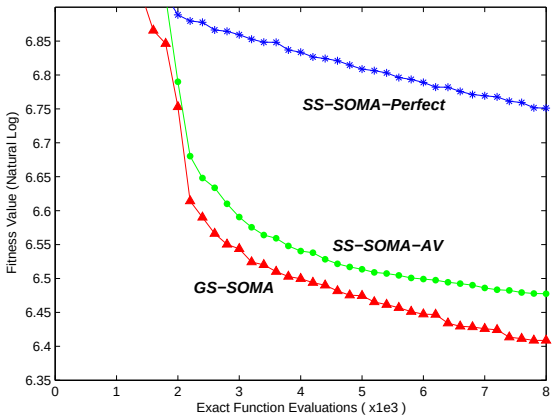
Figure 4.4: Convergence trends of F1-F4.



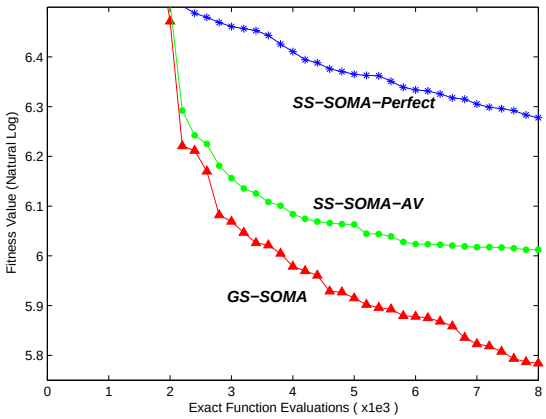
4.5.a: Convergence trends of F5 obtained from GS-SOMA, SS-SOMA-Perfect, and SS-SOMA-AV.



4.5.b: Convergence trends of F6 obtained from GS-SOMA, SS-SOMA-Perfect, and SS-SOMA-AV.

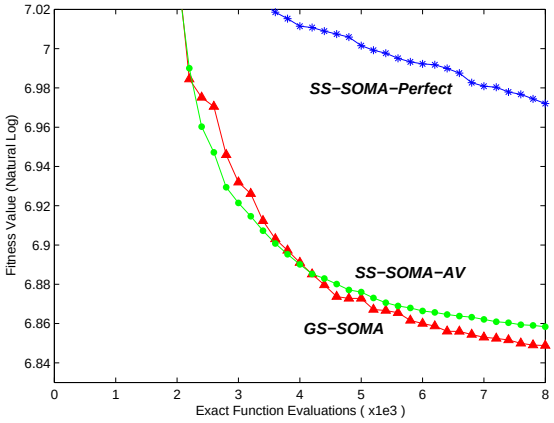


4.5.c: Convergence trends of F7 obtained from GS-SOMA, SS-SOMA-Perfect, and SS-SOMA-AV.

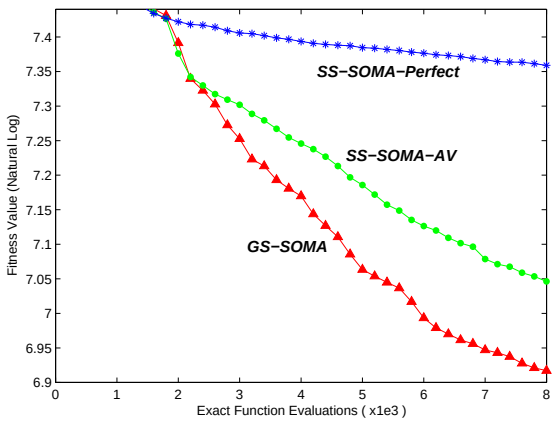


4.5.d: Convergence trends of F8 obtained from GS-SOMA, SS-SOMA-Perfect, and SS-SOMA-AV.

Figure 4.5: Convergence trends of F5-F8.



4.6.a: Convergence trends of F9 obtained from GS-SOMA, SS-SOMA-Perfect, and SS-SOMA-AV.



4.6.b: Convergence trends of F10 obtained from GS-SOMA, SS-SOMA-Perfect, and SS-SOMA-AV.

Figure 4.6: Convergence trends of F9-F10.

On the other hand, the results in Tables 4.5-4.14, indicate that GS-SOMA consistently performs well on all the benchmark problems. The *t-test* at 95% confidence level for the different algorithms, as reported in Table 4.15, confirms that GS-SOMA outperforms or is competitive to the SS-SOMAs on 43/50 cases. On the remaining 7 cases, GS-SOMA also displays solution qualities close to that of the superior SS-SOMA, see the highlighted results in Tables 4.5-4.14. Note that this is a significant achievement considering that no *a priori* knowledge is available to select an appropriate surrogate modeling scheme for the problems considered. This highlights the reliability of the generalized framework.

The search convergence trends of GS-SOMA, SS-SOMA-AV, and SS-SOMA-Perfect are also plotted in Fig. 4.4, 4.5, and 4.6. Note that SS-SOMA-AV represents the estimated performance one might expect to get when an approximation technique is randomly chosen for use. Hence, SS-SOMA-AV is generated from the average of the results obtained by all 3 SS-SOMAs, *i.e.*, SS-SOMA-GP, SS-SOMA-PR, and SS-SOMA-RBF. It is evident from the search convergence trends that GS-SOMA is superior over SS-SOMA-AV on the 10 benchmark problems. This indicates that the generalized framework is more

Table 4.15: *p-value* of *t-test* with 95% confidence level comparing statistical values for GS-SOMA and those of SS-SOMA-GP, SS-SOMA-PR, SS-SOMA-RBF, SS-SOMA-Perfect on F1-F10 ($s+$, $s-$, or $\approx$ indicates that GS-SOMA is significantly better, significantly worse, or indifferent, respectively).

	GA	SS-SOMA-GP	SS-SOMA-PR	SS-SOMA-RBF	SS-SOMA-Perfect
F1	<0.0001($s+$)	<0.0001($s+$)	<0.0001($s-$)	<0.0001($s+$)	<0.0001($s+$)
F2	<0.0001($s+$)	<0.0001($s+$)	0.1348($\approx$)	<0.0001($s+$)	0.0203($s-$)
F3	<0.0001($s+$)	0.0164($s-$)	0.0203($s+$)	0.7716($\approx$)	<0.0001($s-$)
F4	<0.0001($s+$)	0.2111($\approx$)	0.1291($\approx$)	<0.0001($s-$)	<0.0001($s+$)
F5	<0.0001($s+$)	1.0000($\approx$)	<0.0001($s-$)	0.0319($s+$)	<0.0001($s+$)
F6	<0.0001($s+$)	<0.0001($s+$)	<0.0001($s+$)	<0.0001($s+$)	<0.0001($s+$)
F7	<0.0001($s+$)	<0.0001($s+$)	0.0221($s+$)	0.2993($\approx$)	<0.0001($s+$)
F8	<0.0001($s+$)	0.0006($s+$)	0.0182($s+$)	0.0542($\approx$)	<0.0001($s+$)
F9	<0.0001($s+$)	1.0000($\approx$)	0.0264($s-$)	<0.0001($s+$)	<0.0001($s+$)
F10	<0.0001($s+$)	<0.0001($s+$)	0.0503($\approx$)	0.0012($s+$)	<0.0001($s+$)

reliable when one has no knowledge on the suitability of the approximation scheme for the problem in hand.

4.3.1.2 Analyzing the Generalized Evolutionary Framework in Single-Objective Optimization

To gain a better understanding of the generalized framework, we further analyze the reliability and effectiveness of the ensemble (M_1) and smoothing (M_2) surrogate models in contributing to the evolutionary search.

To facilitate this, the normalized root mean square errors ($N-rmse$) of fitness predictions based on the ensemble surrogate model, *i.e.*, M_1 in GS-SOMA search, for the benchmark problems are presented in Fig. 4.7. The $N-rmse$ of model i is determined as follows:

$$\text{Normalized } rmse_i = \frac{rmse_i}{\sum_{j=1}^{s} rmse_j}, \qquad \text{(Eq. 4.19)}$$

where s is the total approximation methods used in shaping the ensemble. From these figures, the consistently low $N-rmse$ of the ensemble model generated in the GS-SOMA search across all benchmark problems, demonstrates the high reliability of the fitness prediction generated by M_1 across the different optimization problems over any single surrogates.

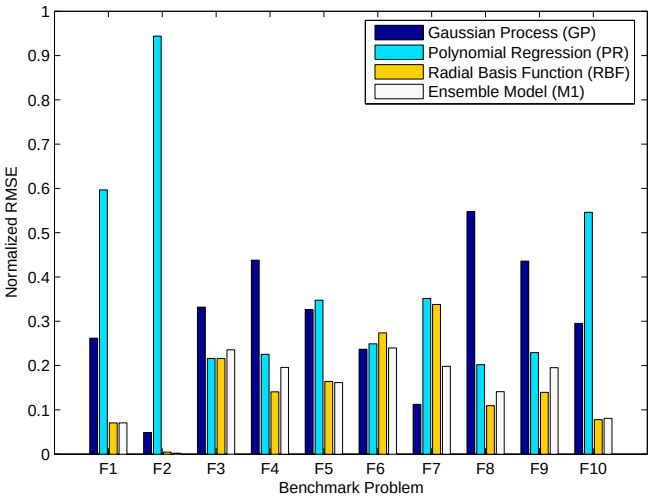


Figure 4.7: The normalized rmse by GP, PR, RBF, and weighted average ensemble.

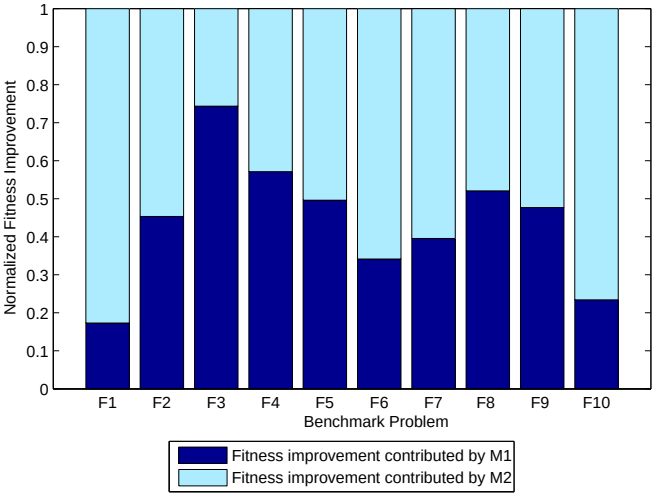


Figure 4.8: The normalized fitness improvement during the runs of GS-SOMA contributed by M_1 (Imp_{M1}) and M_2 (Imp_{M2}).

Further, it is worth noting that the use of M_2 contributes to the fitness improvement in GS-SOMA, which confirms the possible benefits of bless of uncertainty in surrogate model. The normalized average fitness improvement of the local searches contributed via the use of M_1 (Imp_{M1}) and M_2 (Imp_{M2}) during the GS-SOMA searches are summarized in Fig. 4.8 and is defined by:

$$\begin{aligned} \text{Normalized } Imp_{M1} &= \frac{Imp_{M1}}{Imp_{M1}+Imp_{M2}}, \\ \text{Normalized } Imp_{M2} &= \frac{Imp_{M2}}{Imp_{M1}+Imp_{M2}}. \end{aligned} \qquad \text{(Eq. 4.20)}$$

Imp_{M1} is the total fitness improvements attained by local refinements, *i.e.*, through Lamarckian learning, when $f(\mathbf{x}_{opt}^{(1)}) < f(\mathbf{x}_{opt}^{(2)})$, while Imp_{M2} is the total fitness improvements when $f(\mathbf{x}_{opt}^{(2)}) < f(\mathbf{x}_{opt}^{(1)})$.

From the statistical results given in Fig. 4.8, it is notable that M_1 and M_2 surrogates have contributed to the surrogate-assisted memetic search in their unique ways. This provides a means for explaining the results that were obtained in Fig. 4.4-4.6 and Tables 4.5-4.14. In particular, the reason for that all surrogate-assisted SOMAs outperform SS-SOMA-Perfect on F1 (Ackley) suggests the presence of *'bless of uncertainty'* through the use of surrogate(s), since the notion of curse or blessing of uncertainty cannot exist in the latter. Further, SS-SOMA-PR is shown to be most superior on F1 (Ackley) highlights the strength of the PR model in contributing to the search via smoothing the rugged landscape of the Ackley function. This hypothesis is clearly supported by the large portion of fitness improvements that are contributed by $M2$ (*i.e.*, the PR model) on F1, see Fig. 4.8. On the other hand, neither SS-SOMAs nor GS-SOMA manage to outperform the SS-SOMA-Perfect on F3(Rosenbrock), suggesting the presence of *'curse of uncertainty'* due to the surrogate(s). Further, the results in F3 of Fig. 4.8 also indicate that M_2 (*i.e.*, the smoothing PR model) did not contribute significantly to the search since the problem landscape of this function is originally smooth. Rather, the

use of ensemble model in GS-SOMA had contributed to reliable fitness improvement on F3(Rosenbrock) by generating reliable prediction accuracy. On the other test problems, both M_1 and M_2 surrogates were shown to contribute significantly to GS-SOMA in their own unique ways.

4.3.2 Multi-Objective Optimization

In this subsection, we present the empirical study of the GS-MOMA on six moderate to high dimensional MO benchmark problems, labeled here as MF1-MF6 [212]. The MO benchmark problems used in the study are summarized in Appendix B.2.

Table 4.16: Definition of the Multi-Objective MAs (MOMAs) compared

Algorithms	**Definition**
NSGA-II	No surrogate is used
GS-MOMA	Generalized surrogate MOMA with M_1: *weighted-average* ensemble of GP, PR, and RBF M_2: PR
SS-MOMA-I	Single surrogate MOMA with M_1: Ensemble of GP, PR, and RBF
SS-MOMA-II	Single surrogate MOMA with M_1: PR
SS-MOMA-Perfect	Single surrogate MOMA with M_1: Perfect model

Performance comparisons are then made between the standard non-dominated sorting genetic algorithm-II (NSGA-II) [36] and variants of MOMA. For fair comparison, GS-MOMA is compared with several SS-MOMAs and the NSGA-II since the formers are demonstrated with NSGA-II as the baseline by building on top of it. Hence, all algorithms compared inherit the same evolutionary operators as the NSGA-II used in our experiment. In SS-MOMAs, an offspring will be replaced in the spirit of Lamarckian learning during local search if its aggregated fitness function is found to be better than the original offspring. Similarly, SS-MOMA-Perfect is introduced here to assess the effects of approximation error on surrogate-assisted evolutionary search performance. For the sake of brevity, the notations and definitions of the MO algorithms studied are tabulated

in Table 4.16 while the common parameter settings of the MO algorithms used in the experimental study are defined in Table 4.17[4].

Table 4.17: Setting of experiments for NSGA-II, GS-MOMA, and SS-MOMA.

Parameters Setting	
Population size (N_{pop})	100
Crossover probability (P_{cross})	0.9
Mutation probability (P_{mut})	0.1
Maximum number of exact evaluations	MF1-MF2: 8000 MF3-MF4: 16000 MF5: 30000 MF6: 20000
Evolutionary operators	simulated binary crossover, polynomial mutation, binary tournament selection, elitism, non-domination rank, and crowded distance
Number of trust region iteration(k_{term}) for SS-MOMA and GS-MOMA	2
Database building phase (G_{db}) for SS-MOMA and GS-MOMA (in number of generations)	MF1-MF2, MF5-MF6: 10 MF3-MF4: 20
Number of independent runs	20

Many performance indicators exist for assessing the performance of MOEAs, such as those summarized in [216][33]. Here, the following three performance indicators are used, *i.e.*,

- **Generational Distance (GD)** [200][199]: This measurement indicates the gap between the true Pareto front (PF^*) and the evolved Pareto front (PF). Mathematically, it can be formulated as:

$$GD = \frac{1}{n_{PF}} \sqrt{\sum_{i=1}^{n_{PF}} {d_i}^2}, \qquad \text{(Eq. 4.21)}$$

  where n_{PF} is the number of members in PF, d_i is the Euclidean distance (in objective space) between member i of PF and its nearest member in PF^*. A low value of GD is more desirable since it reflects a good convergence to the true Pareto fronts.

[4]Since MF3 and MF4 have higher dimensionality, *i.e.*, $d = 50$, greater initial database size is required. For these cases, G_{db} is set to 20.

- **Maximum Spread (MS)** [211]: It is used to measure how well the true Pareto front (PF^*) is covered by the evolved Pareto front (PF). The MS measurement used in this chapter is formulated as:

$$MS = \sqrt{\frac{1}{r}\sum_{i=1}^{r}\left[\frac{\min(f_i^{max}, F_i^{max}) - \max(f_i^{min}, F_i^{min})}{F_i^{max} - F_i^{min}}\right]^2}, \qquad \text{(Eq. 4.22)}$$

where r is the number of objectives, f_i^{max} and f_i^{min} are the maximum and minimum of the i^{th} objective in the evolved PF, respectively. F_i^{max} and F_i^{min} are the maximum and minimum of the i^{th} objective in PF^*, respectively. Higher value of MS reflects a larger area of PF^* covered by PF, which is desirable.

- **Hypervolume Ratio (HR)** [199]: This indicates the ratio between the hyper-area/hypervolume (H) [214] dominated by the evolved PF and PF^*, where HR is defined as:

$$HR = \frac{H(PF)}{H(PF^*)},$$
$$H = volume\left(\bigcup_{i=1}^{n_{PF}} v_i\right). \qquad \text{(Eq. 4.23)}$$

Here, v_i denotes the hypercube constructed from member i of a particular Pareto front and the reference point. A HR value close to 1 indicates that the evolved Pareto front is quite close to the true Pareto front, in both convergence and spread of solutions.

4.3.2.1 Experimental Results

The obtained Pareto fronts of the benchmark problems for 20 independent runs are combined and depicted in Figs. 4.11-4.16. The respective performance metrics are then summarized in Figs. 4.17-4.22. From these results, all surrogate-assisted multi-objective evolutionary algorithms, *i.e.*, SS-MOMAs and GS-MOMA, are shown to outperform the

standard NSGA-II on MF1, MF2, MF5, and MF6. MF6 (ZDT4) is generally regarded as a challenging problem and hence commonly used by many in the literature. Here, our results are validated on ZDT4 against those obtained by Deb *et al.* in [37]. While [37] reported to solve ZDT4 with from 21781 to 22730 exact function evaluations with an achieved spread measure[5] of 0.332 to 0.422, GS-MOMA requires only 20000 exact evaluations at a competitive spread measure of 0.410 ± 0.046. On MF3 and MF4, some SS-MOMAs perform competitively or slightly poorer than NSGA-II (see Figs. 4.13.d and 4.14.d). On the other hand, GS-MOMA searches more efficiently than all the SS-MOMA variants and NSGA-II on the 6 benchmark problems considered. Note that GS-MOMA also outperforms the SS-MOMA-Perfect on a majority of the MOO benchmarks with respect to all three performance metrics, thus suggesting the positive synergy of the ensemble and smoothing surrogate models in the GSM framework.

4.3.2.2 Analyzing the Generalized Evolutionary Framework in Multi-Objective Optimization

To arrive at better understanding of the generalized framework in the context of multi-objective optimization, we analyze next the reliability and effectiveness of the ensemble (M_1) and smoothing (M_2) surrogate models in contributing to evolutionary search.

The $N - rmse$, *i.e.*, see equation Eq. 4.19, of fitness predictions based on GP, PR, RBF, or ensemble in GS-MOMA is summarized in Fig. 4.9. From the results, the ensemble model, M_1, is shown to arrive at low $N - rmse$ on all the multi-objective test problems considered, which is consistent with observations obtained in the single-objective context. M_1 generates high reliability predictions in comparison to other single surrogate model counterparts, *i.e.*, GP, PR or RBF.

[5]The spread metric [35] considers the distance between two extreme ends of Pareto front as well as the uniformity of distribution for solutions between the two extremes. This metric may be used for measuring the diversity of converged Pareto fronts. Note that a lower spread metric is desirable.

Besides $N - rmse$, the *solution archiving to replacement ratio*, labelled here as Γ, of the GS-MOMA search is also reported in Figure 4.10. Γ indicates the degree of solution diversity (through archival of new non-dominating solutions) against search convergence (through the process of Lamarckian learning replacement) in the GS-MOMA search. While Lamarckian learning helps to speedup convergence towards the desired Pareto front, the large Γ ratio observed on all benchmark problems implies frequent discovery of potential non-dominating solutions when using both M_1 and M_2 with local refinements. This suggests *'bless of uncertainty'* may take the form of faster search convergence and better solution diversity in the context of multi-objective evolutionary search.

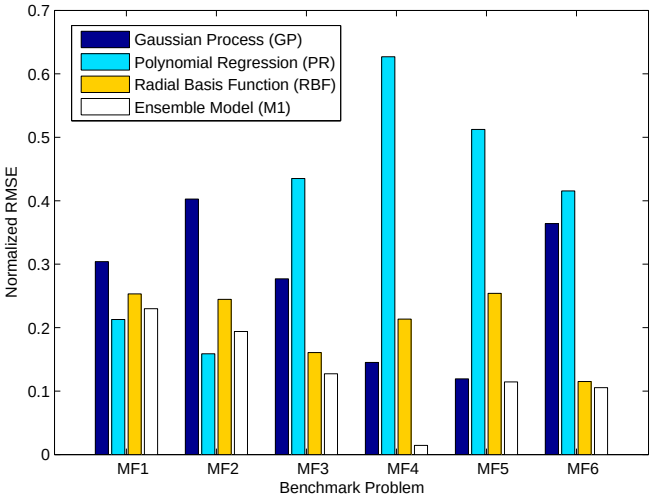


Figure 4.9: The normalized rmse by GP, PR, RBF, and weighted average ensemble on MF1-MF6.

4.3.3 Computational Complexity

In this subsection, an analytical study on the computational complexity of the GSM framework is presented. The computational effort, referred here by T_{comp}, of GS-SOMA or GS-MOMA is formulated as follows:

$$T_{comp} = G_{db}N_{pop}N_{obj} + (G_{max} - G_{db})[N_{pop} \\ (T_{ens} + T_{PR} + 2T_f k_{term} N_{obj} + T_{overhead})] \quad \text{(Eq. 4.24)}$$

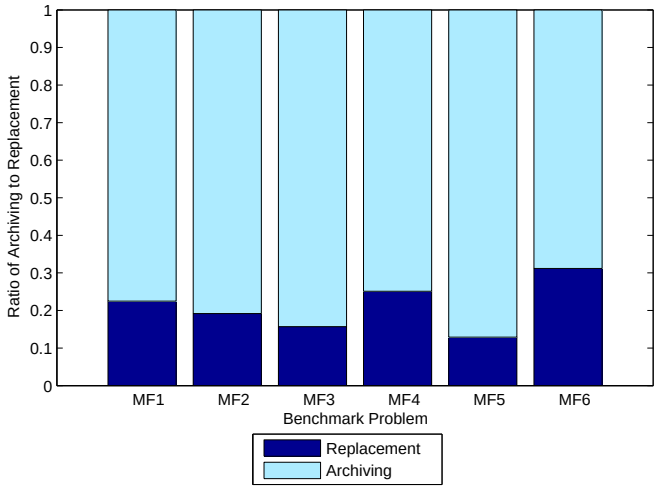
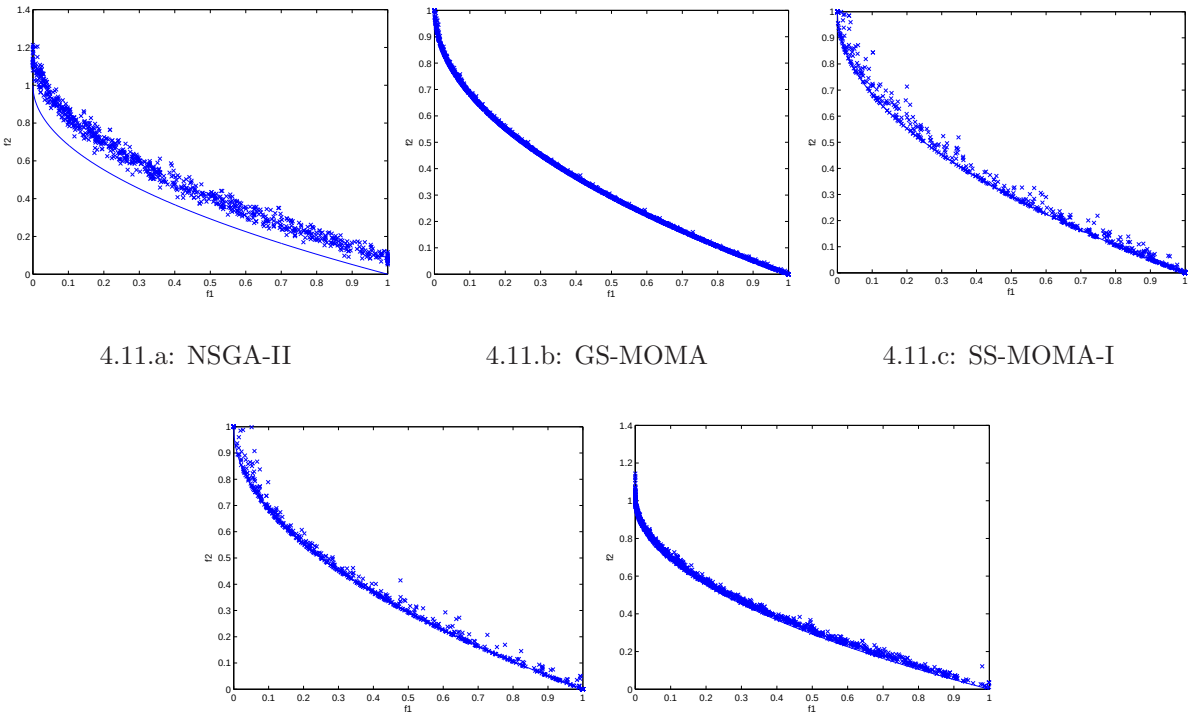


Figure 4.10: Archiving to Replacement Ratio of GS-MOMA on MF1-MF6.



