The University Of Sheffield.

Investigations in multi-objective evolutionary algorithms for design optimization workflows

Submitted July 2025, in partial fulfillment of
the conditions for the award of the degree **MPhil - Automatic Control and Systems Engineering.**

Daniel C. Oara

Supervised by Professor Robin Purshouse

Department of Automatic Control and Systems Engineering
The University of Sheffield

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature _____
Date 22 /07/2025

Acknowledgements

To my supervisor Professor Robin Purshouse I would like to show my profound and warmest gratitude. He provided me with continuous support and guidance throughout this time. Without his help this work wouldn't have reached its current stage. I would also like to show my sincere appreciation to Joao Duro who has been a point of contact in ACSE lab, he was always willing to help me and point me into right direction.

I would like to thank to all my friends from Sheffield University, and finally I owe my heartfelt appreciation to my family for always being there to support me along this journey.

Abstract

In the past decade, complex engineering systems have seen increased development towards computer simulations and visualization. This trend is driven by the growing capabilities and availability of information technology resources. As these systems developed, it is now possible to solve difficult problems that could not be solved in the past.

When approaching problems using virtual engineering design, the optimization stage is important step as most of the times the modelling stage still relies on costly black-box simulations. Real-world applications often involve multiple objectives that must be optimized simultaneously, and these objectives are frequently in conflict with one another. It is important to identify which algorithm performs best for a given set of problem features, while considering the constraint of a limited evaluation budget. Another important question is which optimization software is better to be used by practitioners when attempting to solve these type of optimization problems.

In order to address these questions, it is first essential to discuss the algorithms available for solving multi-objective optimization problems. A new optimization software named Liger is used in this work, this software shows promising capabilities and can be used easily by non-experts in optimization to obtain a good approximation set of optimal design on the true Pareto front. This work focusses on the surrogate based optimization algorithms in Liger and a benchmarking procedure for testing these algorithms. To simulate real world problem features a set of test problems based on Walking Fish Group (WFG) framework are used. The user can adjust the uncertainty in either the radial or perpendicular direction, or simultaneously in both directions. This thesis also demonstrates how an indicator-based multi-objective optimization algorithm from the literature can be easily implemented within the Tigon library. The results obtained using Liger are validated against the ones published in the literature.

The benchmarking framework established in this research was designed to compare stochastic surrogate-based optimization algorithms — specifically, the sParEGO algorithm — with simpler variants of the ParEGO algorithm for robust multi-objective optimization problems. Although sParEGO was specifically designed as a hybrid algorithm to handle uncertainties and provide optimal robust solutions, it will be shown that in some cases, depending on the features of the stochastic problem, the Monte Carlo (MC) version of ParEGO can achieve faster convergence toward the Pareto front and deliver competitive performance in terms of robust solutions for the decision maker.

Contents

1 Introduction **1**

1.1 Background and Motivation 1

1.2 Aims and Objectives 3

1.3 Research gaps and contributions 3

1.4 Description of the thesis 4

2 Literature Review **5**

2.1 Optimization 5

2.2 Problem formulation 6

2.2.1 Multi-objective deterministic optimization 6

2.2.2 Multi-objective robust optimization 6

2.2.3 Constraint handling 9

2.3 Multi-objective optimization algorithms 12

2.3.1 Pareto-based methods 12

2.3.2 Indicator-based methods 13

2.3.3 Decomposition-based methods 13

2.4 Methods for robust optimization 14

2.5 Black-box optimization 16

2.6 Research gap 21

3 Multi-objective optimizer design in Tigon and Liger **23**

3.1 Introduction 23

3.2 The concept and architecture 24

3.3 Tigon library 25

3.4 Related work 27
3.5 Implementation of SMS-EMOA algorithm in Tigon library 29
3.6 Results 31

4 Comparison of sParEGO with simpler versions of ParEGO algorithm for robust multi-objective optimization 37

4.1 sParEGO optimization algorithm 37
4.2 CODeM Test problems 41
4.3 Results 46
4.4 Discussion 60

5 Conclusion and Future Work 61

5.1 Ideas for future work 63

REFERENCES 63

List of Figures

1.1 The virtual engineering workflow. 2

2.1 Robust system design of a black-box optimization framework. The system will account for three types of uncertainties: (1) The operating conditions, (2) Tolerance of design parameters, (3) Observed system performance. 17

2.2 Interpolation of the evaluated points, presented in the figure with black and the Gaussian process model (blue line).The red line is the estimated error. The grey point are the new estimated sampled points where the EI is maximum. (Source:WP1 M4 (2015)) 18