4.11.a: NSGA-II

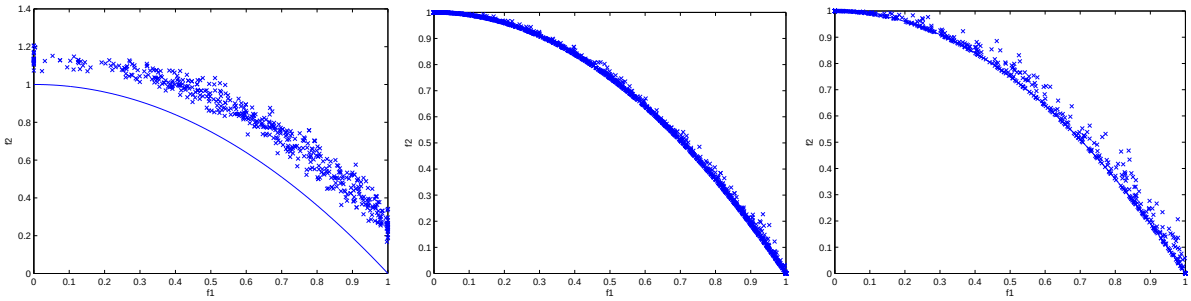
4.11.b: GS-MOMA

4.11.c: SS-MOMA-I

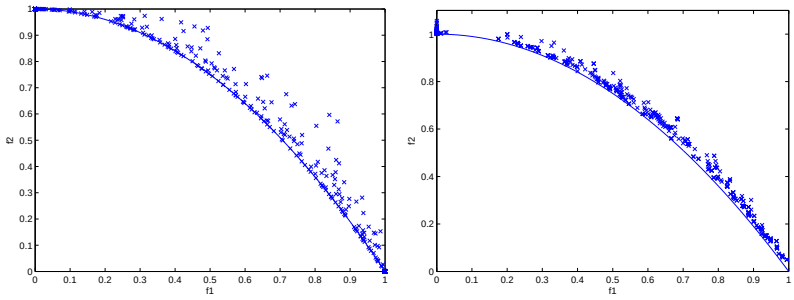
4.11.d: SS-MOMA-II

4.11.e: SS-MOMA-Perfect

Figure 4.11: Pareto Front evolved for benchmark problem MF1 in NSGA-II, GS-MOMA, SS-MOMA-I, SS-MOMA-II, and SS-MOMA-Perfect.

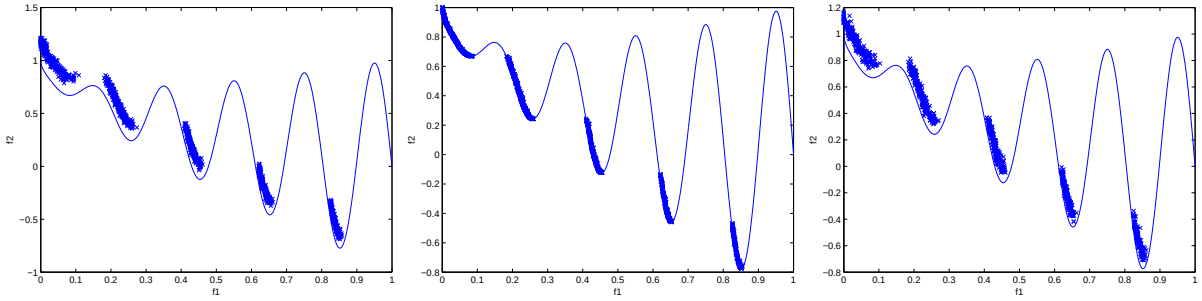


4.12.a: NSGA-II 4.12.b: GS-MOMA 4.12.c: SS-MOMA-I



4.12.d: SS-MOMA-II 4.12.e: SS-MOMA-Perfect

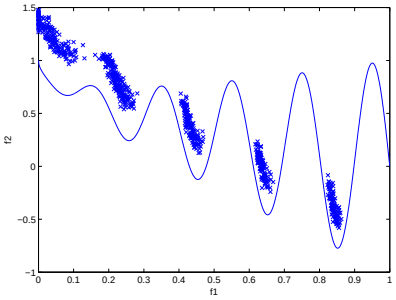
Figure 4.12: Pareto Front evolved for benchmark problem MF2 in NSGA-II, GS-MOMA, SS-MOMA-I, SS-MOMA-II, and SS-MOMA-Perfect.



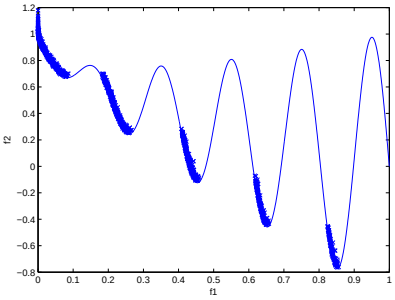
4.13.a: NSGA-II

4.13.b: GS-MOMA

4.13.c: SS-MOMA-I

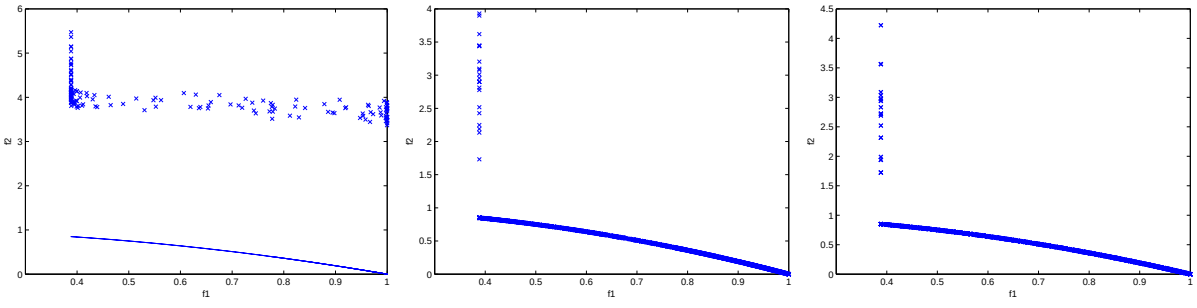


4.13.d: SS-MOMA-II

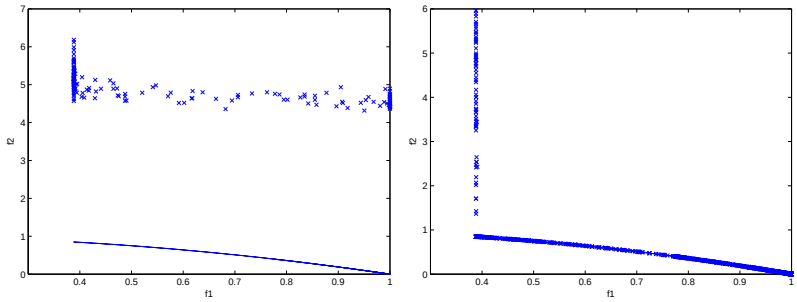


4.13.e: SS-MOMA-Perfect

Figure 4.13: Pareto Front evolved for benchmark problem MF3 in NSGA-II, GS-MOMA, SS-MOMA-I, SS-MOMA-II, and SS-MOMA-Perfect.

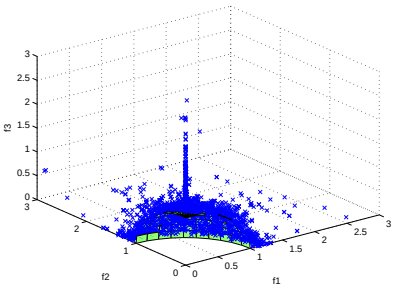


4.14.a: NSGA-II 4.14.b: GS-MOMA 4.14.c: SS-MOMA-I

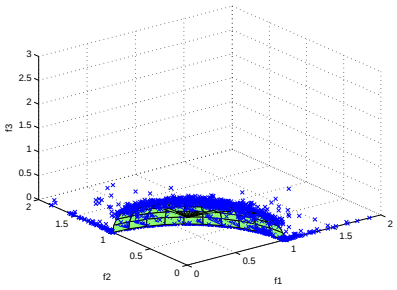


4.14.d: SS-MOMA-II 4.14.e: SS-MOMA-Perfect

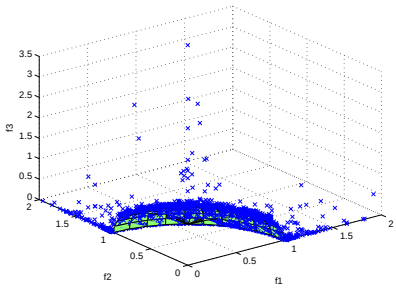
Figure 4.14: Pareto Front evolved for benchmark problem MF4 in NSGA-II, GS-MOMA, SS-MOMA-I, SS-MOMA-II, and SS-MOMA-Perfect.



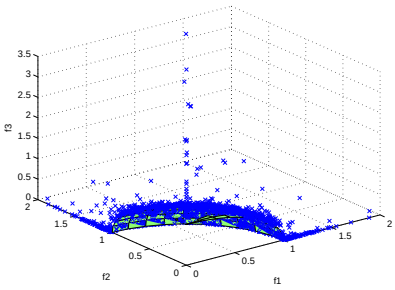
4.15.a: NSGA-II



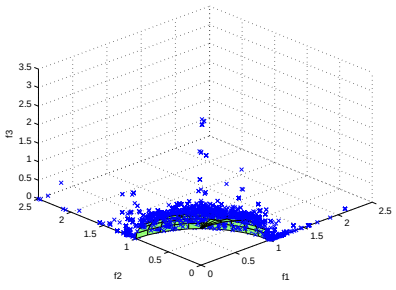
4.15.b: GS-MOMA



4.15.c: SS-MOMA-I

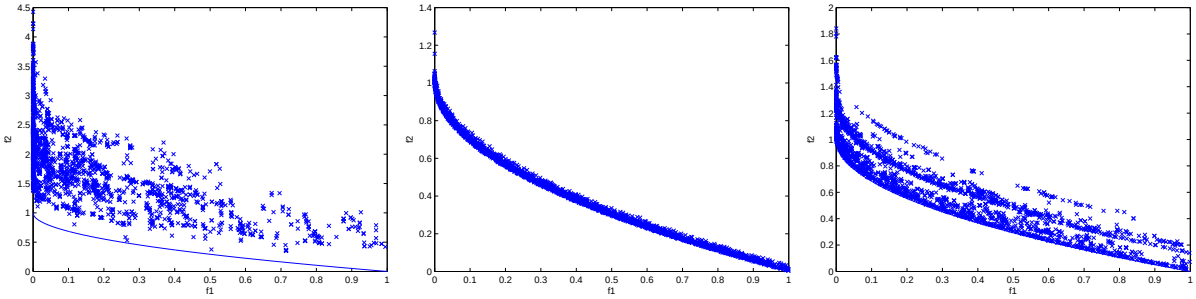


4.15.d: SS-MOMA-II



4.15.e: SS-MOMA-Perfect

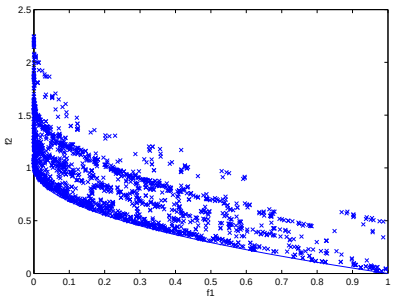
Figure 4.15: Pareto Front evolved for benchmark problem MF5 in NSGA-II, GS-MOMA, SS-MOMA-I, SS-MOMA-II, and SS-MOMA-Perfect.



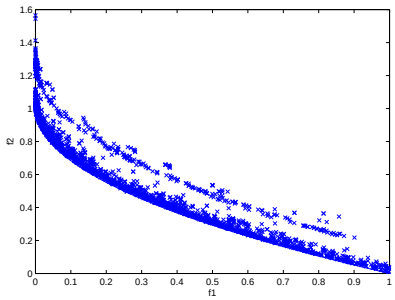
4.16.a: NSGA-II

4.16.b: GS-MOMA

4.16.c: SS-MOMA-I

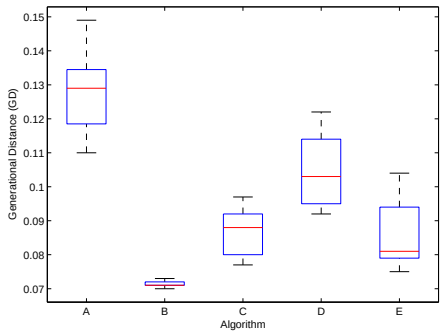


4.16.d: SS-MOMA-II

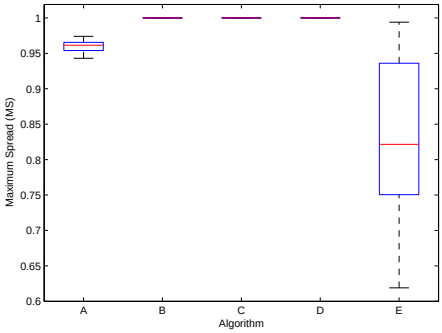


4.16.e: SS-MOMA-Perfect

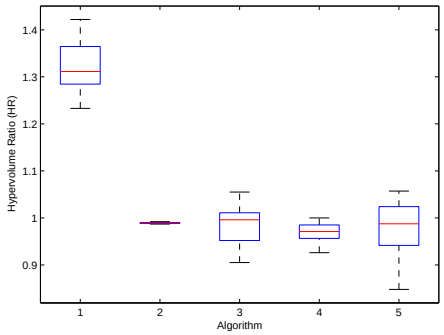
Figure 4.16: Pareto Front evolved for benchmark problem MF6 in NSGA-II, GS-MOMA, SS-MOMA-I, SS-MOMA-II, and SS-MOMA-Perfect.



4.17.a: Generational Distance (GD)

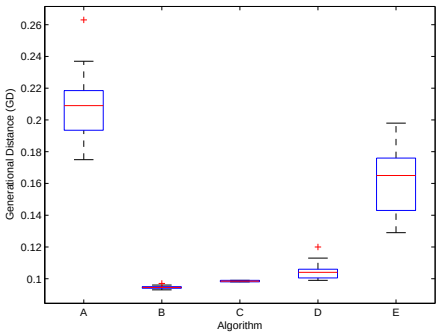


4.17.b: Maximum Spread (MS)

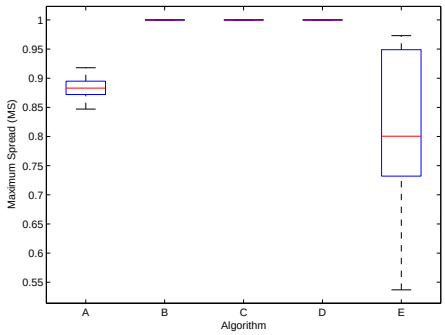


4.17.c: Hypervolume Ratio (HR)

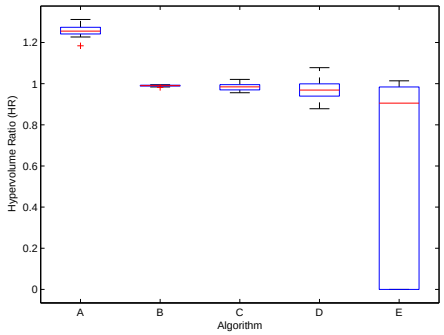
Figure 4.17: Generational Distance(GD), Maximum Spread(MS), and Hypervolume Ratio(HR) performance metrics for benchmark problem MF1. (A:NSGA-II, B:GS-MOMA, C:SS-MOMA-I, D:SS-MOMA-II, E:SS-MOMA-Perfect)



4.18.a: Generational Distance (GD)

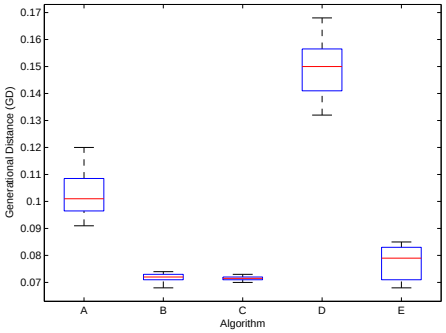


4.18.b: Maximum Spread (MS)

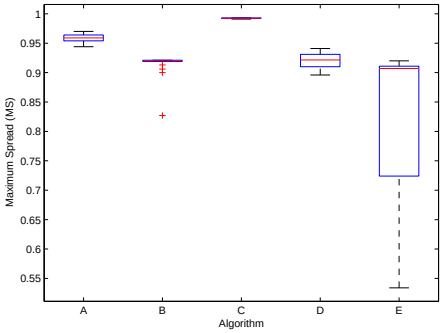


4.18.c: Hypervolume Ratio (HR)

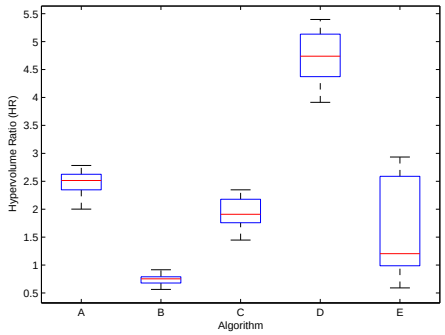
Figure 4.18: Generational Distance(GD), Maximum Spread(MS), and Hypervolume Ratio(HR) performance metrics for benchmark problem MF2. (A:NSGA-II, B:GS-MOMA, C:SS-MOMA-I, D:SS-MOMA-II, E:SS-MOMA-Perfect)



4.19.a: Generational Distance (GD)

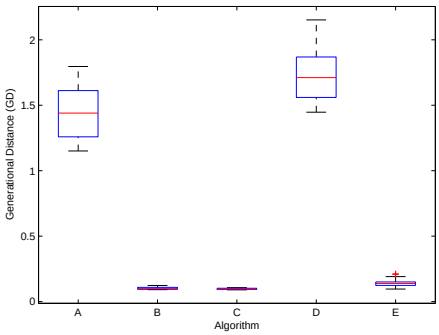


4.19.b: Maximum Spread (MS)

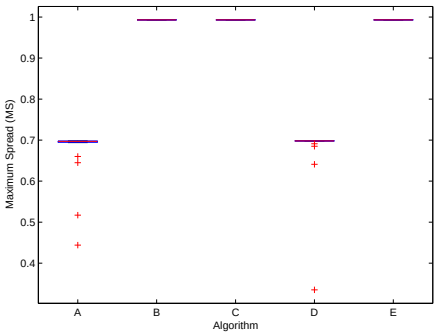


4.19.c: Hypervolume Ratio (HR)

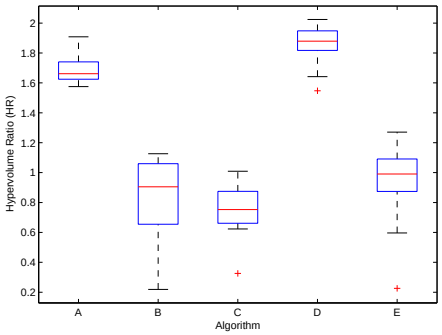
Figure 4.19: Generational Distance(GD), Maximum Spread(MS), and Hypervolume Ratio(HR) performance metrics for benchmark problem MF3. (A:NSGA-II, B:GS-MOMA, C:SS-MOMA-I, D:SS-MOMA-II, E:SS-MOMA-Perfect)



4.20.a: Generational Distance (GD)

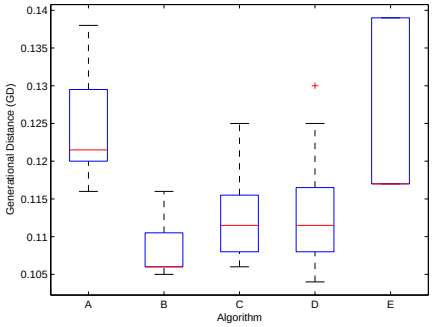


4.20.b: Maximum Spread (MS)

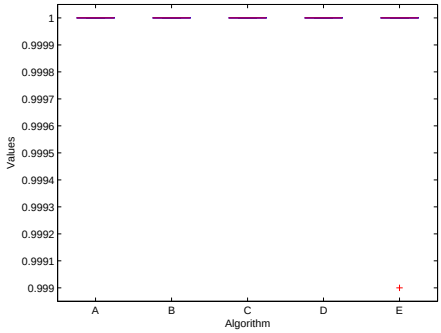


4.20.c: Hypervolume Ratio (HR)

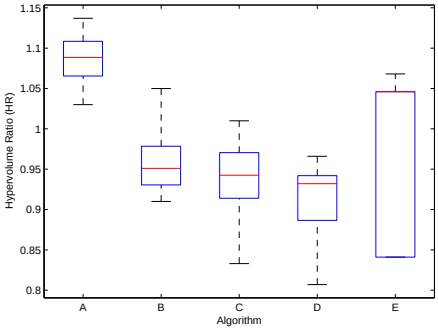
Figure 4.20: Generational Distance(GD), Maximum Spread(MS), and Hypervolume Ratio(HR) performance metrics for benchmark problem MF4. (A:NSGA-II, B:GS-MOMA, C:SS-MOMA-I, D:SS-MOMA-II, E:SS-MOMA-Perfect)



4.21.a: Generational Distance (GD)

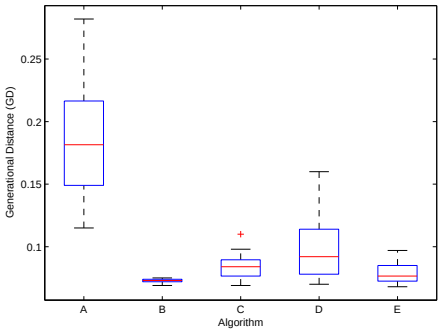


4.21.b: Maximum Spread (MS)

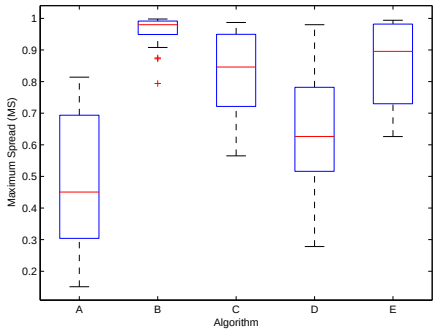


4.21.c: Hypervolume Ratio (HR)

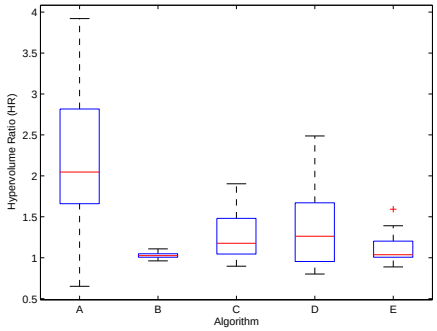
Figure 4.21: Generational Distance(GD), Maximum Spread(MS), and Hypervolume Ratio(HR) performance metrics for benchmark problem MF5. (A:NSGA-II, B:GS-MOMA, C:SS-MOMA-I, D:SS-MOMA-II, E:SS-MOMA-Perfect)



4.22.a: Generational Distance (GD)



4.22.b: Maximum Spread (MS)



4.22.c: Hypervolume Ratio (HR)

Figure 4.22: Generational Distance(GD), Maximum Spread(MS), and Hypervolume Ratio(HR) performance metrics for benchmark problem MF6. (A:NSGA-II, B:GS-MOMA, C:SS-MOMA-I, D:SS-MOMA-II, E:SS-MOMA-Perfect)

where:

- G_{db} : number of standard SO/MOEA search generations configured for building the database of training data points at the initial search phase of the DSM framework,
- G_{max} : maximum number of search generations,
- N_{pop} : population size,
- N_{obj} : number of objectives to optimize,
- k_{term} : number of iterations made in the trust-region-regulated local searches,
- T_f : Original/exact function evaluation cost,
- T_{ens} : time to build M_1 *i.e.*, the ensemble model,
- T_{PR} : time to build M_2, *i.e.*, the polynomial regression model, which might be not applicable if PR is already built when constructing M_1,
- $T_{overhead}$: other additional cost such as for fitness predictions and finding nearest points, which might be implementation-dependent.

While at a glance, there seems to be plenty of elements involved in the equation, it is worth noting that when considering optimization for computationally expensive problems, the most significant part contributing to the total computational effort required is T_f. Hence, when T_f is large, in fact T_{ens}, T_{PR}, and $T_{overhead}$ might become negligible.

On the other hand, the computational cost for SS-SOMA or SS-MOMA variants is:

$$T_{comp} = G_{db}N_{pop}N_{obj} + (G_{max} - G_{db})[N_{pop} \\ (T_m + T_f k_{term} N_{obj} + T_{overhead})] \quad \text{(Eq. 4.25)}$$

where T_m is the time taken to build the particular surrogate model used. While it is true that greater computational cost are spent per individual in GS-SOMA or GS-MOMA compared to its single surrogate counterpart in SS-SOMA or SS-MOMA, this extra cost appears to pay off as shown by the significant improvement achieved by the latter at the end of the optimization runs, as shown in empirical results in previous subsections.

4.4 Real-world Application: Drag/Lift Ratio of Aerodynamic Airfoil Design

Empirical results on benchmark problems in Section 4.3.1 and 4.3.2 have shown statistically that the proposed GSM framework performs more robustly across problems with different characteristics. In this section, the proposed GS-SOMA and GS-MOMA are applied to the moderately expensive airfoil problem, as described in Appendix C.1. In order to experiment with both SOO and MOO, the same problem has been used in both contexts by setting the objective as minimizing C_d/C_l in SOO; and minimizing C_d while maximizing C_l in MOO. The parameter settings for both SOO and MOO are detailed in Table 4.18 and 4.19. For SOO, the performance of GS-SOMA is compared with that of GA and SS-SOMA-RBF. Meanwhile, for MOO, NSGA-II, SS-MOMA-I, SS-MOMA-II, and GS-MOMA (refer to Table 4.16 for details) are used for comparison. Note that SS-SOMA-RBF is chosen as the representative of single surrogate algorithms since its performance has been shown to be generally acceptable throughout previous experiments.

At the end of the runs for SOO, we obtain the best fitness of 0.014378, 0.012679, and 0.011237 for GA, SS-SOMA-RBF, and GS-SOMA, respectively. In this experiment, it is shown that both SS-SOMA-RBF and GS-SOMA are superior to their non-surrogate GA counterpart. This confirms the effectiveness of surrogate-assisted evolutionary optimization. Between the two, however, GS-SOMA performs better as indicated by the final fitness values obtained. The corresponding airfoil shapes are plotted in Fig. 4.23.

Table 4.18: Setting of experiments for GA, SS-SOMA-RBF, and GS-SOMA.

Parameters Setting	
Population size (N_{pop})	100
Crossover probability (P_{cross})	0.9
Mutation probability (P_{mut})	0.1
Evolutionary operators	uniform crossover & mutation, elitism and ranking selection
Number of trust region iteration(k_{term}) for SS-SOMA and GS-SOMA	3
Database building phase (G_{db}) for SS-SOMA and GS-SOMA (in number of generations)	10

Table 4.19: Setting of experiments for NSGA-II, SS-MOMA-I, SS-MOMA-II, and GS-MOMA.

Parameters Setting	
Population size (N_{pop})	100
Crossover probability (P_{cross})	0.9
Mutation probability (P_{mut})	0.1
Evolutionary operators	simulated binary crossover, polynomial mutation, binary tournament selection, elitism, non-domination rank, and crowded distance
Number of trust region iteration(k_{term}) for SS-MOMA-I, SS-MOMA-II, and GS-MOMA	2
Database building phase (G_{db})	5

The same conclusion can be drawn for its MOO counterpart. In Fig. 4.24, it is shown that GS-MOMA is able to arrive at more converged and diverse solutions compared to NSGA-II and SS-SOMAs.

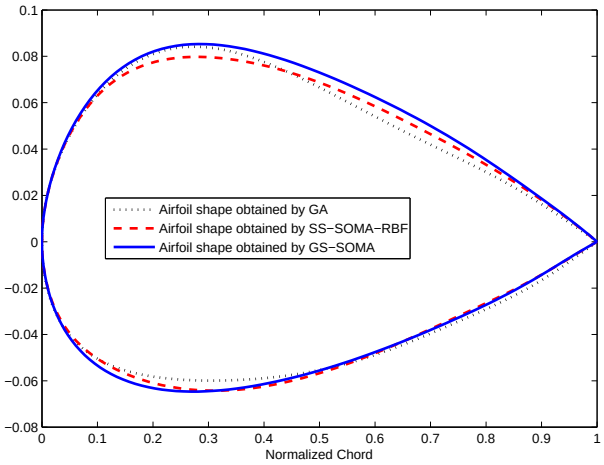


Figure 4.23: Airfoil shapes obtained by GA, SS-SOMA-RBF, and GS-SOMA, respectively.

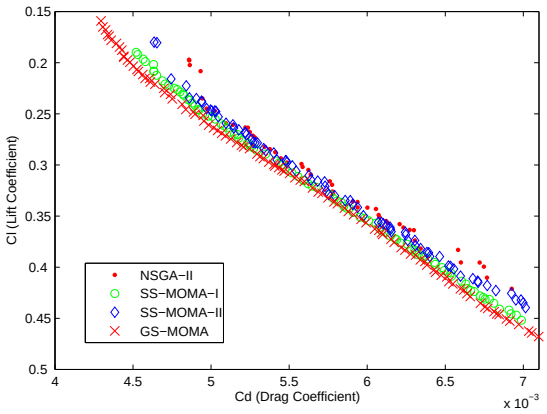


Figure 4.24: Pareto fronts obtained by NSGA-II, SS-SOMA-I, SS-SOMA-II, and GS-MOMA, respectively.

Chapter 5

Evolutionary Computation Using Multi-Scale Computational Models

In science and engineering, computational models which are created to simulate and analyze a set of processes or phenomena observed, often exist in different levels of accuracy and computational cost. For instance, in water cluster optimization, there are empirical models such as Ojamae-Shavitt-Singer 2 (OSS2) [135] or Thole-Type Model 2.1F (TTM2.1F) [30], besides the first principle quantum mechanical computation such as Density Functional Theory (DFT) [167]. A common presumption is that more accurate models, also regarded as having higher fidelity, are generally more costly. As has been discussed previously in Chapter 3, the complexity or level of details of a physical system can manifest in many forms, ranging from variable mathematical models, variable parametric formulations, variable operating conditions to variable residual tolerance levels. While the use of data-fitting surrogate models such as Gaussian Process, Polynomial Regression, Radial Basis Functions, is one possible solution to achieve greater efficiency in evolutionary optimization, the use of multi-scale computational models with varying computational cost, can serve as another potential alternative towards the same direction.

The motivation behind the use of multi-scale computational models is to benefit from the possible close correlation of the multiple models, as illustrated in Fig. 5.1. The

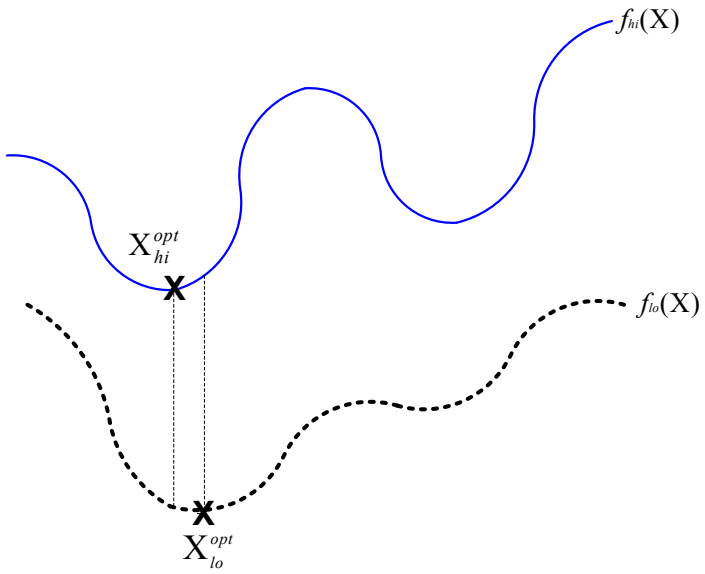


Figure 5.1: The correlation between low-fidelity ($f_{lo}(x)$) and high-fidelity ($f_{hi}(x)$) landscapes.

figure depicts two differing landscapes, possibly resulted from two computational models. While the nominal values of the low-fidelity landscape might be highly offset from those of its high-fidelity counterpart, the search on this low-fidelity landscape is still on track since the global optimum are retained closely to that of its high-fidelity counterpart, *i.e.* $\mathbf{x}_{hi}^{opt} \approx \mathbf{x}_{lo}^{opt}$. This implies that the correlation between the predicted fitness values is of utmost importance [89][68]. For instance, [130] also reported the high correlation between the energy landscape and the structures of water cluster isomers found on OSS2 and TTM2.1F models to those generated by the first principle quantum computations. However, we note that some mild assumption on the correlation between the different fidelity models should exist for any optimization algorithm to take advantage of their existence, otherwise such models should not be considered at all.

An instantiation of the generic SAEA framework tailored towards using multi-scale models, is outlined in Algorithm 11. In the pseudocode, l fidelity levels, F_i for $i \in \{1, 2, \ldots, l\}$ are assumed to exist, where $l > 1$ and F_l refers to the highest fidelity model. Note that similar to the generic SAEA framework described in Algorithm 6, the main idea

is to interleave the use of the highest fidelity model, F_l, and its lower fidelity counterparts, $F_1 - F_{l-1}$, with some pre-defined or adaptively controlled scheme. Two concrete applications of this type of SAEA are demonstrated in optimizing the water cluster structure and aerodynamic airfoil design in this chapter.

Algorithm 11 SAEA using multi-scale models

1: **Initialization:** Generate a database, containing a population of designs with their fitness values at different fidelity levels, F_i for $i \in \{1, 2, \ldots, l\}$.
2: **while** computational budget is not exhausted **do**
3: Generate offspring population using evolutionary operators (selection, crossover, mutation).
4: Evaluate offspring population employing F_1 to F_{l-1} lower fidelity models and the highest fidelity model F_l.
5: Archive all evaluations with their respective fidelity levels into database.
6: **if** Memetic Search **then**
7: Perform local refinement employing F_1 to F_{l-1} lower fidelity models and the highest fidelity model F_l.
8: Archive all evaluations with their respective fidelity levels into database.
9: Replace offspring with refined individuals.
10: **end if**
11: **end while**

5.1 Water Cluster Structural Optimization

In this section, research on discovering the globally minimum energy of pure water clusters, $((H_2O)_n)$ for the number of molecules, $n = 4-8$, is presented. In this work, two low fidelity models, OSS2 and TTM2.1F (see Appendix C.3), are employed in place of the more computationally expensive DFT model in a memetic optimization framework. Similar to most optimization works in science and engineering, the water cluster structural optimization also involves the use of models that are computationally exorbitant, *i.e.*, the DFT model in this case. This often limits their practicality to simulations involving only a small number of molecules. In what follows, the Molecular Memetic Algorithm (MMA) as an instantiation of Algorithm 11 is presented to optimize this problem.

5.1.1 Molecular Memetic Algorithm

Here, we discuss in detail the Molecular Memetic Algorithm used, which is in principle a steady-state memetic algorithm tailored towards molecular optimization. To begin with, the representation and evolutionary operators used are described as follows.

- **Representation**

  Search space, $\mathbf{\Omega}$ of this optimization problem is the set of water clusters, $(H_2O)_n$, where n is the number of water molecules in each cluster. Hence, each member $\mathbf{x} \in \mathbf{\Omega}$ is a vector of $9n$ real numbers[1] representing each cluster's coordinates in 3D space measured in Angstroms unit. This representation is illustrated in Fig. 5.2.

- **Crossover**

  Uniform molecular-level crossover is performed by swapping every molecule position between two clusters. Additional mechanisms are also included to avoid any invalid structure, see Algorithm 12 for the details.

- **Mutation.** To mutate an individual cluster, translation, rotation, and relocation are performed to the selected molecules. Molecular translation is achieved by adding a vector of three random real numbers, normalized and scaled to a random distance < 2.0 Angstroms, to the coordinates of each atom in the selected molecule. Molecular rotation is performed by rotating the entire selected molecule by an arbitrary degree around the axis formed by the oxygen atom and a randomly generated point in 3D space. On the other hand, relocation operator is designed to be highly disruptive, by relocating randomly selected molecules to random locations on the surface of water cluster. The three different operators used for mutating an individual water cluster are depicted in Fig. 5.3. The sequence of mutation performed

[1]Note that 9n real numbers are required for each cluster since every H_2O molecule has 3 atoms, and each atom is represented by 3 real numbers coordinates.

in our optimization is described in Algorithm 13. Note that the validity of the resultant individual is preserved.

- **Local Search**

  Separate local optimizers are used on the different fidelity computational models. In particular, Davidon-Fletcher-Powell (DFP) [184] and DFT at B3LYP/6-31+G* level are deployed for the low and high fidelity computational models, respectively.

Molecule 1: H: x, y, z O: x, y, z O: x, y, z	Molecule 2: H: x, y, z O: x, y, z O: x, y, z	...	Molecule *n*: H: x, y, z O: x, y, z O: x, y, z

Figure 5.2: Chromosome representation for a single individual water cluster, $(H_2O)_n$.

5.3.a: Translation 5.3.b: Rotation 5.3.c: Relocation

Figure 5.3: The mutation operators.

In MMA (refer to Algorithm 14), the search begins with generating and evaluating a population of molecule clusters. These molecule clusters then undergo the evolutionary operators to create the child individual, $\mathbf{x}_c$. $\mathbf{x}_c$ is first locally optimized in the low-fidelity search space to arrive at $\mathbf{x}_{lo}^{opt}$. The idea is to bring $\mathbf{x}_c$ to $\mathbf{x}_{lo}^{opt}$ with lower computational effort than using the high-fidelity DFT, and $\mathbf{x}_{lo}^{opt}$ is near to $\mathbf{x}_{hi}^{opt}$, where $\mathbf{x}_{hi}^{opt}$ represents the local optimum of DFT model. In this way, further refinement from $\mathbf{x}_{lo}^{opt}$ to $\mathbf{x}_{hi}^{opt}$ based on the expensive DFT will be reduced significantly.

Algorithm 12 Procedure : Molecular Crossover

```
Input: Parent 1 and Parent 2.
Output: Child.
Child = Parent 1.
for each molecule location, l_i, in the molecule cluster do
    if RandomNumber ∈ [0, 1.0] < 0.5 then
        Set l_i^child = l_i^parent 1
        if Child has invalid structure then
            Revert Child to latest valid structure.
            Set l_i^child = l_i^parent 2
            if Child has invalid structure then
                Revert Child to latest valid structure.
            end if
        end if
    else
        Set l_i^child = l_i^parent 2
        if Child has invalid structure then
            Revert Child to latest valid structure.
            Set l_i^child = l_i^parent 1
            if Child has invalid structure then
                Revert Child to latest valid structure.
            end if
        end if
    end if
end for
```

5.1.2 Empirical Study

In this section, empirical study is performed on the water cluster structural optimization problem as described in Appendix C.3. All experimental codes are written in C++ language while the high-fidelity model employs DFT energy computation from the Gaussian software[2]. Here, numerical comparisons of the MMA-OSS, MMA-TTM, and MMA without any lower fidelity models (MMA-0), are conducted. In MMA-0, all mutated individuals are directly optimized using the expensive high-fidelity DFT model. The common parameter settings of the schemes used in the present experimental study are summarized in Table 5.1. Our experience with the water problem indicated the impor-

[2]Gaussian is a commercial software package for computing energies and vibrational frequencies of molecular systems [4].

Algorithm 13 Procedure : Molecular Mutation

```
Input: Child.
Output: Mutated child.
while Number of trials < Maximum trials do
    Randomly select a molecule location, l_i to mutate.
    if RandomNumber ∈ [0, 1.0] < 0.5 then
        Perform mutation by applying translation and rotation on l_i.
    else
        Perform stronger mutation by applying relocation on l_i.
    end if

    if Child has invalid structure then
        Revert Child to latest valid structure.
    end if
end while
```

Algorithm 14 Molecular Memetic Algorithm (MMA)

```
initialization: Generate and evaluate a population, P of molecule clusters.
while computational budget is not exhausted do
    Randomly select two parents, x_p and x_m from the population.
    Perform crossover on x_p and x_m, resulting in the child, x_c.
    Perform mutation on x_c.
    Locally optimize x_c using the lower fidelity model, i.e., OSS2 or TTM2.1F, to arrive at
    x_lo^opt.
    Locally optimize x_lo^opt using expensive DFT model, to arrive at x_hi^opt.
    if x_hi^opt ∉ P then
        Perform Lamarckian replacement on a random individual in P by x_hi^opt.
    end if
end while
```

tance of high diversity in the search. This is the reason for the high mutation rate, *i.e.*, $P_{mut} = 1.0$ is used.

Table 5.1: Setting of experiments for MMA-OSS, MMA-TTM, and MMA-0

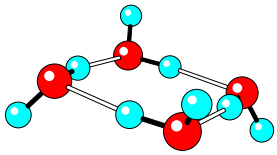
General Parameters	
Population size (N_{pop})	10
Maximum number of exact evaluations	100
Crossover probability (P_{cross})	1.0
Mutation probability (P_{mut})	1.0

In Table 5.2, the results obtained by the different schemes are summarized and compared with the currently known best water cluster structure obtained in [130]. Since, the

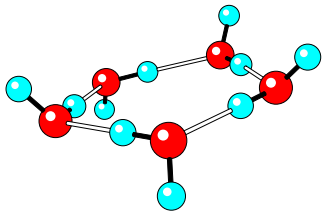
Table 5.2: Results obtained by MMA-OSS, MMA-TTM, MMA-0, and [130] in optimizing $(H_2O)_n$ for $n = 4 - 8$. Note that the bracketed numbers are the amount of high-fidelity evaluations when the search terminates.

	MMA-OSS	MMA-TTM	MMA-0	[130]
$n = 4$	-305.744467(51)	-305.744461(100)	-305.744458(100)	-305.744462
$n = 5$	-382.183785(48)	-382.183777(100)	-382.183770(100)	-382.183783
$n = 6$	-458.623623(92)	-458.623622(100)	-458.623622(100)	-458.623623
$n = 7$	-535.067171(100)	-535.067183(89)	-535.065462(100)	-535.067179
$n = 8$	-611.518540(100)	-611.518571(100)	-611.507184(100)	-611.518813

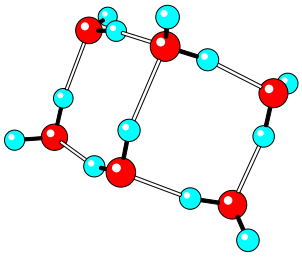
best structure available for this problem is known in advance, the search is set to terminate whenever the best solution found has reached the energy level of the best structure known or a maximum of 100 high-fidelity evaluations. From the results in Table 5.2, it is generally shown that MMA-OSS and MMA-TTM otperforms both MMA-0 and [130] on the smaller water cluster, *i.e.*, $n = 4 - 7$. Meanwhile in other cases which only terminate at the end of 100 iterations, it is shown that they have reached lower energy structure compared to MMA-0, with the precision of 10^{-4} ($n = 4 - 7$) or 10^{-3} ($n = 8$) to the best known structures. The optimized water clusters for $n = 4 - 8$ are depicted in Fig. 5.4.



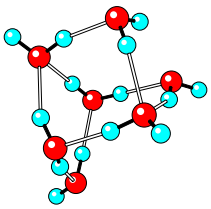
5.4.a: $(H_2O)_4$, obtained by MMA-OSS



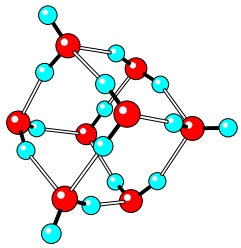
5.4.b: $(H_2O)_5$, obtained by MMA-OSS



5.4.c: $(H_2O)_6$, obtained by MMA-OSS



5.4.d: $(H_2O)_7$, obtained by MMA-TTM



5.4.e: $(H_2O)_8$, obtained by MMA-TTM

Figure 5.4: The obtained global minimum water clusters from our experiment on optimizing $(H_2O)n$ for $n = 4 - 8$ using MMA-OSS and MMA-TTM.

5.2 Aerodynamic Airfoil Design: An Inverse Pressure Problem

This section describes the investigation performed on multi-level evolutionary optimization on the aerodynamic airfoil design, particularly the inverse pressure problem[3]. The purpose of the current study is to demonstrate the adoption of a multi-level evolutionary optimization methodology on problems with only one computational model. In practice, the evaluation of a single individual design on this problem (see Appendix C.2) is an iterative process, usually terminated after a certain residual error threshold has been

[3]In the inverse design problem, the aim is to minimize the difference between the surface pressure P of a given airfoil with the desired pressure profile P_d of a baseline shape. If W is the flow variables and S the shape design variables, the inverse pressure design problem can be formulated as a minimization problem of the form:$f(W,S) = \frac{1}{2}\int_{wall}(P - P_d)^2\, d\sigma$. More details are provided in Appendix C.2.

reached. In the case of limited computational budget, it is also possible to have a maximum number of iterations as the effective termination condition. Hence, data-centric multi-scale models can be obtained from partial results at different levels of residual error tolerance and/or maximum iterations. Generally, lower residual error tolerance or greater number of iterations provides higher fidelity outputs.

The correlation factor (r) and root mean square error ($rmse$) of the models at variable levels of residual error tolerance (*i.e.*, fidelity) to the original high fidelity airfoil analysis model are reported in Figs. 5.5.a and 5.5.b, respectively. In this study, a set of design points, obtained from space-filling Design of Experiment (DOE) sampling, are partitioned into separate local clusters based on their proximity in the search space of the airfoil model. The correlation factor, r_k^j, and root mean square error, $rmse_k^j$ for data cluster j at fidelity level k, are defined as follows:

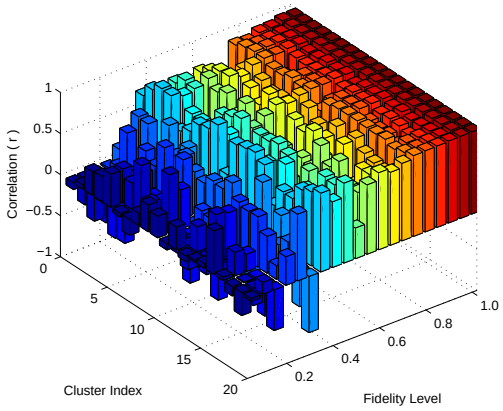
$$r_k^j = \frac{n\sum_{i=1}^{n} f_i\hat{f}_i - \sum_{i=1}^{n} f_i \sum_{i=1}^{n} \hat{f}_i}{\sqrt{[n\sum_{i=1}^{n} f_i^2 - (\sum_{i=1}^{n} f_i)^2][n\sum_{i=1}^{n} \hat{f}_i{}^2 - (\sum_{i=1}^{n} \hat{f}_i)^2]}}. \qquad \text{(Eq. 5.1)}$$

$$rmse_k^j = \sqrt{\frac{\sum_{i=1}^{n} |f_i - \hat{f}_i|^2}{n}} \qquad \text{(Eq. 5.2)}$$

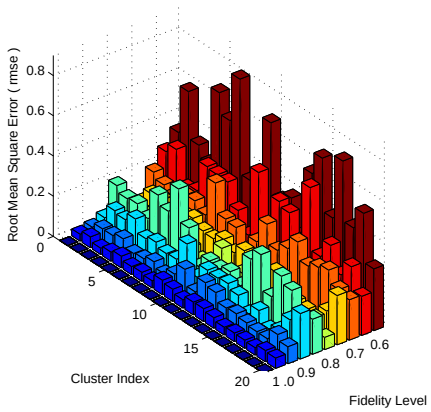
where n is the number of data in cluster k, f_i is the output from the original analysis code for data i, and $\hat{f}_i$ is the output from the fidelity level k of data i. The large $rmse$ values in Fig. 5.5.b highlighted large discrepancies between the low fidelity models and the original analysis code. On the other hand, Fig. 5.5.a displays good fitness landscape correlation between the low fidelity localized models and the original model at early stages of the convergence.

5.2.1 Memetic Algorithm with Dynamic Fidelity Computational Models

Based on the aforementioned analysis, a multi-level evolutionary memetic framework labeled here as Memetic Algorithm with Dynamic Fidelity Models (MA-DFM) is presented.



5.5.a: Correlation factor



5.5.b: Root mean square error ($rmse$)

Figure 5.5: (a) Correlation factor (r) and (b) root mean square error ($rmse$) of the 20 clustered localized models for variable levels of residual error tolerances, while taking the original airfoil analysis code as the reference model (*i.e.*, with a fidelity of 1)

The idea behind the MA-DFM is that the complexity level of the localized computational model to be used in place of the exact analysis code, is dynamically determined at run time as the search progresses. In the process, a user specific correlation factor, η, is utilized to determine the minimum level of fidelity that is considered to match well with the original model. The pseudo codes of the MA-DFM used in the present work is outlined in 15.

The MA-DFM (refer to Algorithm 15) begins with the initialization of a population of design points. During the database building phase, the search operates as a traditional evolutionary algorithm for the initial few generations. In this stage, only the original analysis code is used as the fitness function while at the same time, all partial results encountered along the evaluation of the computational model are archived in database, $\Im$. Subsequently, the algorithm proceeds to the memetic search phase. Henceforth, for each individual $\mathbf{x}$, an appropriate fidelity level, φ, of the computational model to use in the local learning phase may then be determined based on the m nearest points in database $\Im$. For instance, a user-specified confidence based on correlation, η, may be introduced

to obtain the minimum fidelity model to be used in place of the original model in the local search. To obtain this minimum fidelity model, we step up the correlation measure starting from the low to high fidelity data found in database $\Im$, till the fidelity level φ, where $r_\varphi \geq \eta$, is found. Upon local convergence based on the low fidelity model, the locally optimized solution x_{opt} is then validated using the original model and replaces the original starting individual if improved solution is attained, in the spirit of Lamarckian Learning (refer to line 10 of Algorithm 15). The entire search cycle is then repeated until the maximum allowable computational budget is exhausted.

Algorithm 15 MA with Dynamic Fidelity Models (MA-DFM)

1: **Initialization:** Generate a database, $\Im$ containing a population of designs, archive all evaluations made into the database.
2: **while** computational budget is not exhausted **do**
3: **if** database building phase **then**
4: Evolve the population using standard EA, archive all evaluations into the database.
5: **else**
6: **for** each individual $\mathbf{x}$ in the EA population **do**
7: Find m nearest points to $\mathbf{x}$ in database.
8: Based on the m points, find the minimum fidelity level φ, at which the correlation measure between fidelity level of φ and the original analysis code, has reached η, *i.e.*, $r_\varphi \geq \eta$
9: Apply local search on $\mathbf{x}$ using fitness function at fidelity level φ, $f_\varphi(.)$ with probability of P_{ls} and intensity of I_{ls} to find an improved solution, $\mathbf{x}_{opt}$.
10: Replace $\mathbf{x}$ with the locally improved solution, *i.e.*, $\mathbf{x} = \mathbf{x}_{opt}$ and $f(\mathbf{x}) = f(\mathbf{x}_{opt})$.
11: Archive all new function evaluations into database $\Im$.
12: **end for**
13: Apply standard evolutionary operators to create a new population.
14: **end if**
15: **end while**

It is worth highlighting on the novel use of dynamic localized computational model in the proposed framework. Particularly, local models are used in favor of global models since constructing accurate global models is fundamentally flawed due to the curse of dimensionality [142]. Further, this allows more precise estimation on the unique characteristics of the problem landscapes, thus leading to the prediction on the appropriate

level of localized model fidelity over the use of the original computationally expensive model.

5.2.2 Empirical Study

In this section, an empirical study is presented on optimizing the inverse pressure problem as described in Appendix C.2. In the study, the efficacy of the proposed MA-DFM is validated against the standard GA, MA, and a non-adaptive MA that employs a fixed fidelity model or MA-FFM in short. In contrast to MA-DFM, note that MA-FFM performs local searches having a pre-defined fixed model of low fidelity (f_ζ), where $\zeta < 1$. The common parametric configurations of all 4 schemes used in the present experimental study are summarized in Table 5.3 and briefly described in what follows.

Table 5.3: Setting of experiments for GA, MA, MA-FFM, and MA-DFM.

General Parameters	
Population size (N_{pop})	100
Crossover probability (P_{cross})	0.9
Mutation probability (P_{mut})	0.1
Maximum number of exact evaluations	5000
MA, MA-FFM, and MA-DFM - specific Parameters	
Local search probability (P_{ls})	0.2
Local search intensity (I_{ls})	10 local search iterations
MA-FFM and MA-DFM - specific Parameters	
Database building phase	1000 evaluations
Number of nearest neighbours (m)	(d+1)(d+2)/2
MA-FFM - specific Parameters	
Fixed lower fidelity level used (ζ)	0.5
MA-DFM - specific Parameters	
Minimum correlation required (η)	0.75

In the local search procedure, the well-established Feasible Sequential Quadratic Programming (FSQP) method is employed. Although I_{ls} defines the maximum computational budget of each individual in the local learning phase, the actual time incurred do vary according to the fidelity level used. In the present study, the computation cost per evaluation is determined by the fidelity level of the model used, *i.e.*, evaluating a computational model with a fidelity level of 0.7 translates to a compute cost of 0.7 evaluation count. On the other hand, note that each evaluation of the fixed low fidelity in the local

search for MA-FFM is performed at a constant cost of ζ, which is here assumed as 0.5 in our experiment setting, which is half the computational expense of the original model. Since model accuracy is highly dependent on the sufficiency of the m data points used for model building, the size of nearest neighboring points used is defined by $(d+1)(d+2)/2$, where d is the dimensionality of the optimization problem and is 24 for the airfoil problem considered here. A maximum computational budget of 5000 function evaluations is used in the experimental study.

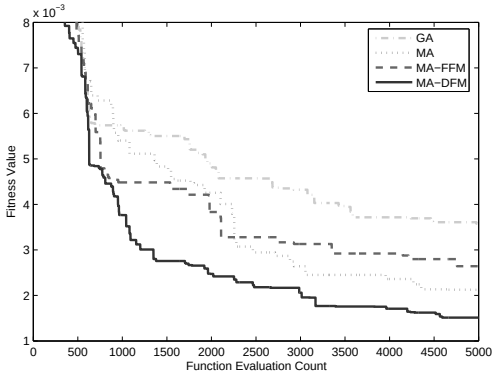


Figure 5.6: The convergence of fitness value (the pressure difference) obtained by the four schemes compared, *i.e.*, GA, MA, MA-FFM, and MA-DFM.

The obtained average convergence trends for each of the 4 algorithms, across 10 independent runs, are summarized in Fig. 5.6. It is also worth noting that in all the algorithms considered, the first 1000 exact evaluations represent the results of the standard GA, hence similar initial search trends are shown in the figure. Nevertheless, the search trends begin to differ after 1000 evaluations where it can be observed that all the memetic schemes (MA, MA-FFM, and MA-DFM) studied outperform the standard GA significantly. This demonstrates the ability of the MAs in converging to good quality solution more efficiently than a standard GA [137, 140].

Next, note that among the two multi-fidelity MA schemes, while both MA-FFM and MA-DFM employs lower fidelity model as the fitness function, the latter is observed to

perform better. A $t-test$ with 95% confidence level for significance of the mean differences on the results based on statistical data from Table 5.4 and Fig. 5.7, is presented in Table 5.5 to confirm the significant difference in performance. Subsequently, the best airfoil shapes and their respective pressure profiles obtained by 4 schemes compared in the experiments are plotted in Figs. 5.8 and Figs. 5.9. It is shown from these figures that the best shape and subsequently the best pressure profile obtained by MA-DFM is closer to the shape of the NACA0015 baseline airfoil. The results obtained on the real-world aerodynamic problem thus demonstrate the ability of the MA-DFM in model fidelity control that decides, at runtime, the appropriate fidelity level of the computational model to be used in place of the exact analysis code as the search progresses.

Table 5.4: The mean and standard deviation of results obtained by GA, MA, MA-FFM, and MA-DFM at the end of 5000 function evaluations.

Scheme	**Mean ± Std. deviation**
GA	3.5776e-3 ± 9.5973e-4
MA	2.1232e-3 ± 4.9329e-4
MA-FFM	2.6417e-3 ± 5.0709e-4
MA-DFM	1.5131e-3 ± 4.0862e-4

Table 5.5: Result of *t-test* at 95% confidence level for GA, MA, MA-FFM, and MA-DFM search ($s+$, $s-$, or $\approx$ at row i and column j indicates that the results of optimization scheme at row i are significantly better, significantly worse, or indifferent compared to those at column j).

Scheme	**GA**	**MA**	**MA-FFM**	**MA-DFM**
GA	$\approx$	$s-$	$s-$	$s-$
MA	$s+$	$\approx$	$s+$	$s-$
MA-FFM	$s+$	$s-$	$\approx$	$s-$
MA-DFM	$s+$	$s+$	$s+$	$\approx$

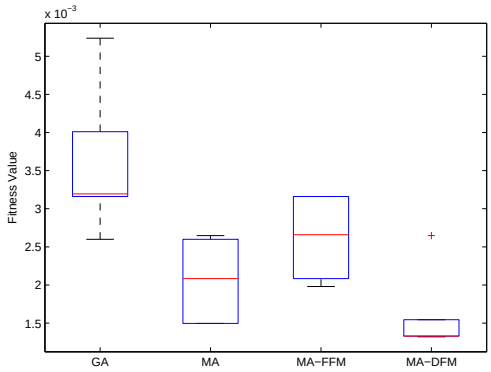
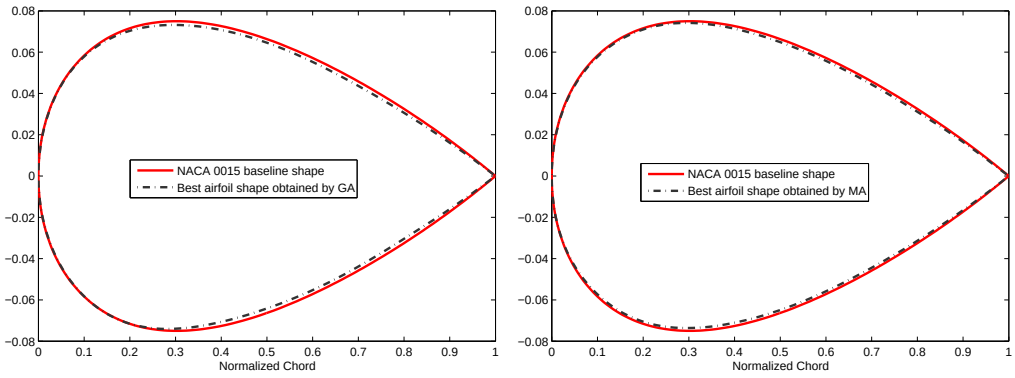
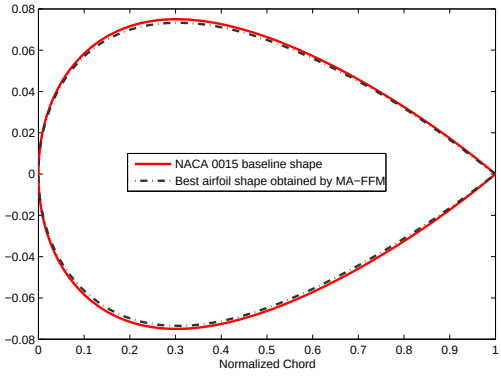


Figure 5.7: The boxplot of fitness values obtained by GA, MA, MA-FFM, and MA-DFM at the end of 5000 function evaluations.

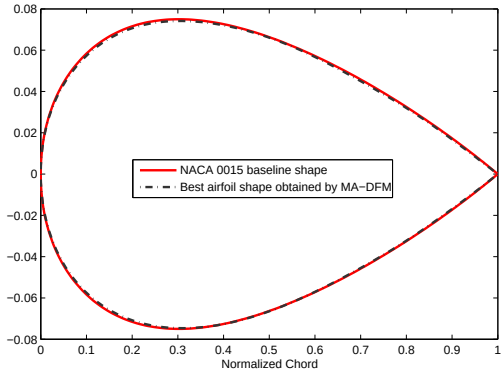


5.8.a: GA

5.8.b: MA

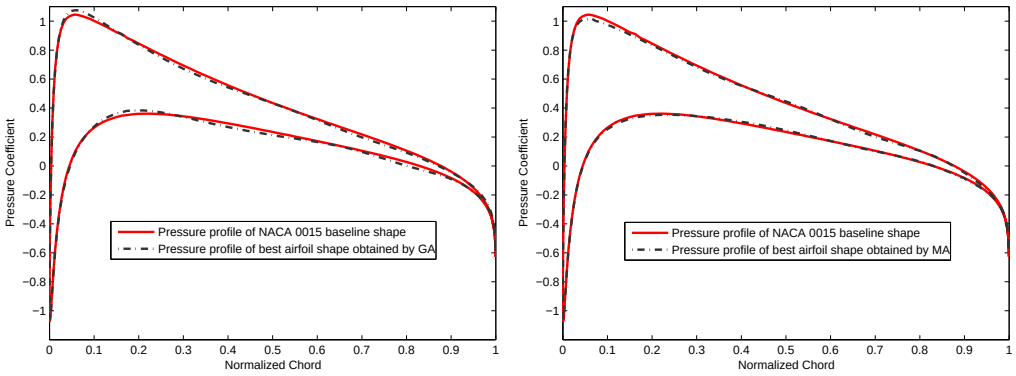


5.8.c: MA-FFM



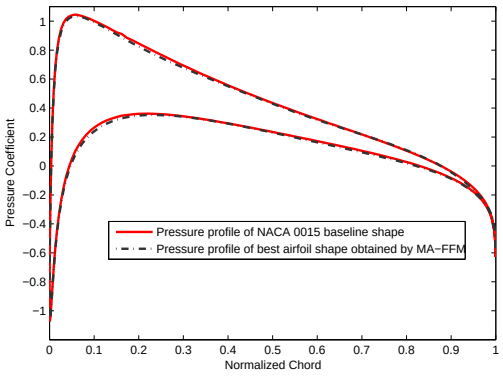
5.8.d: MA-DFM

Figure 5.8: Comparison between best airfoil shapes obtained by the four schemes, *i.e.*, GA, MA, MA-FFM, and MA-DFM.

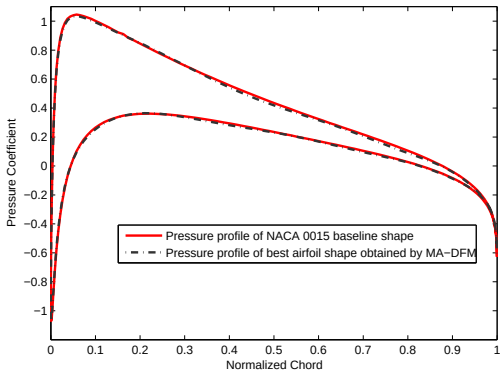


5.9.a: GA

5.9.b: MA



5.9.c: MA-FFM



5.9.d: MA-DFM

Figure 5.9: Comparison between the pressure profiles at the upper and lower surfaces of the best airfoil shapes obtained by the four schemes, *i.e.*, GA, MA, MA-FFM, and MA-DFM.

Chapter 6

Parallel and Distributed Evolutionary Computation Using Grid Computing

While surrogate/approximation techniques have been shown to be beneficial for handling computationally expensive evolutionary optimization, another intuitive way for coping with the same issue is parallelization via parallel and distributed computing. Although various forms of such technology have been exploited for quite a while, they remain to focus on small-scale dedicated computing resources and are not easily extendable towards harnessing computing resources that span across laboratories or even organizations at disparate geographical locations. In a business framework, the sharing of resources at organizational level regardless of network border can have pertinent impacts on return of assets and investments. It is for these reasons that Grid computing [54][21][56][55] has gained widespread attention, as it sets about the notion of establishing a set of open standards for distributed resources and ubiquitous services, for representation, discovery and control. This chapter presents the research work on parallelizing evolutionary optimization under a heterogeneous Grid computing environment to mitigate the issues relating to the intractable design life cycle of computationally expensive problems. In particular, a grid-enabled Parallel EA (GE-PEA) framework is proposed. Since the additional computational cost imposed by surrogate model building in SAEAs is generally

negligible, the proposed parallelization framework should be relevant for SAEAs. Hence, we can safely isolate the discussion in this chapter only on the parallelization aspect of EAs. An interesting demonstration of combining parallelization and surrogate models on accelerating evolutionary optimization of computationally expensive problems is reported in a recent publication by Liakopoulus *et al.* [109].

6.1 Grid Enabled Parallel Evolutionary Algorithms

In this section, a Grid-enabled architecture for Parallel EAs, or GE-PEA in short, is presented. Various Grid enabling technologies have been considered in developing GE-PEA and these are discussed in Section 6.1.1. The detailed workflow of GE-PEA is then discussed in Section 6.1.2.

6.1.1 Grid Enabling Technology

Here, some of the key Grid technologies used in developing the GE-PEA are discussed. Globus Toolkit[1] [53], Commodity Grid Kit (CogKit)[2] [3], Ganglia monitoring tool[3] [120], and NetSolve[4] [11] are some of the core Grid technologies used in developing the GE-PEA. From a survey of the literature, it is possible to establish that the existing GridRPC standard lacks mechanisms for automatic resource discovery and selection of Grid resources. As a result, users are naturally required to perform manual look-up and select resources when assigning new tasks onto the Grid computing resources. This is clearly impractical since a large amount of computing resources exist on the Grid and are often dynamic

[1] **Globus** is the de-facto Grid Middleware, which provides the Grid infrastructures for security, data management, resource management, and information service.

[2] **CogKit** is the API for Globus.

[3] **Ganglia** is a distributed monitoring system for high-performance computing systems such as clusters and Grids, which has been in deployed on over 500 clusters in the world.

[4] **NetSolve** is a Grid tool, based on the agent-client-server model that enables the clients to access any services provided by servers registered to an agent.

in practice. Furthermore, most optimization problems have a large number of evaluation tasks that require many identical computations of different analysis parameter sets represented in the form of chromosomes. It is therefore extremely inefficient if the interactions between the client (master) and resources (slaves) are repeated many times for the same remote procedure call. The uniqueness of our GE-PEA framework is therefore an extended GridRPC API with the inclusion of a meta-scheduler.

The meta-scheduler performs discovery, bundling and load balancing using online information gathered from the computing clusters and Grid services[5] that exist on the Grid. In our GE-PEA framework, the Globus Monitoring and Discovery Service (MDS) [5] and Ganglia [120] have been used to provide the online information. The MDS maintains a database of available resource information and acts as a centralized directory service keeping track of the resources, their locations on the Grid and how they may be consumed. Ganglia, on the other hand, monitors and provides workload information about the available clusters, computing nodes and Grid services.

In any Grid computing setup, it is necessary to first enable the software components as Grid services so that they can live in a Grid environment. Hence, our GE-PEA is equipped with an extended GridRPC API [128][77] based on the Commodity Grid Kit (CoGKit) for 'gridifying' new or existing analysis/simulation codes or objective/fitness function as Grid services. This choice is down to its simplicity in implementations and its ability to offer high-level abstraction, thus concealing the high-level of complexity of Grid computing environments from the end users. For the sake of brevity, the implementation details of our extended GridRPC API is not described here but refer the readers to [77] for further exposition. Other Grid technologies utilized in the present work include the Globus Grid Security Infrastructure (GSI) [196] for secure and authenticated access over

[5]Here, 'Grid service' refers to any shared software components that is wrapped to live on the Grid environment. In the context of optimization in science and engineering, the Grid services are wrapped forms of analysis/simulation codes or objective/fitness functions.

the Grid. For data flow, the Globus Grid File Transfer Protocol (GridFTP) [154] is used for conducting all forms of data transfer.

6.1.2 Workflow

In this subsection, the workflow of the GE-PEA framework is described. In the master node of each cluster, the 'subpopulation evaluation' is enabled as a Grid service for remote invocation using the Globus job submission protocol. This service, in turn, will direct the incoming evaluation calls to local cluster scheduler, such as NetSolve [11], Sun Grid Engine [59], Condor [57] or otherwise. In the GE-PEA, the 'fitness function evaluator' is wrapped as a NetSolve service on each compute node, and waiting to be consumed by the 'subpopulation evaluation' service at the respective cluster's master node.

The detailed workflow of the GE-PEA can be outlined as nine crucial stages and is depicted in Figure 6.1. These stages are described as follows:

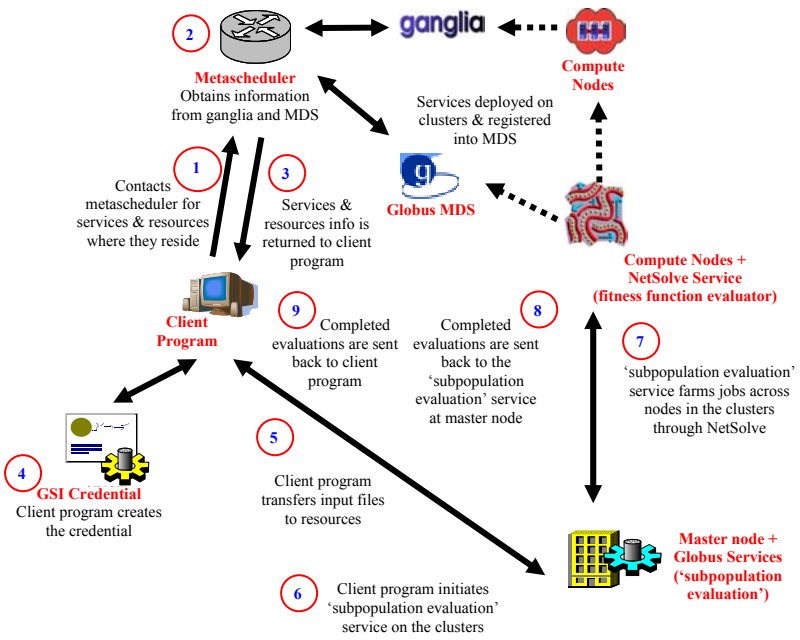


Figure 6.1: Workflow of GE-PEA framework.

(i) Prior to the start of the evolutionary search, the GE-PEA master program contacts the meta-scheduler to request for suitable computing resources that provide 'subpopulation evaluation' Grid service.

(ii) The meta-scheduler then obtains a list of available resources together with their status of availability. Such status information is acquired from the Globus MDS and Ganglia.

(iii) Grid computing resource and service information maintained by the Globus MDS and Ganglia are then provided to the GE-PEA master program to proceed with parallel evolutionary search.

(iv) To access Grid resources, Grid Security Infrastructure (GSI) credentials are subsequently generated. This forms the authentication or authority for consuming any form of resources living in the Grid environment.

(v) Represented in the form of ASCII or XML data files, the GE-PEA individuals/subpopulations are then transferred onto the identified remote computing clusters that offer the required 'subpopulation evaluation' services.

(vi) Parallel evaluation of the multiple individuals/subpopulations then commences at the selected remote computing clusters using Globus job submission protocol. Whenever the cluster receives a request to launch the 'subpopulation evaluation' service, an instance of this service gets instantiated.

(vii) The 'subpopulation evaluation' requests are subsequently farmed across the field of processing nodes available in the cluster by invoking 'fitness function evaluator' services through NetSolve.

(viii) Once the service request is completed, the obtained fitness of individual/subpopulation is marshalled back to the calling service at the master node.

(ix) The obtained fitness values are then delivered back to the GE-PEA master program to proceed with the evolutionary operations. This process repeats until the search termination criteria are met.

6.2 Theoretical Speed-up Analysis

One of the major performance issues when running a parallel algorithm is how much speed-up they can offer compared to a sequential run of the same algorithm [156]. This speed-up (S) measurement can be defined by:

$$S = \frac{T_s}{T_p} \qquad \text{(Eq. 6.1)}$$

where T_s and T_p denote the execution time when the algorithm is executed in serial and parallel, respectively. The whole computation of a parallelizable algorithm can be divided into three major parts, *i.e.*, sequential computation (λ), parallelizable computation (γ), and parallelization overheads (O). In many cases, the communication overhead incurred represents the most dominating parallelization overheads. For a problem of size n and number of processors p, it is possible to derive from equation (Eq. 6.1) that

$$S(n,p) = \frac{\lambda(n) + \gamma(n)}{\lambda(n) + \frac{\gamma(n)}{p} + O(n,p)} \qquad \text{(Eq. 6.2)}$$

which provides an upper bound for the maximum speed-up achievable by a parallel computer having p processors when computation is divided equally among the processors. Further simplification to the upper bound of maximum speed-up may be obtained by using *Amdahl's Law* [16] as shown in equation (Eq. 6.2) becomes:

$$S(n,p) = \frac{\lambda(n) + \gamma(n)}{\lambda(n) + \frac{\gamma(n)}{p} + O(n,p)} \leq S^{max}(n,p) = \frac{\lambda(n) + \gamma(n)}{\lambda(n) + \frac{\gamma(n)}{p}} \qquad \text{(Eq. 6.3)}$$

This provides a theoretical bound on the maximum speed-up that can be achieved by the parallel algorithm.

Note that while the developed GE-PEA framework is capable to cope with both synchronous and asynchronous population replacement schemes, the focus is more on the synchronous version[6]. The main concern when dealing with synchronous PEA is due to the fact that the search can proceed to the next generation only when all individuals in current generation have been evaluated. Considering computationally expensive fitness functions, problems arise when some individuals require greater time to evaluate compared to others, which might happen due to difference in computational power of resources and/or variable complexity of fitness evaluators. By maintaining the flow of an algorithm as in its sequential version, it follows naturally that no algorithmic behavior changes should occur[7].

The simplest parallel architecture is to maintain the algorithmic flow at the main program and only parallelize the fitness evaluations. Having such architecture, distribution of fitness evaluations can be performed in different manners. The most naive way would be to distribute them based on a one-to-one mapping between individuals and processors. However by doing so, one of the most essential aspects in a Grid environment, *i.e.*, the heterogeneity in computational power of each resource, has been ignored blindly. In [129], it is recommended to utilize benchmarking information and/or past historical computational effort to compute the proper distribution of individuals into each cluster, *i.e.*,:

$$R_i = \frac{C_i}{\sum_{j=1}^{M} C_j} \text{ or } R_i = \frac{t_i^{-1}}{\sum_{j=1}^{M} t_j^{-1}} \qquad \text{(Eq. 6.4)}$$

[6]In synchronous replacement, the search can only proceed to the next generation when all individuals in the current generation have been evaluated and hence can be replaced entirely. Meanwhile in its asynchronous counterpart, the search is allowed to continuously replace the parental population as soon as any new evaluated results become available. Due to such appealing property, asynchronous replacement is often considered as a suitable candidate for EA parallelization in a Grid environment. However, it is worth noting that [203][170] reported that modifications to the original asynchronous GA replacement are necessary to ensure consistent performance.

[7]It was reported in [65] that heterogeneity in the computational power between compute nodes used to evolve multiple subpopulations of an asynchronous parallel island GA can bring about premature convergence to the false global optimum. The reason behind such phenomenon is that individuals from faster nodes will easily overwhelm those at the slower nodes via migration.

where $R_i \in [0,1]$ is the ratio of individuals to be distributed to cluster i, C_i is the computational power benchmarked at cluster i, while t_i refers to the past historical computational effort required to compute a population of certain size.

Hence, considering the use of proper distribution ratio, the sequential execution time, T_s, and parallel execution time, T_p, can be derived as:

$$T_s = \sum_{i=1}^{G} \sum_{j=1}^{N} F_{ij} \qquad \text{(Eq. 6.5)}$$

$$T_p = \sum_{i=1}^{G} \left(MO_{inter} + \max_{j=1,\ldots,M} \left[\alpha F_{ij} + N_{ij} O_{intra} \right] \right) \qquad \text{(Eq. 6.6)}$$

where:

G	:	Number of maximum generation count.
N	:	Population size in T_s or subpopulation size in T_p
C	:	Number of individuals in a subpopulation.
M	:	Number of clusters used.
P	:	Number of processors used.
α	:	Parallelism factor of a cluster, which is a function of the subpopulation size and number of processors.
O_{intra}	:	Intra-cluster communication overhead to parallelize a chromosome within a cluster.
O_{inter}	:	Inter-cluster communication overhead to parallelize a chromosome within a cluster.
F	:	Function cost or time to complete a single fitness evaluation.

For the GE-PEA to provide any speed-up, the following must apply:

$$T_p < T_s \qquad \text{(Eq. 6.7)}$$

$$\Longleftrightarrow G\left(MO_{inter}^{max} + \alpha^{max} F^{max} + N_j^{max} O_{intra}^{max}\right) < GNF^{min} \qquad \text{(Eq. 6.8)}$$

To simplify further, it is noted that O_{intra} is often negligible and hence:

$$\Longleftrightarrow MO_{inter}^{max} + \alpha^{max} F^{max} < NF^{min} \qquad \text{(Eq. 6.9)}$$

From Eq. 6.9, it is obvious that the efficiency of GE-PEA is influenced by several factors, *i.e.*, total population size, N, the number of clusters involved, M, the function cost, F, the parallelism factor, α, and inter cluster communication overhead, O_{inter}. Note that in the setting of multi-cluster Grid computing environment, where computing cluster may be geographically separated across laboratory or even country boundaries, O_{inter}^{max} cannot be simply ignored if it is comparable to F. Otherwise, when F is significantly larger than O_{inter}, Eq. 6.9 can be simplified further to:

$$\Longleftrightarrow F^{max} < \frac{NF^{min}}{\alpha^{max}} \qquad \text{(Eq. 6.10)}$$

i.e., the slowest evaluation, F^{max}, usually prevails at the slowest processor, is upper-bounded by $\frac{NF^{min}}{\alpha^{max}}$.

6.3 Empirical Study

In this section, an empirical study on GE-PEA for distributed and heterogeneous Grid computing environments is presented. All experimental codes are produced in C++ codes. In particular, we present empirical results obtained prior and after the final design of GE-PEA. For readability purpose, we refer to the prior GE-PEA design as GE-PEA-0, and the final design remains as GE-PEA. Note that our aim is to justify some design issues on GE-PEA, which materialize in the difference between GE-PEA-0 and GE-PEA, *i.e.*, *1*) GE-PEA-0 couples evolutionary operators in the service called 'subpopulation evolution' at master node of each cluster, and *2*) GE-PEA-0 equally assigns the subpopulation size.

We begin the first experiment using only GE-PEA-0 to show how such design may be unsuitable for a heterogeneous Grid environment. Using 24 design variables and

the cost function in Appendix C.2, the GE-PEA-0 is applied for the optimization of the subsonic inverse pressure design problem. The control parameter are configured as follows: population size for every subpopulation is 80, crossover probability of 0.9, mutation probability of 0.1, migration period of 10 generations interval, linear fitness scaling, elitism, and termination upon maximum number of generation 100.

Table 6.1: Summary of the Grid environment considered for optimizing the airfoil design problem.

Cluster	Number of CPU(s)	CPU Type	Memory
pdpm	8	8 × Xeon 2.6GHz	4GB
cemnet	5	4 × Xeon 2.4GHz 1 × PIV 2.66GHz	2GB
surya	7	7 × PIII 733MHz	3.2GB
ec − pdccm	8	8 × PIII 650MHz	2.2GB

Here, a Grid environment consisting of the four clusters summarized in Table 6.1, is considered. The average wall-clock time of each computing cluster for evaluating a subpopulation of 50 GA chromosomes in parallel are given in Table 6.2. Note the significant difference in the computational efforts required by these heterogeneous clusters. The search performances of the n-cluster GE-PEA-0, *i.e.*, 2, 3, or 4 subpopulations GE-PEA-0, for optimizing the inverse pressure design problem is then reported in Figure 6.2. To obtain a more conservative result, whenever the single cluster setting is considered, the fastest cluster will be used. Due to the heterogeneity of the Grid environment considered, the slowest *surya* cluster has become the bottleneck of the GE-PEA-0 since it uses a synchronous migration model which waits for all the 'subpopulation evolution' services in a generation to complete before the migration operation and subsequent search generations may proceed. This results in the poorer search efficiency of the multi-cluster environment, which is significantly shown in the 4-cluster GE-PEA-0.

The results obtained can be easily explained as follows. Consider the 4-subpopulation GE-PEA-0 run on 4 clusters, it can be estimated from Table 6.2 that the required

computational effort is significantly higher than in a single cluster GE-PEA-0, *i.e.*, $852.34 > 4 \times 164.13 = 656.52$. Note that this calculation is used as a simpler estimation to Eq. 6.9, since some parameter values are not precisely known in order to apply the equation directly.

Table 6.2: Computational efforts required to evaluate a subpopulation of 50 designs using the moderate-fidelity airfoil analysis code (inclusive of the communication overhead incurred).

Cluster	Average wall clock time (of 10 independent runs) for evaluating 50 individuals
pdpm	164.13 s
cemnet	366.48 s
surya	465.79 s
ec − pdccm	852.34 s

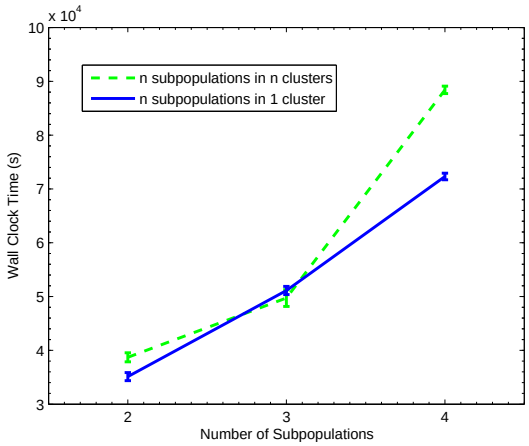


Figure 6.2: Average wall clock time of GE-PEA-0 in single and multiple clusters on the airfoil design problem.

To compromise with the heterogeneity of the computing resources, a simple solution to maintain the benefit of parallelization is to configure the subpopulation size according to the computational capabilities of the clusters. Nevertheless, such an approach has the disadvantage of possibly altering the standard behavior of a synchronous island PEA due to improper design of coupling the evolutionary operators into the 'subpopulation evolution' service at each cluster. Since the search behavior could be unpredictable in such

approach, it is advisable to be more conservative by maintaining uniform subpopulation sizes. A preferable solution should still provide the speed-up regardless of the heterogeneity in the Grid environment while preserving the standard behavior of the parallel genetic search. To achieve such a solution, we arrive at the final design of GE-PEA.

The core idea lies in: *1*) the decoupling of evolutionary operators from the 'subpopulation evolution' Grid service as in GE-PEA-0, *2*) well-balanced execution time obtained by non-uniform distribution of individuals into each cluster based on computational power. In this way, the computing clusters are solely meant for fitness evaluation purpose and all evolutionary operations proceed at the client side. By doing so, the uniformity of the subpopulation size can be maintained at the client side while the computing clusters are actually allocated non-uniform ensembles of individuals for fitness evaluations. This is slightly different from the original GE-PEA-0 where both evolutionary operations and fitness evaluations are coupled in the 'subpopulation evolution' service at each computing cluster, resulting in inflexibility of the design.

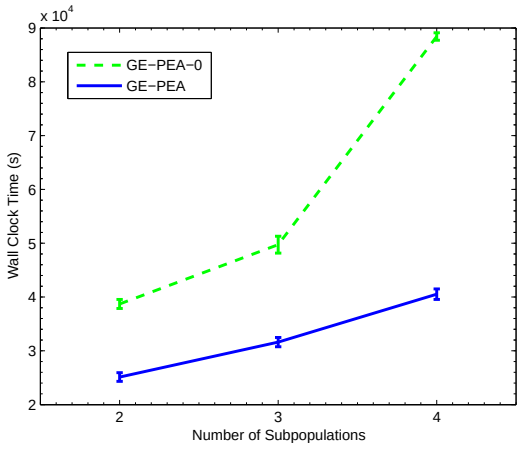


Figure 6.3: Average wall clock time of GE-PEA-0 and GE-PEA in 2, 3, and 4-subpopulation runs in optimizing the aerodynamic airfoil design problem.

To minimize the idling time of processing nodes in fast clusters while waiting for synchronization in the GE-PEA, we hope to obtain a well-balanced execution time in

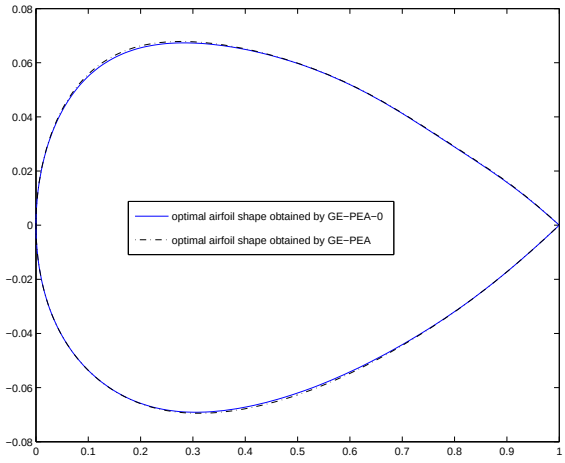


Figure 6.4: Optimal airfoil shape obtained after 100 generations of the 4-subpopulation GE-PEA-0 and GE-PEA.

each cluster and for each search generation. In each n-subpopulation GE-PEA search generation, each non-uniform ensemble of individuals, C_i, assigned to the i-th cluster for evaluations is estimated by:

$$C_i = \frac{t_i^{-1}}{\sum_{j=1}^{n} t_j^{-1}} \times n \times C \qquad \text{(Eq. 6.11)}$$

where C is the subpopulation size, n is the number of subpopulations(or clusters), and t_i is the time required by the i-th cluster to evolve a subpopulation in one generation. Note that, here such distribution strategy has been adopted since the past historical data on computational effort is available. Otherwise, one could resort to using computational power benchmarking information such as reported in [129].

The average wall-clock time and optimum shape obtained when using GE-PEA-0 and GE-PEA to search on the subsonic inverse pressure design problem for a maximum of 100 generations are reported in Figures 6.3 and 6.4, respectively. Note that the n-subpopulation GE-PEA-0 has a uniform subpopulation size of 50. In contrast, the size of the ensembles of individuals are defined using equation (Eq. 6.11) in the n-subpopulation

GE-PEA, *i.e.*, 2 subpopulations ($pdpm : cemnet = 69 : 31$), 3 subpopulations ($pdpm : cemnet : surya = 83 : 37 : 30$), and 4 subpopulations ($pdpm : cemnet : surya : ec - pdccm = 100 : 45 : 35 : 20$).

More importantly, the results in Figure 6.3 and 6.4 show that the GE-PEA converges to the same optimal design airfoil shape as the GE-PEA-0 at significantly lesser amount of wall-clock time spent. Hence, this validates the efficacy of the proposed framework for parallel evolutionary optimization under diverse Grid environment.

Chapter 7

Conclusions & Future Works

7.1 Conclusions

Solving computationally expensive problems using evolutionary frameworks is a research field that has attracted significant attention among the evolutionary computation community and practitioners in science and engineering. Two major strategies commonly adopted in optimizing computationally expensive problems using EA are to use *1*) computationally less expensive surrogate models, and *2*) parallel and distributed computing technology. In this thesis, the different design issues arising from the management of surrogate models and parallel computing technology in evolutionary optimization of computationally expensive problems have been addressed. Some of these issues raised and addressed in this thesis include:

- *Inaccuracy of surrogate models.*
- *Search improvement through the use of imprecise surrogate models.*
- *Suitability of the response surface approximation method to the problem in hand.*
- *Management of multi-scale computational models.*
- *Use of surrogate models in multi-objective evolutionary optimization.*

- *Heterogeneity of computing resources due to Grid resources and variable fidelity fitness functions.*

In particular, the thesis has introduced and presented constructive methodologies and architectures necessary for the use of surrogates and parallel technologies in evolutionary optimization of computationally expensive problems that lead to good solution quality under a limited computational budget. In addition, novel surrogate-assisted evolutionary schemes have been proposed for dealing with problems plagued with computationally expensive objective functions. The main contribution of this dissertation work is summarized as follows:

- Chapter 3 provided a comprehensive survey on the major topics of this thesis, i.e., Surrogate-assisted and parallel evolutionary computation in heterogeneous Grid environment. This chapter, which poses as an appetizer to the reader before proceeding to the main highlights of the thesis in the subsequent chapters, serves to be a helpful reference for researchers or practitioners on related works and technology.

- Chapter 4 presented the proposed Generalized Surrogate Memetic framework for solving both single and multi-objective optimization problems, where a detailed empirical study on commonly used benchmark problems and a real-world problem on the airfoil design problem is reported.

- Chapter 5 extended the surrogate-assisted memetic frameworks discussed in previous chapter to multi-level evolutionary optimization of two real-world applications, *i.e.*, the water cluster structural optimization and aerodynamic airfoil design problems. From the second optimization problem, it is worth noting that multi-scale data-centric models can be fabricated through simple statistical measure of the original computational model.

- Finally, Chapter 6 described the research work on parallel evolutionary optimization on heterogeneous Grid computing environment, where a detail analysis on possible speed-up of the proposed Grid-enabled Parallel EA is presented. This is further substantiated with an empirical validation on both benchmark test problem and realistic airfoil design problem.

7.2 Future Works

Despite the extensive works towards the development of evolutionary frameworks for solving computationally expensive problems, there remains many challenging topics in the field that deserve further research studies. Some of these include:

- **Evolutionary frameworks for solving extremely high-dimensional and/or numerous objectives computationally expensive problems.** In real life situation, engineers and scientists are often faced with problems which are not only computationally expensive, but also in extremely high dimensionality and/or numerous objectives, *e.g.*, in the order of hundreds or even thousands of dimensions and/or objectives. A potentially credible idea is to mine for the crucial design variables and/or objectives. Subsequently, less meaningful variables may be removed or less emphasized, while less useful objectives are translated to equality or non-equality constraints.

- **Evolutionary frameworks for solving computationally expensive non-stationary problems.** Another relevant real-life scenario is the optimization of computationally expensive problems with time-variant input dimensions, constraints, and/or objectives. The common approach is to reuse information from previous environments to assist the search after the changes are detected. However, coupled with the issue of possible irrelevant information and high cost of each

new function evaluation, this topic remains a huge challenge for the evolutionary computation community.

- **Evolutionary search for robust solutions of computationally expensive problems.** Searching for robust solutions, *i.e.*, those which are resilient against uncertainties due to manufacturing errors or fluctuation in operating conditions or others, is generally regarded to be significantly more computationally expensive to optimize. This is due to many redundant function evaluations required to perform sensitivity analysis on each candidate design in order to estimate its robustness measure.

- **Greater exploration on the convergence properties of SAEAs.** In this thesis, the basic theoretical proof for SAEAs using Finite Markov Chain analysis, has been reported. It is worth noting that there may exist a variety of theoretical underpinnings that can be considered to arrive at the convergence proof. For instance, it would be interesting to see if stronger conditions for convergence in Markov chain analysis, such as the *detailed balance condition* could be applied in the context of SAEAs.

- **SAEAs in Grid computing environment.** Perhaps, the most intuitive motivation for combining SAEAs and Grid computing technologies is solely for gaining further acceleration of the evolutionary search by distributing the remaining expensive fitness evaluations across multiple compute nodes. Likewise, parallelization in Grid could be useful for accelerating the building of moderately expensive surrogate models. This is especially true in the case where multiple surrogates are used and the surrogate models' building time scales unfavorably with the problem size or fidelity level. Last but not least, another interesting motivation is to capitalize on

the possible algorithmic behavior changes due to high diversity generated by the heterogeneous Grid.

References

[1] Ansys. Online [http://www.ansys.com].

[2] CFD flow modeling software and services from fluent inc. Online [http://www.fluent.com].

[3] Cog kit wiki. Online [http://www.cogkit.org].

[4] Gaussian. Online [http://www.gaussian.com].

[5] Globus: Information services/mds. Online [http://www-unix.globus.org/toolkit/mds].

[6] Sysnoise. Online [http://ludit.kuleuven.be/software/packages/sysnoise.html].

[7] K. Abboud and M. Schoenauer. Surrogate deterministic mutation: Preliminary results. In P. Collet et al., editor, *EA 2001, LNCS 2310*, page 104116, 2002.

[8] D. Abramson and J. Abela. A parallel genetic algorithm for solving the school timetabling problem. In *Proceedings of the 15th Australian Computer Science Conference (ACSC-15)*, 1992.

[9] E. Acar and M. Rais-Rohani. Ensemble of metamodels with optimized weight factors. *Structural Multidisciplinary Optimization*, 2007. In Press.

[10] P. Adamidis. Review of parallel genetic algorithms bibliography. Technical report, Dept. of Electrical & Computer Engineering, Aristotle University of Thessaloniki, Greece, 1994.

[11] S. Agrawal, J. Dongarra, K. Seymour, and S. Vadhiyar. Netsolve: past, present, and future; a look at a grid enabled server. In F. Berman, G. Fox, and T. Hey, editors, *Grid Computing Making the Global Infrastructure a Reality*, pages 613–622. John Wiley and Sons, 2002.

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

[13] E. Alba and J. M. Troya. Influence of the migration policy in parallel distributed gas with structured and panmictic populations. *Appl. Intell.*, 12(3):163–181, 2000.

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

[15] N. Alexandrov, J. E. Dennis, R. M. Lewis, and V. Torczon. A trust region framework for managing the use of approximation models in optimization. *Journal on Structural Optimization*, 15(1):16–23, 1998.

[16] G. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *AFIPS Conference*, volume 30, pages 483–485, 1967.

[17] J. D. Anderson. *Introduction to Flight*. Oxford University Press, 2000.

[18] T. Back, D. B. Fogel, and Z. Michalewicz. *Handbook of evolutionary computation*. IOP, 1997.

[19] T. Baeck. *Evolutionary Algorithms in theory and practice*. Oxford University Press, 1996.

[20] J. E. Baker. Adaptive selection methods for genetic algorithms. In *Proc. the First International Conference on Genetic Algorithms and Their Applications*, Pittsburgh, PA, USA, 1985.

[21] M. Baker, R. Buyya, and D. Laforenza. The grid: international efforts in global computing. In *International Conference on Advances in Infrastructures for Electronic Business, Science, and Education on the Internet*, 2000.

[22] T. C. Belding. The distributed genetic algorithm revisited. In *6th International Conference on Genetic Algorithms*, pages 114–121, 1995.

[23] J. A. Biles. Genjam: A genetic algorithm for generating jazz solos. In *Proceedings of International Computer Music Conference*, pages 131–137, 1994.

[24] C. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

[25] J. Branke. Creating robust solutions by means of evolutionary algorithms. In *Proceedings of Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, pages 119–128. Springer, 1998.

[26] J. Branke, H. C. Andersen, and H. Schmeck. Global selection methods for massively parallel computers. In T. C. Fogarty, editor, *AISB'96 Workshop on Evolutionary Computing, LNCS 1143*, pages 175–188, 1996.

[27] D. Büche, N.N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with gaussian process fitness function models. *IEEE Trans. on Systems, Man, and Cybernetics: Part C*, 35(2):183–194, 2004. In press.

[28] L. T. Bui, H. A. Abbass, and D. Essam. Fitness inheritance for noisy evolutionary multi-objective optimization. In *Proceedings of the Genetic and Evolutionary COmputation Conference*, pages 25–29, 2005.

[29] L. Bull. On model-based evolutionary computation. *Soft Computing*, 3:76–82, 1999.

[30] C. Burnham and S. Xantheas. Development of transferable interaction models for water .iv. a flexible all-atom polarizable potential (ttm2-f) based on geometry dependent charges derived from an ab initio monomer dipole moment surface. *Journal of Chem. Phys.*, 116(12):5115 – 5124, 2002.

[31] E. Cantu-Paz. A survey of parallel genetic algorithms. *Calculateurs Paralleles, Reseaux et Systems Repartis*, 10(2):141–171, 1998.

[32] R. G. Carter. On the global convergence of trust-region algorithms using inexact gradient information. *SIAM Journal of Numerical Analysis*, 28(1):251–265, 1991.

[33] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary algorithms for solving multi-objective problems*. Kluwer Academic, 2002.

[34] J. P. Cohoon, S. U. Hedge, W. N. Martin, and D. Richards. Punctuated equilibria: a parallel genetic algorithm. In *2nd International Conference on Genetic Algorithms*, pages 148–154, 1987.

[35] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, 2001.

[36] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel Problem Solving from Nature VI Conference, LNCS 1917*, pages 849–858, 2000.

[37] K. Deb and P Nain. An evolutionary multi-objective adaptive meta-modeling procedure using artificial neural networks. pages 297–322. Springer, 2007.

[38] J. G. Digalakis and K. G. Margaritis. On benchmarking functions for genetic algorithms. *Intern. J. Computer Math.*, 77(4):481–506, 2001.

[39] D. Eby, R. Averill, W. Punch, and E. Goodman. Evaluation of injection island model GA performance on flywheel design optimization. In *Third Conference on Adaptive Computing in Design and manufacturing*, pages 121–136. Springer, 1998.

[40] A. E. Eiben, E. H. L. Aarts, and H. K. M. Van. Global convergence of genetic algorithms: a markov chain analysis. In *First International Conference on Parallel Problem Solving from Nature (PPSN)*, pages 4–12, 1991.

[41] M. A. El-Beltagy and A. J. Keane. Optimization for multi-level problems: A comparison of various algorithms. In I. Parmee, editor, *Proceedings of Third International Conference on Adaptive Computing in Design and Manufacture*, pages 111–120. Springer, 1998.

[42] L. Elliott and et al. Efficient clustering-based genetic algorithms in chemical kinetic modeling. In *Genetic and Evolutionary Computation Conference*, pages 932–944, 2004.

[43] L. Elliott, D. B. Ingham, A. G. Kyne, N. S. Mera, M. Pourkashanian, and C. W. Wilson. An informed operator based genetic algorithm for tuning the reaction rate parameters of chemical kinetics mechanisms. In *GECCO 2004, LNCS 3103*, pages 945–956, 2004.

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

[45] M. Emmerich, A. Giotis, M. Özdenir, T. Bäck, and K. Giannakoglou. Metamodel-assisted evolution strategies. In *Parallel Problem Solving from Nature*, number 2439 in Lecture Notes in Computer Science, pages 371–380. Springer, 2002.

[46] M. Erickson, A. Mayer, and J. Horn. The niched pareto genetic algorithm 2 applied to the design of groundwater remediation systems. In *First International Conference on Evolutionary Multi-Criterion Optimization - LNCS 1993*, pages 681–695, 2000.

[47] M. Farina. A neural network based generalized response surface multiobjective evolutionary algorithms. In *Congress on Evolutionary Computation*, pages 956–961. IEEE Press, 2002.

[48] C. A. Floudas. *Deterministic global optimization*. Springer, 1999.

[49] T. C. Fogarty and R. Huang. Implementing the genetic algorithm on transputer based parallel processing systems. In *Parallel Problem Solving from Nature*, pages 145–149, 1991.

[50] D. B. Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence*. IEEE, 1995.

[51] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial intelligence through simulated evolution*. Wiley, 1966.

[52] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: formulation, discussion and generalization. In *Fifth International Conference on Genetic Algorithms*, pages 416–423, 1993.

[53] I. Foster. The globus toolkit for grid computing. In *1st International Symposium on Cluster Computing and the Grid*, 2001.

[54] I. Foster and C. Kesselman. *The grid: blueprint for a new computing infrastructure*. Morgan Kaufman Publishers, 1999.

[55] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: an open grid services architecture for distributed systems integration. 2002.

[56] I. Foster, C. Kesselman, and S. Tuecke. The anatomy of the grid: enabling scalable virtual organizations. *International J.Supercomputer Applications*, 15(3), 2001.

[57] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke. Condor-g: A computation management agent for multi-institutional grids. In *Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)*, 2001.

[58] A. Gaspar-Cunha and A. Vieira. A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations. *International Journal of Computers, Systems and Signals*, 6(1):18–36, 2005.

[59] D. Geer. Grid computing using the sun grid engine. [Online], 2003.

[60] K. C. Giannakoglou. Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *International Review Journal Progress in Aerospace Sciences*, 38(5):43–76, 2002.

[61] K. C. Giannakoglou, D. I. Papadimitriou, and I. C. Kampolis. Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodels. *Computer Methods in Applied Mechanics and Engineering*, 195:6312–6329, 2006.

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

[63] D. E. Goldberg. *Genetic Algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.

[64] D. E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: motivation, analysis, and first results. *Complex Systems*, 3:493–530, 1989.

[65] Y. Gong, M. Nakamura, and S. Tamaki. Parallel genetic algorithms on line topology of heterogeneous computing resources. In *2005 Conference on Genetic and Evolutionary Computation*, 2005.

[66] M. Gorges-Schleuter. Asparagos: a population genetics approach to genetic algorithms. *Evolution and Optimization*, 1989.

[67] M. Gorges-Schleuter. Explicit parallelism of genetic algorithms through poplation structures. In *Parallel Problem Solving from Nature*, pages 150–159, 1991.

[68] Gräning, Y. Jin, and B. Sendhoff. Individual-based management of meta-models for evolutionary optimization with applications to three-dimensional blade optimization. In *S. Yang, Y.-S. Ong, Y. Jin(eds.), Evolutionary Computation in Dynamic and Uncertain Environments*, pages 225–250. Springer, 2007.

[69] P. B. Grosso. *Computer simulations of genetic adaptation: parallel subcomponent interaction in a multilocus model.* PhD thesis, The University of Michigan, 1985.

[70] F. Gruau. *Neural networks synthesis using cellular encoding and the genetic algorithm.* PhD thesis, Univ. Claude Bernard-Lyon I, France, 1994.

[71] F. G. Guimaraes, F. Campelo, H. Igarashi, D. A. Lowther, and J. A. Ramirez. Optimization of cost functions using evolutionary algorithms with local learning and local search. *IEEE Transactions on Magnetics*, 43(4):1641–1644, 2007.

[72] H.-M Gutmann. On the semi-norm of radial basis function interpolants. Technical Report DAMTP 2000/NA04, Dept. Applied Math. Theor. Phy., Univ. Cambridge, UK, 2000.

[73] K. Hamza and K. Saitou. Vehicle crashworthiness design via a surrogate model ensemble and a co-evolutionary genetic algorithm. In *ASME 2005 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, 2005.

[74] R. Hauser and R. Maenner. Implementation of standard genetic algorithm on mimd machine. In *Parallel Problem Solving from Nature, PPSN III*, pages 504–513, 1994.

[75] R. M. Hicks and P. A. Henne. Wing design for numerical optimization. *Journal of Aircraft*, 15(7):407 – 412, 1978.

[76] R. M. Hicks and P. A. Henne. Chemical bonding in water clusters. *Journal of Chem. Phys.*, 102:1266, 1995.

[77] Q. T. Ho, W. T. Cai, and Y. S. Ong. Design and implementation of an efficient multi-cluster gridrpc system. In *IEEE Cluster Computing and Grid Conference*, pages 358–365. Springer, 2005.

[78] F. Hoffmeister. Scalable parallelism by evolutionary algorithms. *Lecture Notes in Mathematical Systems and Economics*, 1991.

[79] J. H. Holland. Outline for a logical theory of adaptive system. *J. ACM*, 3:297–314, 1962.

[80] 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, 2003.

[81] J. Horn and N. Nafpliotis. Multiobjective optimization using the niched pareto genetic algorithm. Technical Report 93005, IlliGAl, University of Illinois, Urbana-Champaign, Urbana, Illinois, USA, 1993.

[82] H. Iba, H. deGaris, and T. Sato. Genetic programming with local hill-climbing. In *PPSN 3*, 1994.

[83] H. Ishibuchi and T. Murata. Multi-objective genetic local search algorithm. *IEEE International Conference on Evolutionary Computation*, pages 119–124, 1996.

[84] A. Jaszkiewicz. Genetic local combinatorial optimization. Technical Report RA-014/98, Institute of Computing Science, Poznan University of Technology, 1998.

[85] X. Jiang, D. Chafekar, and K. Rasheed. Constrained multi-objective GA optimization using reduced models. In *Proceedings of GECCO Workshop on Learning, Adaptation and Approximation in Evolutionary Computation*, pages 174–177, 2003.

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

[87] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments - a survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–317, 2005.

[88] Y. Jin, M. Hüsken, M. Olhofer, and B. Sendhoff. Neural networks for fitness approximation in evolutionary optimization. In Y. Jin, editor, *Knowledge Incorporation in Evolutionary Computation*, pages 281–305. Springer, Berlin, 2004.

[89] Y. Jin, M. Hüsken, and B. Sendhoff. Quality measures for approximate models in evolutionary computation. In *Proceedings of GECCO Workshops: Workshop on Adaptation, Learning and Approximation in Evolutionary Computation*, pages 170–174, Chicago, 2003.

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

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

[92] 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.

[93] Y. Jin and B. Sendhoff. Reducing fitness evaluations using clustering techniques and neural networks ensembles. In *Genetic and Evolutionary Computation Conference*, volume 3102 of *LNCS*, pages 688–699. Springer, 2004.

[94] B. Johanson and R. Poli. GP-music: An interactice genetic programming system for music generation with automated fitness raters. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo, editors, *Proceedings of the Third Annual Conference on Genetic Programming*, pages 181–186, 1998.

[95] C. G. Johnson and J. J. R. Caldalda. Introduction: Genetic algorithms in visual art and music. *Leonardo*, 35(2):175–184, 2002.

[96] Hirotaka Kaji and Hajime Kita. Acceleration of experiment-based evolutionary multi-objective optimization of internal-combustion engine controllers using fitness estimation. In *IEEE Congress on Evolutionary Computation*, pages 1777–1784, 2007.

[97] I. C. Kampolis, A. S. Zymaris, V. G. Asouti, and K. C. Giannakoglou. Multilevel optimization strategies based on metamodel - assisted evolutionary algorithms, for computationally expensive problems. In *IEEE Congress on Evolutionary Computation*, pages 4116–4123, 2007.

[98] M. K. Karakasis and K. C. Giannakoglou. Metamodel-assisted multi-objective evolutionary optimization. In *Evolutionary and Deterministic Methods for Design,Optimization and Control with Applications to Industrial and Societal Problems EUROGEN 2005*, 2005.

[99] A. J. Keane. The options design exploration system *http://www.soton.ac.uk/ ajk/options/welcome.html*. 1995.

[100] C. T. Kelley. *Solving Nonlinear Equations with Newton's Method*. in 'Fundamentals of Algorithms' series, SIAM Press., 2003.

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

[102] 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.

[103] J. Knowles and D. Corne. The pareto archived evolution strategy: a new baseline algorithm for multiobjective optimization. In *Congress on Evolutionary Computation*, pages 98–105, 1999.

[104] J. Knowles and D. Corne. Memetic algorithms for multiobjective optimization: issues, methods andprospects. In W. E. Hart, N. Krasnogor, and J. E. Smith, editors, *Recent Advances in Memetic Algorithms*, pages 313–352. 2005.

[105] J. Knowles and E. J. Hughes. Multiobjective optimization on a budget of 250 evaluations. In *EMO 2005, LNCS 3410*, pages 176–190, 2005.

[106] J. R. Koza. *Genetic Programming: on the programming of computers by means of natural selection*. MIT Press, 1992.

[107] C. T. Lawrence and A. L. Tits. A computationally efficient feasible sequential quadratic programming algorithm. *Society for Industrial and Applied Mathematics*, 11(4):1092–1118, 2001.

[108] F. H. Lesh. Multi-dimensional least-square polynomial curvefitting. *Communications of ACM*, 2(9):29–30, 1959.

[109] P. Liakopoulos, I. Kampolis, and K. Giannakoglou. Grid-enabled, hierarchical distributed metamodel-assisted evolutionary algorithms for aerodynamic shape optimization. *Future Generation Computer Systems*, 24:701–708, 2008.

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

[111] 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.

[112] X. Liao, Q. Li, X. Yang, W. Zhang, and W. Li. Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural Multidisciplinary Optimization*, (DOI 10.1007/s00158-007-0163-x). In Press.

[113] S. H. Lin, E. D. Goodman, and W. F. Punch. Investigating parallel genetic algorithms on job shop scheduling problems. In *6th International Conference on Evolutionary Programming*, pages 383–393, 1997.

[114] Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.

[115] X. Llora, K. Sastry, D. E. Goldberg, A. Gupta, and L. Lakshmi. Combating user fatigue in igas: partial ordering, support vector machines, and synthetic fitness. In *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, pages 1363–1370, 2005.

[116] K. Watanabe M. M. A. Hashem and K. Izumi. Evolutionary trajectory learning for autonomous robots by means of geometric approximations of polygonal obstacles. volume 2, pages 734–739, 1999.

[117] D. J. C. Mackay. Introduction to gaussian processes. *Neural Networks and Machine Learning*, 168:133–165, 1998.

[118] B. Manderick and P. Spiessens. Fine-grained parallel genetic algorithms. In *3rd International Conference on Genetic Algorithms*, pages 428–433, 1989.

[119] D. Marjavaara, S. Lundström, and W. Shyy. Hydraulic turbine diffuser shape optimization by multiple surrogate model approximations of pareto fronts. *ASME Journal of Fluids Engineering*, 129(9):1228–1240, 2007.

[120] M. Massie, B. Chun, and D. Culler. The ganglia distributed monitoring system: design, implementation, and experience. Technical report, University of California, Berkeley, 2003.

[121] M. Mella, D. C. Clary, J.-L. Kuo, and M. L. Klein. Nuclear quantum effects on the structure and energetics of $(h_2o)_6h+$. *Journal of Chem. Phys.*, 7:2324, 2005.

[122] Z. Michalewicz. Genetic algorithms, numerical optimization, and constraints. In *University of Pittsburgh*, pages 151–158. Morgan Kaufmann, 1995.

[123] K. M. Miettinen. *Nonlinear multiobjective optimization*. Kluwer Academic Publishers, 2004.

[124] G. E. Moore. Cramming more components onto integrated circuits. *International J.Supercomputer Applications*, 38(8), 1965.

[125] H. Mühlenbein. Parallel genetic algorithms, population genetics, and combinatorial optimization. *Evolution and Optimization*, pages 79–85, 1989.

[126] P. Nain and K. Deb. A computationally effective multi-objective search and optimization techniques using coarse-to-fine grain modeling. In *2002 PPSN Workshop on Evolutionary Multiobjective Optimization*, 2002.

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

[128] H. Nakada, S. Matsuoka, K. Seymour, J. Dongarra, C. Lee, and H. Casanova. Gridrpc: A remote procedure call api for grid computing. In *Grid Computing - Grid 2002 - LNCS 2536*, pages 274–278. Springer, 2002.

[129] H. K. Ng, D. Lim, Y. S. Ong, B. S. Lee, L. Freund, S. Parvez, and B. Sendhoff. A multi-cluster grid enabled evolution framework for aerodynamic airfoil design optimization. In *Lecture Notes in Computer Science*, volume 3611, pages 1112–1121, 2005.

[130] Q. C. Nguyen, Y. S. Ong, H. Soh, and J.-L Kuo. Multiscale approach to explore the potential energy surface of water clusters $(h_2o)_8$ $n \leq 8$. *Journal of Phys. Chem. A*, 112(28):6257–6261, 2008.

[131] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, 1999.

[132] N. Noman and H. Iba. Accelerating differential evolution using an adaptive local search. *IEEE/ACM Transactions on Evolutionary Computation*, 12(1):107–125, 2007.

[133] N. Noman and H. Iba. Inferring gene regulatory networks using differential evolution with local search heuristics. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(4):634–647, 2007.

[134] M. Nowostawski and R. Poli. Parallel genetic algorithm taxonomy. In *Third International conference on knowledge-based intelligent information engineering systems*, pages 88–92, 1999.

[135] L. Ojamae. Potential models for simulations of the solvated proton in water. *Journal of Chem. Phys.*, 109(13):5547, 1998.

[136] Y. S. Ong and A. J. Keane. A domain knowledge based search advisor for design problem solving environments. *Engineering Applications of Artificial Intelligence*, 15(1):105–116, 2002.

[137] Y. S. Ong and A. J. Keane. Meta-lamarckian learning in memetic algorithm. *IEEE Transactions on Evolutionary Computation*, 8(2):99–110, April 2004.

[138] Y. S. Ong, A.J. Keane, and P.B. Nair. Surrogate-assisted coevolutionary search. In *9th International Conference on Neural Information Processing, Special Session on Trends in Global Optimization*, pages 2195–2199, Singapore, 2002.

[139] Y. S. Ong, J. Knowles, C. K. Goh, and K. C. Tan. Special session on evolutionary computation for expensive optimization problems. In *IEEE Congress on Evolutionary Computation*, 2007.

[140] Y. S. Ong, M. H. Lim, N. Zhu, and K. W. Wong. Classification of adaptive memetic algorithms: a comparative study. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 36(1):141–152, 2006.

[141] Y. S. Ong, K. Y. Lum, and P. B. Nair. Evolutionary algorithm with hermite radial basis function interpolants for computationally expensive adjoint solvers. *Computational Optimization and Applications*, 39(1):97–119, 2008.

[142] 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.

[143] Y. S. Ong, P. B. Nair, A. J. Keane, and K. W. Wong. Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In Y. Jin, editor, *Knowledge Incorporation in Evolutionary Computation*, Studies in Fuzziness and Soft Computing, pages 307–332. Springer, 2004.

[144] 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.

[145] Y. S. Ong, Z. Zhou, and D. Lim. Curse and blessing of uncertainty in evolutionary algorithm using approximation. In *Congress on Evolutionary Computation*, pages 2928–2935, 2006.

[146] I. Paenke, J. Branke, and Y. Jin. Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. *IEEE Transactions on Evolutionary Computation*, 10(4):405420, 2006.

[147] M. F. Pantoja, P. Meincke, and A. R. Bretones. A hybrid genetic-algorithm space-mapping tool for the optimization of antennas. *IEEE Transactions on Antennas and Propagation*, 55(3):777–781, 2007.

[148] M. Papadrakakis, N. Lagaros, and Y. Tsompanakis. Structural optimization using evolution strategies and neural networks. In *Fourth U.S. National Congress on Computational Mechanics*, San Francisco, 1997.

[149] M. Papadrakakis, N. Lagaros, and Y. Tsompanakis. Optimization of large-scale 3D trusses using evolution strategies and neural networks. *Int. J. Space Structures*, 14(3):211–223, 1999.

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

[151] C. Pettey, M. R. Leuze, and J. J. Grefenstette. A parallel genetic algorithm. In *2nd International Conference on Genetic Algorithms*, pages 155–161, 1987.

[152] S. Pierret. Turbomachinery blade design using a Navier-Stokes solver and artificial neural network. *ASME Journal of Turbomachinery*, 121(3):326–332, 1999.

[153] J. D. Pinter. *Global optimization in action*. Kluwer, 1996.

[154] The Globus Project. Gridftp universal data transfer for the grid. [Online] The Globus Project White Paper, 2000.

[155] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P.K. Tucker. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 37:59–118, 2001.

[156] M. J. Quinn. *Parallel Programming in C with MPI and OpenMP*. McGraw Hill, 2004.

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

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

[159] A. Ratle. Optimal sampling strategies for learning a fitness model. In *Proceedings of 1999 Congress on Evolutionary Computation*, volume 3, pages 2078–2085, Washington D.C., July 1999.

[160] T. Ray, H. Tsai, and C. Tan. Effects of solver fidelity on a parallel search algorithm's performance for airfoil shape optimization problems. In *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization Conference*, 2002.

[161] I. Rechenberg. Cybernetic solution path of an experimental problem. Library Transalation No. 1122 Royal Aircraft Establishment, Farnborough, UK, 1965.

[162] I. Rechenberg. Evolutionsstrategie '94. *Werkstatt Bionik und Evolutionstechnik*, 1, 1994.

[163] 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):490505, 2004.

[164] R. G. Regis and C. A. Shoemaker. Constrained global optimization of expensive black box functions using radial basis functions. *Journal of Global Optimization*, 31:153–171, 2005.

[165] M. Reyes-Sierra and C. A. Coello Coello. Fitness inheritance in multi-objective particle swarm optimization. In *IEEE Swarm Intelligence Symposium*, pages 116–123, 2005.

[166] M. Reyes-Sierra and C. A. Coello Coello. A study of tehniques to improve the efficiency of a multi-objective particle swarm optimizer. In *Evolutionary Optimization in Dynamic and Uncertain Environments*, pages 269–296. Springer, 2007.

[167] G. Robinson. *Water in biology, chemistry and physics: experimental overviews and computational methodologies*, volume 9. World Scientific, 1996.

[168] J. F. Rodriguez, J.E. Renaud, and L.T. Watson. Convergence of trust region augmented lagrangian methods using variable fidelity approximation data. *Structural Optimization*, 15(3-4):141–156, 1998.

[169] G. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Network*, 5(1):96–101, 1994.

[170] T. Runarsson and X. Yao. Continuous selection and self-adaptive evolution strategies. In *CEC*, 2002.

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

[172] R. Salomon. Evolutionary algorithms and gradient search: similarities and differences. *IEEE Transactions on Evolutionary Computation*, 2(2):45–55, 1998.

[173] A. Samad and K.-Y. Kim. Multiple surrogate modeling for axial compressor blade shape optimization. *Journal of Propulsion and Power*, 24(2):302–310, 2008.

[174] A. Samad, K.-D. Lee, K.-Y. Kim, and R. T. Haftka. Application of multiple-surrogate model to optimization of a dimpled channel. In *7th World Congress on Structural and Multidisciplinary Optimization*, pages 2276–2282, 2007.

[175] E. Sanchez, S. Pintos, and N. V. Queipo. Toward an optimal ensemble of kernel-based approximations with engineering applications. *Structural Multidisciplinary Optimization*, 2007. In press.

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

[177] J. Sarma and K. D. Jong. An analysis of the effects of neighbornood size and shape on local selection algorithms. In *Parallel Problem Solving from Nature IV*, pages 236–244, 1996.

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

[179] K. Sastry, D. E. Goldberg, and M. Pelikan. Efficiency enhancement of probabilistic model building genetic algorithm. In *GECCO 2004*, 2004.

[180] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *First International Conference on Genetic Algorithms*, pages 93–100, 1985.

[181] G. Schneider. Neural networks are useful tools for drug design. *Neural Networks*, 13:15–16, 2000.

[182] G. Schneider, W. Schrödl, and G. Wallukat. Peptide design by artificial neural networks and computer-based evolutionary search. *Proceedings of National Academy of Science*, 95:12197–12184, 1998.

[183] G. Schneider, J. Schuchhardt, and P. Wrede. Artificial neural networks and simulated molecular evolution are potential tools for sequence-oriented protein design. *CABIOS*, 10(6):635–645, 1994.

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

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

[186] H. Soh, Y. S. Ong, M. Salahuddin, T. Hung, and B. S. Lee. Playing in the objective space: coupled approximators for multi-objective optimization. In *IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, 2007.

[187] W. Song. Evolutionary shape optimization using gaussian process. In Y.-S. Ong S. Yang and Y. Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, pages 251–267. Springer, 2007.

[188] J. C. Spall. *Introduction to stochastic search and optimization*. Wiley, 2003.

[189] N. Srinivas and K. Deb. Multi-objective optimization using non-dominated sorting in genetic algorithms. *IEEE Transaction on Evolutionary Computation*, 2(3):221–248, 1994.

[190] R. Storn and K. V. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Opt.*, 11(4):341–359, 1997.

[191] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical Report 2005005, Nanyang Technological University (Singapore) and KanGAL (India), 2005.

[192] J. Suzuki. A markov chain analysis on simple genetic algorithms. *IEEE Transactions on System, Man, and Cybernetics*, 25(4):655–659, 1995.

[193] R. Tanese. Parallel genetic algorithm for a hypercube. In *2nd International Conference on Genetic Algorithms*, pages 177–183, 1987.

[194] P. L. Toint. Global convergence of a class of trust region methods for nonconvex minimization in hilbert space. *IMA Journal of Numerical Analysis*, 8(2):231–252, 1988.

[195] A. Torn and A. Zilinskas. *Global Optimization*. Lecture Notes in Computer Science, Vol. 350, Springer-Verlag, 1989.

[196] S. Tuecke. Grid security infrastructure (gsi) roadmap. [Online] Internet Draft Document: draft-Gridforum-gsi-roadmap-02.txt, 2001.

[197] H. Ulmer, F. Streichert, and A. Zell. Evolution strategies assisted by gaussian processes with improved pre-selection criterion. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 692–699, 2003.

[198] H. Ulmer, F. Streichert, and A. Zell. Evolution strategies with controlled model assistance. In *Congress on Evolutionary Computation*, pages 1569–1576. IEEE, 2004.

[199] D. A. Van Veldhuizen. *Multiobjective evolutionary algorithms: classsifications, analysis, and new innovations*. PhD thesis, Air Force Institute of Technology Dayton, OH, 1999.

[200] D. A. Van Veldhuizen and G. B. Lamont. Evolutionary computation and convergence to a pareto front. In *Late Breaking Papers at the Genetic Programming*, pages 221–228. 1998.

[201] H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics - The Finite Volume Method*. Longman Group Ltd., 1995.

[202] G.-C. Vosniakos, A. Tsifakis, and P. Benardos. Neural network simulation metamodels and genetic algorithms in analysis and design of manufacturing cells. *Int. J. Adv. Manuf. Technol.*, 29:541–550, 2006.

[203] J. Wakunda and A. Zell. A new selection scheme for steady-state evolution strategies. In *GECCO 2000*.

[204] T. Weise. *Global optimization algorithms - theory and application*. e-book, 2007.

[205] 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*, 47:197–208, 2005.

[206] X. Zheng, B. A. Julstrom, and W. Cheng. Design of vector quantization codebooks using a genetic algorithm. In *IEEE International Conference on Evolutionary Computation*, pages 525–529, 1997.

[207] Z. Zhou, Y. S. Ong, M. H. Lim, and B. S. Lee. Memetic algorithm using multi-surrogates for computationally expensive optimization problem. *Soft Computing Journal*, 11(10):957–971, 2007.

[208] Z. Zhou, Y. S. Ong, and P. B. Nair. Hierarchical surrogate-assisted evolutionary optimization framework. In *Congress on Evolutionary Computation*, pages 1586–1593. IEEE, 2004.

[209] 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*, 37(1):66–76, 2007.

[210] Z. Zhou, Y. S. Ong, M. H. Nguyen, and D. Lim. A study on polynomial regression and gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm. In *Congress on Evolutionary Computation*, volume 3, pages 2832–2839. IEEE, 2005.

[211] E. Zitzler. *Evolutionary algorithms for multiobjective optimization: methods and applications*. PhD thesis, Swiss Federal Institute of Technology (ETH) Zurich, Switzerland, TIK-Schriftenreihe Nr. 30, Diss ETH No. 13398, Aachen, Germany, 1999.

[212] E. Zitzler, K. Deb, and L. Thiele. Comparison of multi-objective evolutionary algorithms: empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

[213] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: improving the strength pareto evolutionary algorithm. Technical report, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.

[214] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms a comparative case study. In *Fifth International Conference on Parallel Problem Solving from Nature (PPSN-V)*. Springer, 1998.

[215] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[216] E. Zitzler, L. Thiele, M. Laumanns, C. M. Foneseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

Appendix A

Approximation/Surrogate Modeling Techniques

Here, we provide a brief review on three different surrogate modeling techniques used in this thesis, namely: Kriging/Gaussian Process (GP), Polynomial Regression (PR), and Radial Basis Function (RBF). Throughout this section, let $\mathcal{D} = \{\mathbf{x}_i, t_i\}, i = 1 \dots m$ denote the training dataset, where $\mathbf{x}_i \in \mathbb{R}^d$ is an input design vector and $t_i \in \mathbb{R}$ is the corresponding target value.

A.1 Kriging/Gaussian Process

The GP surrogate model [117] assumes the presence of an unknown true modeling function $f(\mathbf{x})$ and an additive noise term v to account for anomalies in the observed data. Thus:

$$t = f(\mathbf{x}) + v \qquad \text{(Eq. A.1)}$$

The standard analysis requires the specification of prior probabilities on the modeling function and the noise model. From a stochastic process viewpoint, the collection $\mathbf{t} = \{t_1, t_2, ..., t_m\}$ is called a Gaussian process if every subset of $\mathbf{t}$ has a joint Gaussian distribution. More specifically,

$$P(\mathbf{t}|\mathbf{C}, \{\mathbf{x}_m\}) = \frac{1}{Z} \exp\left(-\frac{1}{2}(\mathbf{t} - \boldsymbol{\mu})^T \mathbf{C}^{-1} (\mathbf{t} - \boldsymbol{\mu})\right) \qquad \text{(Eq. A.2)}$$

where $\mathbf{C}$ is a covariance matrix parameterized in terms of hyperparameters $\boldsymbol{\theta}$, *i.e.*, $\mathbf{C}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j; \theta)$ and $\boldsymbol{\mu}$ is the process mean. The Gaussian process is characterized by this

covariance structure since it incorporates prior beliefs both about the true underlying function as well as the noise model. In the present study, we use the following exponential covariance model

$$k(\mathbf{x}_i, \mathbf{x}_j) = e^{-(\mathbf{x}_i - \mathbf{x}_j)^T \Theta (\mathbf{x}_i - \mathbf{x}_j)} + \theta_{d+1} \qquad \text{(Eq. A.3)}$$

where $\Theta = \text{diag}\{\theta_1, \theta_2, ..., \theta_d\} \in \mathbb{R}^{d \times d}$ is a diagonal matrix of undetermined hyperparameters, and $\theta_{d+1} \in \mathbb{R}$ is an additional hyperparameter arising from the assumption that noise in the dataset is Gaussian (and output dependent). We shall henceforth use the symbol $\boldsymbol{\theta}$ to denote the vector of undetermined hyperparameters, *i.e.*, $\boldsymbol{\theta} = \{\theta_1, \theta_2, ..., \theta_{d+1}\}$. In practice, the undetermined hyperparameters are tuned to the data using the evidence maximization framework. Once the hyperparameters have been estimated from the data, predictions can be readily made for a new testing point.

A.2 Polynomial Regression

In PR metamodeling technique [108], we define an exponent vector ε containing positive integers $(\pi_1, \pi_2, \ldots, \pi_d)$ and define $\mathbf{x}_i^{\varepsilon}$ as an exponent input vector $({x_{i_1}}^{\pi_1}, {x_{i_2}}^{\pi_2}, \ldots, {x_{i_d}}^{\pi_d})$.

Given a set of exponent vectors $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_o$ and the set of data $(\mathbf{x}_i, t_i)$, where $i = 1, 2, \ldots, m$, the polynomial model of $(o-1)^{th}$ order has the form:

$$\hat{t}_i = C_1 \mathbf{x}_i^{\varepsilon_1} + C_2 \mathbf{x}_i^{\varepsilon_2} + \ldots + C_m \mathbf{x}_i^{\varepsilon_o} \qquad \text{(Eq. A.4)}$$

where $C_1, C_2, \ldots, C_o$ are the coefficient vectors to be estimated, and $C_j = (c_{j_1}, c_{j_2}, \ldots, c_{j_d})$, $j = 1, 2, \ldots, o$.

The least square method is then used to estimate the coefficients of the polynomial model. By definition, the least square error E to be minimized is:

$$E = \sum_{i=1}^{m} [t_i - \hat{t}_i]^2 \qquad \text{(Eq. A.5)}$$

It may be easily shown that $t_i = f(\mathbf{x}_i)$, and by multiplying both sides of equation (Eq. A.4) with $\mathbf{x}_i^{\varepsilon_j}$ and taking the sum of m pairs of input-output data, we arrive at

$$C_1 \sum_i \mathbf{x}_i^{\varepsilon_1 + \varepsilon_j} + \ldots + C_o \sum_i \mathbf{x}_i^{\varepsilon_o + \varepsilon_j} = \sum_i t_i \mathbf{x}_i^{\varepsilon_j} \qquad \text{(Eq. A.6)}$$

For $j = 1, 2, \ldots, o$, the polynomial model for the training dataset can be represented in the matrix notation as follows

$$A\gamma^T = \boldsymbol{b}^T \tag{Eq. A.7}$$

where

$$A = \begin{bmatrix} \sum_i \mathbf{x}_i^{\varepsilon_1+\varepsilon_1} & \ldots & \sum_i \mathbf{x}_i^{\varepsilon_1+\varepsilon_o} \\ \vdots & & \vdots \\ \sum_i \mathbf{x}_i^{\varepsilon_o+\varepsilon_1} & \ldots & \sum_i \mathbf{x}_i^{\varepsilon_o+\varepsilon_o} \end{bmatrix} \tag{Eq. A.8}$$

$$\boldsymbol{b} = (\sum t_i \mathbf{x}_i^{\varepsilon_1}, \ \ldots \ , \sum t_i \mathbf{x}_i^{\varepsilon_o}) \tag{Eq. A.9}$$

$$\gamma = (C_1, C_2, \ldots, C_o) \tag{Eq. A.10}$$

Then the coefficient matrix of the polynomial is:

$$\gamma = (A^{-1}\boldsymbol{b}^T)^T \tag{Eq. A.11}$$

Let $B_i = (\mathbf{x}_i^{\varepsilon_1}, \ldots, \mathbf{x}_i^{\varepsilon_o})$, the following equations may be derived:

- $A = \sum_i B_i^T B_i$
- $\boldsymbol{b} = \sum_i t_i B_i$
- $\hat{t}_i = \gamma . B_i^T$

The predicted output for a new input pattern is then given by $\hat{t}_i = \gamma . B_i^T$.

A.3 Radial Basis Function

The surrogate models of RBF used in this thesis are interpolating radial basis function networks of the form

$$\hat{t} = \hat{f}(\mathbf{x}) = \sum_{i=1}^{m} \alpha_i K(||\mathbf{x} - \mathbf{x}_i||) \tag{Eq. A.12}$$

where $K(||\mathbf{x} - \mathbf{x}_i||) : \mathbb{R}^d \rightarrow \mathbb{R}$ is a RBF and $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_m\} \in \mathbb{R}^m$ denotes the vector of weights. Hence, the number of hidden nodes in the RBF here is as many as the number of training points.

Typical choices for the kernel include linear splines, cubic splines, multiquadrics, thin-plate splines, and Gaussian functions [24]. Recent studies in [163][72], indicate that the linear, cubic, and thin plate spline RBFs have better theoretical properties than the multiquadric and Gaussian RBFs. Hence, in this thesis, we opt to use linear spline kernel function. The structure of some commonly used radial basis kernels and their parameterization are shown in Table A.1 Given a suitable kernel, the weight vector can be computed by solving the linear algebraic system of equations $\mathbf{K}\boldsymbol{\alpha} = \mathbf{t}$, where $\mathbf{t} = \{t_1, t_2, \ldots, t_m\} \in \mathbb{R}^m$ denotes the vector of outputs and $\mathbf{K} \in \mathbb{R}^{m \times m}$ denotes the Gram matrix formed using the training inputs (*i.e.*, the ijth element of $\mathbf{K}$ is computed as $K(||\mathbf{x}_i - \mathbf{x}_j||)$).

Table A.1: Radial Basis Kernels

Linear Splines	$\|\|\mathbf{x} - \mathbf{c}_i\|\|$
Thin Plate Splines	$\|\|\mathbf{x} - \mathbf{c}_i\|\|^k ln \|\|\mathbf{x} - \mathbf{c}_i\|\|$
Cubic Splines	$\|\|\mathbf{x} - \mathbf{c}_i\|\|^3$
Gaussian	$e^{-\frac{\|\|\mathbf{x}-\mathbf{c}_i\|\|^2}{\beta_i}}$
Multiquadrics	$\sqrt{1 + \frac{\|\|\mathbf{x}-\mathbf{c}_i\|\|^2}{\beta_i}}$
Inverse Multiquadrics	$(1 + \frac{\|\|\mathbf{x}-\mathbf{c}_i\|\|^2}{\beta_i})^{-\frac{1}{2}}$

Appendix B

Single/Multi-objective Benchmark Problems

Here, we provide details on the single and multi-objective benchmark problems used throughout this thesis.

B.1 Single-Objective Benchmark Functions

Single-objective benchmark functions used in this thesis are presented in this section. The shifted and/or rotated functions are taken from [38] and [191]. From F4-F10, the following nomenclature applies:

$\mathbf{o} = [o_1, o_2, \ldots, o_d]$: the shifted global optimum

$\mathbf{M}$: linear transformation matrix, obtained from [191].

Ackley

$$F(\mathbf{x}) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}} - e^{\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)} \qquad \text{(Eq. B.1)}$$

$$-32.768 \leq x_i \leq 32.768, i = 1, 2, \ldots, d.$$

Global optimum $x_i^* = 0.0$ for $i = 1, \ldots, d$, $F(\mathbf{x}^*) = 0.0$

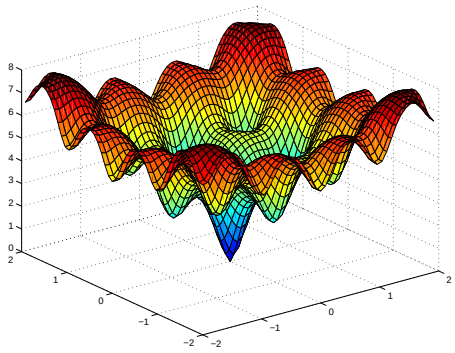


Figure B.1: Ackley Function

Griewank

$$F(\mathbf{x}) = 1 + \sum_{i=1}^{d} x_i^2/4000 - \prod_{i=1}^{d} \cos(x_i/\sqrt{i}) \qquad \text{(Eq. B.2)}$$
$$-600 \leq x_i \leq 600, i = 1, 2, \ldots, d.$$

Global optimum $x_i^* = 0.0$ for $i = 1, \ldots, d$, $F(\mathbf{x}^*) = 0.0$

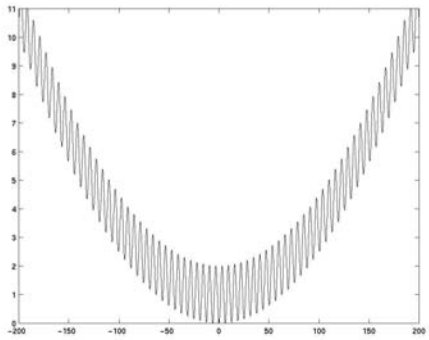


Figure B.2: Griewank Function

Rosenbrock

$$F(\mathbf{x}) = \sum_{i=1}^{d-1} (100 \times (x_{i+1} - x_i^2)^2 + (1 - x_i)^2) \qquad \text{(Eq. B.3)}$$
$$-2.048 \leq x_i \leq 2.048, i = 1, 2, \ldots, d.$$

Global optimum $x_i^* = 1.0$ for $i = 1, \ldots, d$, $F(\mathbf{x}^*) = 0.0$

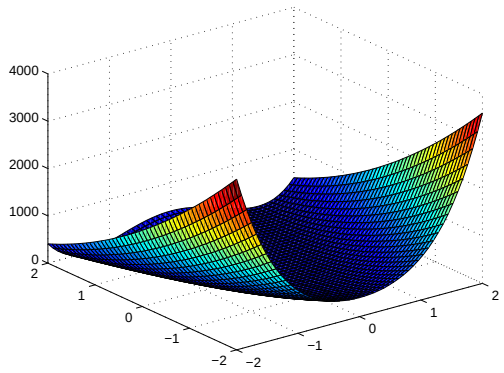


Figure B.3: Rosenbrock Function

Shifted Rotated Rastrigin

$$F(\mathbf{x}) = \sum_{i=1}^{d}(z_i^2 - 10cos(2\pi z_i) + 10) - 330 \qquad \text{(Eq. B.4)}$$

$$\mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M},$$

$$-5 \leq x_i \leq 5, i = 1, 2, \ldots, d.$$

Global optimum $\mathbf{x}^* = \mathbf{o}$, $F(\mathbf{x}^*) = f_{bias} = -330$.

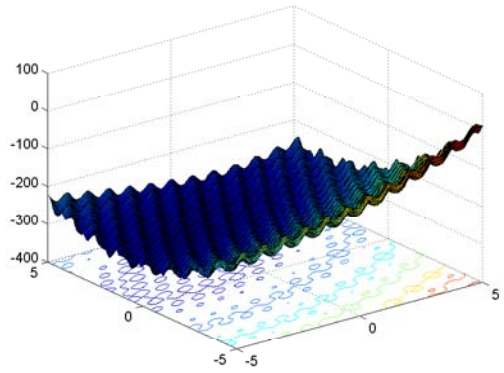


Figure B.4: Shifted Rotated Rastrigin Function

Shifted Rotated Weierstrass

$$F(\mathbf{x}) = \sum_{i=1}^{d}(\sum_{k=0}^{k_{max}}[a^k cos(2\pi b^k(z_i + 0.5))]) \quad \text{(Eq. B.5)}$$
$$-d\sum_{k=0}^{k_{max}}[a^k cos(2\pi b^k.0.5)] + 90$$
$$\mathbf{z} = (\mathbf{x} - \mathbf{o}) * \mathbf{M},$$
$$-0.5 \leq x_i \leq 0.5, i = 1, 2, \ldots, d.$$

Global optimum $\mathbf{x}^* = \mathbf{o}$, $F(\mathbf{x}^*) = f_{bias} = 90$. $a = 0.5$, $b = 3$, k_{max}=20.

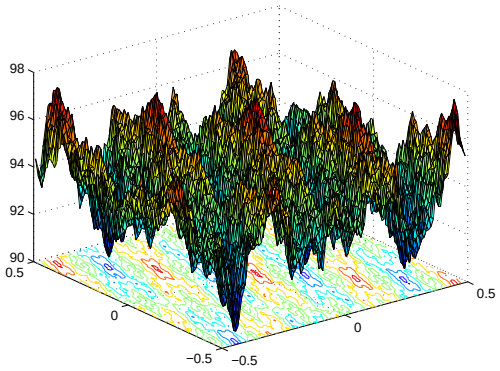


Figure B.5: Shifted Rotated Weierstrass Function

Shifted Expanded Griewank plus Rosenbrock

$$F(\mathbf{x}) = F_2(F_3(z_1, z_2)) + F_2(F_3(z_2, z_3)) + \ldots \quad \text{(Eq. B.6)}$$
$$+F_2(F_{ros}(z_{d-1}, z_d)) + F_2(F_3(z_d, z_1)) - 130$$
$$\mathbf{z} = \mathbf{x} - \mathbf{o} + \mathbf{1},$$
$$-3 \leq x_i \leq 1, i = 1, 2, \ldots, d.$$

Global optimum $\mathbf{x}^* = \mathbf{o}$, $F(\mathbf{x}^*) = f_{bias} = -130$

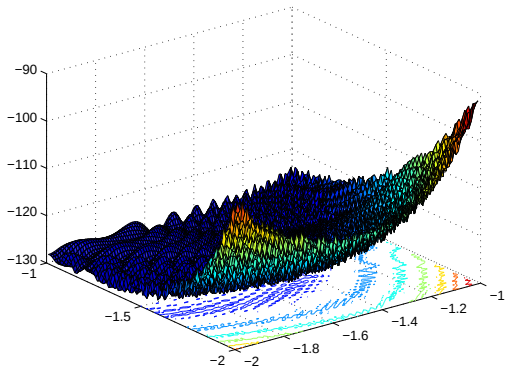


Figure B.6: Shifted Expanded Griewank plus Rosenbrock Function

Hybrid Composition Function

for $i = 1 : 10$ **do**

$w_i = exp\left(-\frac{\sum_{k=1}^{d}(x_k - O_{ik})^2}{2d\sigma^2}\right)$

$fit_i = f_i(((x - o_i)/\lambda_i) * \mathbf{M}_i)$

$fmax_i = f_i((y/\lambda_i) * M_i)$

$fit_i = C * fit_i/fmax_i$

end for

$SW = \sum_{i=1}^{10} w_i$

$MaxW = max(w_i)$

for $i = 1 : 10$ **do**

$w_i = \{w_i$ if $w_i = MaxW$

$w_i = w_i * (1 - MaxW^{10})$ if $w_i \neq MaxW$

$w_i = w_i/SW$

end for

$$F(\mathbf{x}) = \sum_{i=1}^{10}\{w_i * [fit_i + bias_i]\} \qquad \text{(Eq. B.7)}$$

$$F(\mathbf{x}) = F(\mathbf{x}) + f_{bias}$$

$f_{1-2}(\mathbf{x})$: Rastrigin Function

$f_{3-4}(\mathbf{x})$: Weierstrass Function

$f_{5-6}(\mathbf{x})$: Griewank Function

$f_{7-8}(\mathbf{x})$: Ackley Function

$f_{9-10}(\mathbf{x})$:Sphere Function

$F_{sphere} = \sum_{i=1}^{d} x_i^2$

$\sigma_i = 1$ for $i = 1, 2, \ldots, d$

$\lambda = [1, 1, 10, 10, 5/60, 5/60, 5/32, 5/32, 5/100, 5/100]$

bias=$[0, 100, 200, 300, 400, 500, 600, 700, 800, 900]$

$\mathbf{M}_i$ are all identity matrices

$C = 2000$

Global optimum $\mathbf{x}^* = \mathbf{o_1}$, $F(\mathbf{x}^*) = f_{bias} = 120$

$-5 \leq x_i \leq 5, i = 1, 2, \ldots, d.$

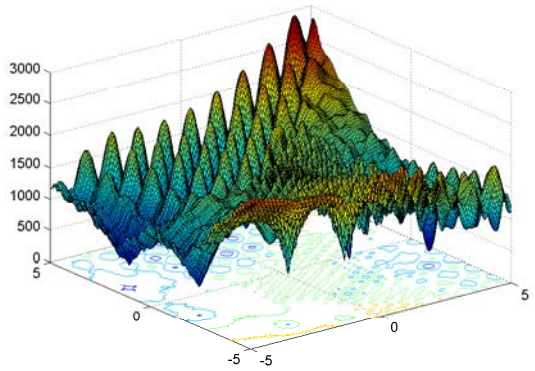


Figure B.7: Hybrid Composition Function

Rotated Hybrid Composition Function of F7

Same as Hybrid Composition Function above, except $\mathbf{M}_i$ are different linear transformation matrices with condition number of 2.

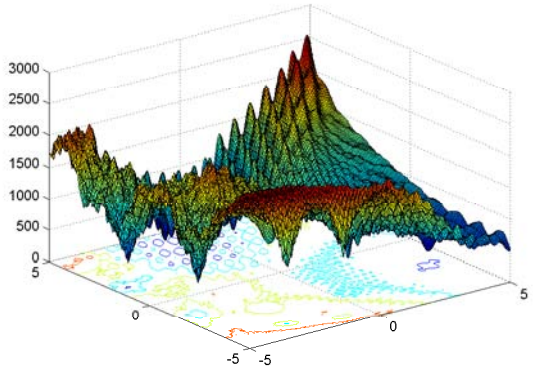


Figure B.8: Rotated Hybrid Composition function of F7

Rotated Hybrid Composition Function with Narrow Basin Global Optimum

$f_{1-2}(\mathbf{x})$: Ackley Function

$f_{3-4}(\mathbf{x})$: Rastrigin Function

$f_{5-6}(\mathbf{x})$: Sphere Function

$f_{7-8}(\mathbf{x})$: Weierstrass Function

$f_{9-10}(\mathbf{x})$:Griewank Function

$\sigma_i = [0.1, 2, 1.5, 1.5, 1, 1, 1.5, 1.5, 2, 2]$

$\lambda = [0.1 * 5/32, 5/32, 5/32, 2 * 1, 1, 2 * 5/100, 5/100, 2 * 10, 10, 2 * 5/60, 5/60]$

$\mathbf{M}_i$ are all rotation matrices. Condition numbers are [2 3 2 3 2 3 20 30 200 300]

Global optimum $\mathbf{x}^* = o_1$, $F(\mathbf{x}^*) = f_{bias} = 10$

$-5 \leq x_i \leq 5, i = 1, 2, \ldots, d.$

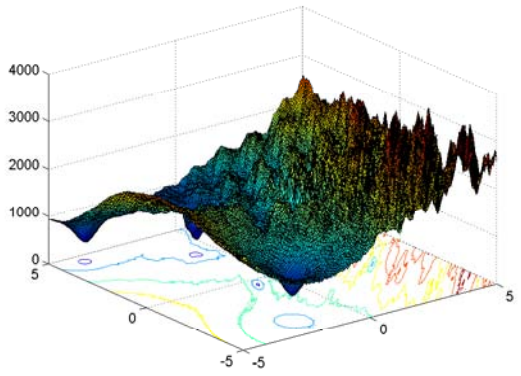


Figure B.9: Rotated Hybrid Composition Function with narrow basin global optimum

Non-continuous Rotated Hybrid Composition Function

$f_{1-2}(\mathbf{x})$: Rotated Expanded Schaffer Function
$f(x,y) = 0.5 + \frac{(sin^2(\sqrt{(x^2+y^2)})-0.5)}{(1+0.001(x^2+y^2))^2}$
$F_{schaffer}(\mathbf{x}) = f(x_1,x_2) + f(x_2,x_3) + \ldots + f(x_{d-1},x_d) + f(x_d,x_1)$
$f_{3-4}(\mathbf{x})$: Rastrigin Function
$f_{5-6}(\mathbf{x})$: F6 Function
$f_{7-8}(\mathbf{x})$: Weierstrass Function
$f_{9-10}(\mathbf{x})$:Griewank Function
$\sigma_i = [1,1,1,1,1,2,2,2,2,2]$
$\lambda = [5*5/100, 5/100, 5*1, 1, 5*1, 5*10, 10, 5*5/200, 5/200]$
$\mathbf{M}_i$ are all orthogonal matrix. Condition numbers are [2 3 2 3 2 3 20 30 200 300]
Global optimum $\mathbf{x}^* = o_1$, $F(x^*) = f_{bias} = 360$
$-5 \leq x_i \leq 5, i = 1,2,\ldots,d.$

$$x_j = \begin{cases} x_j & \text{if } |x_j - o_{1j}| < 0.5 \\ round(2x_j)/2 & \text{if } |x_j - o_{1j}| \geq 0.5 \end{cases}$$

$$round(\mathbf{x}) = \begin{cases} a-1 & \text{if } x \leq 0 \text{ and } b \geq 0.5 \\ a & \text{if } b < 0.5 \\ a+1 & \text{if } x > 0 \text{ and } b \geq 0.5 \end{cases}$$

where a and b are $\mathbf{x}$'s integral and decimal parts, respectively.

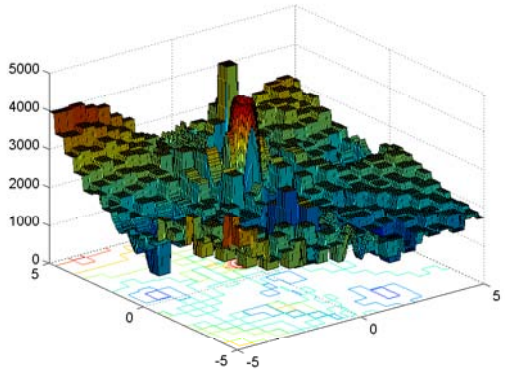


Figure B.10: Rotated Hybrid Composition Function with global optimum on the bounds

B.2 Multi-Objective Benchmark Functions

Multi-objective benchmark problems (MF1-MF6) [212] used in the thesis are summarized in this section. Parametric domain used is $[0,1]^d$, where d is the problem dimensionality considered.

Benchmark Function	**Formulation**	**Characteristics**
MF1 ($d=30$)	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})[1-\sqrt{f_1(\mathbf{x})/g(\mathbf{x})}]$ $g(\mathbf{x}) = 1+9(\sum_{i=2}^{d} x_i)/(d-1)$	Convex, 2-objective Pareto front
MF2 ($d=30$)	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})[1-f_1(\mathbf{x})/g(\mathbf{x})^2]$ $g(\mathbf{x}) = 1+9(\sum_{i=2}^{d} x_i)/(d-1)$	Non-convex, 2-objective Pareto front
MF3 ($d=50$)	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})[1-\sqrt{f_1/g}-(f_1/g)sin(10\pi f_1)]$ $g(\mathbf{x}) = 1+9(\sum_{i=2}^{d} x_i)/(d-1)$	Convex, disconnected, 2-objective Pareto front
MF4 ($d=50$)	$f_1(\mathbf{x}) = 1-\exp(-4x_1)sin^6(6\pi x_1)$ $f_2(\mathbf{x}) = g(\mathbf{x})[1-(f_1(\mathbf{x})/g(\mathbf{x}))^2]$ $g(\mathbf{x}) = 1+9[\sum_{i=2}^{d} x_i/(d-1)]^{0.25}$	Non-convex, 2-objective Pareto front
MF5 ($d=20$)	$f_1(\mathbf{x}) = cos(\frac{\pi}{2}x_1)cos(\frac{\pi}{2}x_2)(1+g(\mathbf{x}))$ $f_2(\mathbf{x}) = cos(\frac{\pi}{2}x_1)sin(\frac{\pi}{2}x_2)(1+g(\mathbf{x}))$ $f_3(\mathbf{x}) = cos(\frac{\pi}{2}x_1)(1+g(\mathbf{x}))$ $g(\mathbf{x}) = \sum_{i=3}^{d}(x_i-x_1)^2$	Non-convex, 3-objective, Pareto front
MF6 ($d=10$)	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})[1-\sqrt{f_1(\mathbf{x})/g(\mathbf{x})}]$ $g(\mathbf{x}) = 1+10(d-1)+\sum_{i=2}^{d}(x_i^2-10\cos(4\pi x_i))$	Convex, 2-objective, multiple local Pareto front

Appendix C

Real-world Problems

C.1 Drag/Lift Ratio of Aerodynamic Airfoil Design

The drag D and lift L on an airplane are the components of the total aerodynamic force parallel and vertical to the direction of flight, respectively, as shown in Fig. C.1.a. The importance of the ratio D/L in design can be understood, for example, in two airplane performance considerations [17]. First, the engine thrust required for level and unaccelerated flight, *i.e.*, cruise, is given by

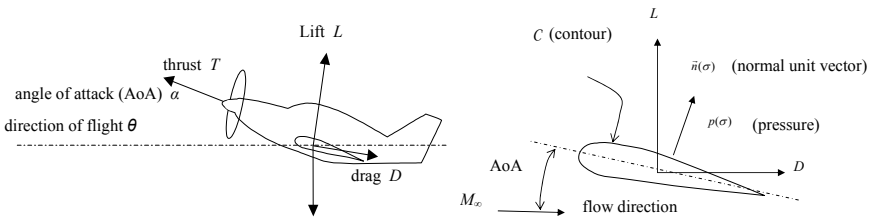
$$T_{cruise} = (\text{weight of aircraft}) \times D/L \qquad \text{(Eq. C.1)}$$

Second, an airplane in a power-off gliding flight will descent at an angle $\theta_{gliding}$ given by

$$\tan\theta_{gliding} = D/L \qquad \text{(Eq. C.2)}$$

In both cases, it is obvious that the smaller the ratio D/L, the better the performance. In the first case, a small ratio means less engine power is required for cruising flight, thus saving fuel. In the second case, low drag over lift entails a safer gliding flight in the case of engine failure. While the drag and lift forces on an airplane are determined by various body components, the contribution of the wings is dominant. This motivates the development of an approach for designing airfoil geometries by minimizing the D/L ratio.

In an airfoil shape-optimization problem using computation fluid dynamics, the drag and lift forces can be obtained by calculating the flow field around the airfoil under prescribed operating conditions, defined by the Mach number which represents the incident



C.1.a: Aircraft C.1.b: Airfoil

Figure C.1: (Forces acting on: (a) an aircraft, and (b) an airfoil (2D cross-section of the wing)

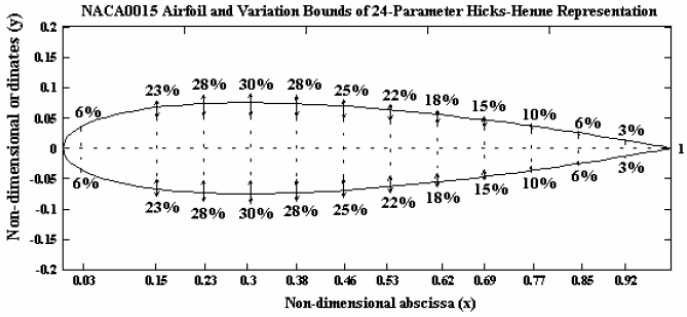


Figure C.2: Airfoil geometry characterized using 24 design variables with the NACA 0015 as baseline.

flow rate, and the angle of attack (see Fig. C.1.b). Ignoring friction, the flow is governed by the 2D Euler equations:

$$\frac{\partial w}{\partial t} + \frac{\partial f_1}{\partial z_1} + \frac{\partial f_2}{\partial z_2} = 0 \qquad \text{(Eq. C.3)}$$

with t as the time variable,

$$w = [\rho\ \rho u_1\ \rho u_2\ \rho E]^T,\ f_1 = [\rho u_1\ \rho u_1^2 + p\ \rho u_1 u_2\ \rho u_1 H]^T,$$
$$f_2 = [\rho u_2\ \rho u_1 u_2\ \rho u_2^2\ \rho u_2 H]^T \qquad \text{(Eq. C.4)}$$

where ρ is the density, u_1 and u_2 ar the flow velocity components in the Cartesian space with coordinates z_1 and z_2, p is the pressure, E is the total specific energy and H is the total specific enthalpy. Moreover, the pressure is given by $p = (\gamma - 1)\rho(E - \frac{1}{2}u_1^2 - \frac{1}{2}u_2^2)$, where γ is the specific heat [201].

Thus, the drag D and lift L are simply the components opposite the direction of flight $\vec{u}_\infty$, and the direction perpendicular to flight $\vec{\tau}_\infty$, respectively, of the resultant force due

to pressure acting along the contour C of the airfoil (see Figure C.1.b). They are given by the following integrals:

$$D = \oint_C p(\sigma)\vec{n}(\sigma).\vec{(u)}_\infty d\sigma$$
$$L = \oint_C p(\sigma)\vec{n}(\sigma).\vec{(\tau)}_\infty d\sigma \qquad \text{(Eq. C.5)}$$

Here, a single computation of the drag-to-lift ratio profile for typical airfoil geometry takes around 20 minutes on a Pentium III processor. The geometry of the airfoil is represented using 24-parameter Hicks-Henne functions as illustrated in Figure C.2 [75]. The lower and upper bounds on these parameters ($\mathbf{x} = x_1, x_2, \ldots, x_{24}$) are listed in Table C.1.

Table C.1: Lower and upper bounds of the design parameters in the airfoil problem.

Parameter (x_i)	Lower Bound (x_i^l)	Upper Bound (x_i^u)
x_1	-0.005000	0.007500
x_2	-0.007000	0.005000
x_3	-0.007000	0.005000
x_4	-0.005000	0.005000
x_5	-0.005000	0.002500
x_6	-0.003000	0.001500
x_7	-0.003000	0.001500
x_8	-0.003000	0.001500
x_9	-0.002500	0.001250
x_{10}	-0.002000	0.001250
x_{11}	-0.002000	0.000625
x_{12}	-0.001000	0.000500
x_{13}	-0.007500	0.005000
x_{14}	-0.005000	0.007000
x_{15}	-0.005000	0.007000
x_{16}	-0.005000	0.005000
x_{17}	-0.002500	0.005000
x_{18}	-0.001500	0.003000
x_{19}	-0.001500	0.003000
x_{20}	-0.001500	0.003000
x_{21}	-0.001250	0.002500
x_{22}	-0.001250	0.002000
x_{23}	-0.000625	0.002000
x_{24}	-0.000500	0.001000

Note that the Hicks-Henne representation describes the upper and lower surfaces of the airfoil as linear combinations of a finite number of basis functions. The free-stream condition in this problem are subsonic speed of Mach 0.5 and AOA=2.0 is used for this problem. An example on optimizing the same problem in the context of evolutionary robust engineering design can also be obtained from [144].

C.2 Inverse Pressure of Aerodynamic Airfoil Design

In the inverse design problem, the aim is to minimize the difference between the surface pressure P of a given airfoil with the desired pressure profile P_d of a baseline shape. If W is the flow variables and S the shape design variables, the inverse pressure design problem can be formulated as a minimization problem of the form:

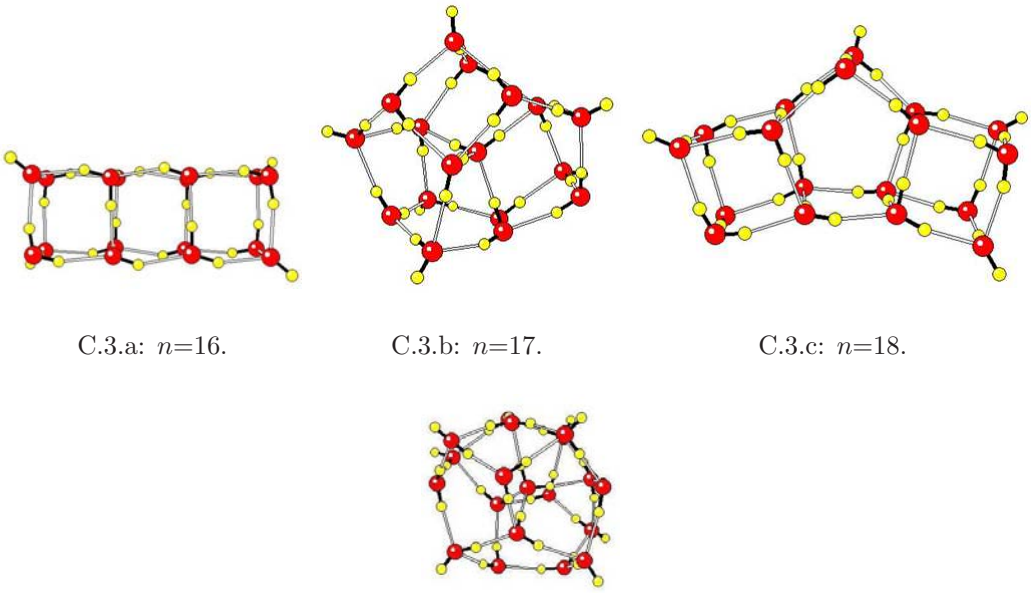
$$f(W,S) = \frac{1}{2}\int_{wall} (P - P_d)^2 \, d\sigma \qquad \text{(Eq. C.6)}$$

The target pressure profile is generated from the NACA 0015 airfoil[1] as the baseline shape. The airfoil geometry used here is characterized using 24-parameter Hicks-Henne representation as described in Fig. C.2. The lower and upper bounds on these parameters ($\mathbf{x} = x_1, x_2, \ldots, x_{24}$) listed in Table C.1 are also used for this problem. Since we only consider compressible non-viscous flow, a finite-volume Euler solver with body-fitted grid and explicit time-stepping is employed for the purpose of this study. Similar to the aforementioned ratio airfoil problem, the free-stream conditions in this problem are also subsonic speed of Mach 0.5, and 2.0 angle of attack (AOA).

C.3 Water Cluster Structure

Here, we provide details on the water cluster structural optimization problem used throughout this thesis. Neutral water clusters (see Fig. C.3) have been investigated extensively for a long period because they provide important understanding of properties of water molecules in aqueous media. In this particular project, our aim is to find low-energy structures. Several models have been developed for estimating the interaction energies and to reproduce the ground state structures of first principle calculation. To date, many researchers have focused on using the global minima to validate and compare the different potential models. Lee *et al.* applied simulated annealing method with the potential function of Cieplak, Kollman, and Lybrand to optimize water clusters up to n=20 [76]. In this study, we experimented with two sophisticated, flexible potential models, Thole-Type Model 2.1-F (TTM2.1-F) and Ojamae-Shavitt-Singer 2 (OSS2).

[1]The NACA 0015 is one of the four-digit series airfoils. The first two digits '00' implies the none existence of camber. The next two digits, *i.e.*, '15' indicates a 15% thickness to chord length ratio.



C.3.a: n=16. C.3.b: n=17. C.3.c: n=18.

C.3.d: n=19.

Figure C.3: Ground state structures of pure water clusters (for n=16-19).

TTM2-F is a flexible, polarizable, Thole-type interaction potential developed by Burnham and coworkers [30]. Although the model was parameterized using water dimer only, it was shown to reproduce the binding energies that are in close agreement to MP2 calculations for $(H_2O)_n$ $(n = 2-6)$ and $(H_2O)_{20}$. In this study, we first considered the TTM2.1-F model, which is a revised version of the original TTM2-F, that reportedly resolves issues relating to dipole moment of individual water and short intermolecular interaction.

The second sophisticated model considered in this study is the OSS2 potential [135]. It was developed to describe water as participant in ionic chemistry. Since it is an atomic and polarizable potential, OSS2 is useful for studying proton-transfer reaction and water disassociation. While originally designed for $H + (H_2O)_n$, it was later demonstrated by some researchers to model small-sized pure water clusters well [121].

Appendix D

List of Publications

Journal Papers

- **Dudy Lim**, Yaochu Jin, Yew Soon Ong, and Bernhard Sendhoff. Generalizing Surrogate-assisted Evolutionary Computation. *IEEE Transactions on Evolutionary Computation*. In Press. **Impact Factor: 3.736(ISI).**

- **Dudy Lim**, Yew Soon Ong, Yaochu Jin, Bernhard Sendhoff, and Bu Sung Lee. Efficient Hierarchical Parallel Genetic Algorithms Using Grid Computing. *Future Generation Computer Systems - The International Journal of Grid Computing Theory, Methods, and Applications*. Vol. 23, No. 4, pp. 658-670. Elsevier. May 2007. **Citation: 11(ISI), 17(Scopus), 30(Google Scholar). Impact Factor: 1.476(ISI).**

- **Dudy Lim**, Yew Soon Ong, Yaochu Jin, Bernhard Sendhoff, and Bu Sung Lee. Inverse Multi-Objective Robust Evolutionary Design. *Genetic Programming and Evolvable Machines*. Vol 7, No. 4, pp. 383-404. Springer. December 2006. **Citation: 6(Scopus), 4(Google Scholar). Impact Factor: 0.46(Citeseer).**

Patent

- A Dual Surrogate Memetic Framework for Single/Multi-Objective Evolutionary Optimization of Computationally Expensive Problems, Filed Date: 11 Oct 2007. Application No./Patent No. : 07118276.0-1225.

Conference Papers

- **Dudy Lim**, Yew Soon Ong, Yaochu Jin, and Bernhard Sendhoff. Evolutionary Optimization with Dynamic Fidelity Computational Model. *Proceedings of International Conference on Intelligent Computing*, LNAI 5227, pp. 235-242. Shanghai, China. September 2008.

- **Dudy Lim**, Yew Soon Ong, Yaochu Jin, and Bernhard Sendhoff. A Study on Metamodeling Techniques, Ensembles, and Multi-Surrogates in Evolutionary Computation. *Proceedings of Genetic and Evolutionary Computation Conference*, pp. 1288-1295, London, UK. July 2007. **Citation: 1(Scopus), 4(Google Scholar).**

- **Dudy Lim**, Yew Soon Ong, Yaochu Jin, and Bernhard Sendhoff. Trusted Evolutionary Algorithm. *Proceedings of IEEE Congress on Evolutionary Computation*. pp. 149-156. Vancouver. July 2006. **Citation: 4(Google Scholar).**

- Yew Soon Ong, Zongzhao Zhou, and **Dudy Lim**. Curse and Blessing of Uncertainty in Evolutionary Algorithm Using Approximation. *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 2928-2935. Vancouver. July 2006. **Citation: 5(Scopus), 8(Google Scholar).**

- **Dudy Lim**, Yew Soon Ong, and Bu Sung Lee. Inverse Multi-Objective Robust Evolutionary Design Optimization in the Presence of Uncertainty. *Proceedings of Workshops on Genetic and Evolutionary Computation Conference*, pp. 55-62. Washington D.C., USA. June 2005. **Citation: 13(Google Scholar).**

- Hee Khiang Ng, **Dudy Lim**, Yew Soon Ong, Bu Sung Lee, Lars Freund, Shuja Parvez, and Bernhard Sendhoff. A Multi-Cluster Grid Enabled Evolution Framework for Aerodynamic Airfoil Design Optimization. *Proceedings of International Conference on Natural Computation*, LNCS 3611, pp. 1112-1121. Changsha, China. August 2005. **Citation: 3(ISI), 5(Scopus), 12(Google Scholar).**

- Zongzhao Zhou, Yew Soon Ong, My Hanh Nguyen, and **Dudy Lim**. A Study on Polynomial Regression and Gaussian Process Global Surrogate Model in Hierarchical Surrogate-Assisted Evolutionary Algorithm, *Proceedings of IEEE Congress on*

Evolutionary Computation, pp. 2832-2839, Edinburgh, United Kingdom. September 2005. **Citation: 2(Scopus), 2(Google Scholar).**

Book Chapters

- **Dudy Lim**, Yew Soon Ong, Meng Hiot Lim, and Yaochu Jin. Single/multi-objective inverse robust evolutionary design methodology in the presence of uncertainty. *Evolutionary Computation in Dynamic and Uncertain Environments*, Studies in Computational Intelligence (SCI) 51, pp. 437-456. Springer. 2007. **Citation: 1(Google Scholar).**