2.3 Objective Decomposition in Reference Directions. 19

3.1 Liger high level architecture. 24

3.2 Interaction between GUI and Tigon library. (Duro et al. 2021) 25

3.3 The relationship between clases in Tigon. (Duro et al. 2020) 26

3.4 A optimization workflow in Liger GUI. 27

3.5 Visualization nodes in Liger GUI. 27

3.6 The SMS-EMOA algorithm workflow. 31

3.7 Pareto front approximated by the SMS-EMOA algortihm for the ZDT1 function . 33

3.8 Pareto front approximated by the SMS-EMOA algortihm for the ZDT2 function . 33

3.9 Pareto front approximated by the SMS-EMOA algortihm for the ZDT4 function . 34

3.10 Pareto front approximated by the SMS-EMOA algortihm for the DTLZ1 function 34

3.11 Pareto front approximated by the SMS-EMOA algortihm for the DTLZ2 function. 35

4.1 Transformations function in CODeM toolkit.(Salomon, Purshouse, Giaghiozis & Fleming 2016) 43

4.2 Transformations function.(Salomon, Purshouse, Giaghiozis & Fleming 2016) . 43

4.3 CODeM1 median run Hypervolume indicator. 46
4.4 CODeM1 median run non dominated population. 46
4.5 CODeM1 median attainment surfaces. 47
4.6 CODeM1 EAF plots for each individual algorithm. 48
4.7 CODeM2 median run Hypervolume indicator. 49
4.8 CODeM2 median run non dominated population. 49
4.9 CODeM2 median attainment surfaces. 49
4.10 CODeM2 EAF plots for each individual algorithm. 50
4.11 CODeM3 median run Hypervolume indicator. 51
4.12 CODeM3 median run non dominated population. 51
4.13 CODeM3 median attainment surfaces. 51
4.14 CODeM3 EAF plots for each individual algorithm. 52
4.15 CODeM4 median run Hypervolume indicator. 53
4.16 CODeM4 median run non dominated population. 53
4.17 CODeM4 median attainment surfaces. 53
4.18 CODeM4 EAF plots for each individual algorithm. 54
4.19 CODeM5 median run Hypervolume indicator. 55
4.20 CODeM5 median run non dominated population. 55
4.21 CODeM5 median attainment surfaces. 56
4.22 CODeM5 EAF plots for each individual algorithm. 57
4.23 CODeM6 median run Hypervolume indicator. 58
4.24 CODeM6 median run non dominated population. 58
4.25 CODeM6 median attainment surfaces. 58
4.26 CODeM6 EAF plots for each individual algorithm. 59

List of Tables

3.1 Results of several EMOA on ZDT1, ZDT2 and ZDT4 test functions. 32

3.2 Results of several EMOA on DTLZ1 and DTLZ2 test functions. 36

3.3 *p*-values obtained by the two-sample *t* test. The *p* value (significance level of 5%) indicates rejection of the null hypothesis that two samples being compared have equal medians. A *h* value of zero supports that there is not enough evidence to reject the null hypothesis. 36

4.1 Hypervolume indicator results for CODeM test problems. 45

4.2 *p*-values obtained by the two-sample *t* test. The *p* value (significance level of 5%) indicates rejection of the null hypothesis that two samples being compared have equal medians. A *h* value of zero supports that there is not enough evidence to reject the null hypothesis. 56

Chapter 1

Introduction

1.1 Background and Motivation

This thesis examines the effectiveness of surrogate based optimization algorithms in Liger optimization software. Bayesian optimization algorithms (BOAs) have become increasingly popular in solving multi-objective optimization problems. This is because many real-world multi-objective optimization problems have complex and expensive (to evaluate) objective functions. To reduce the high computational cost the BOAs use surrogate models such as Gaussian processes, radial basis functions or neural networks to approximate these expensive evaluations. Before continuing, it is helpful to outline how optimization is integrated into the broader engineering design process.

The engineering design stage is a sequence that an engineer follows to develop a functional product and find the designs that maximise the client's requirements. This is a decision making process where basic science, mathematical and engineering knowledge is applied optimally to fulfil the objectives. Modelling is a critical stage in the engineering design process and it can be accomplished by either prototyping or computer simulation. Various factors determine which of these methods is more suitable for a given project. Most of the times the decision maker will opt in favour of developing a computer simulation due to its advantage of re-usability and the fact that modifications can be made at a relatively lower cost. Although computer models offer significant adaptability, they often come with the drawback of being expensive to develop or to run. A diagram of virtual engineering design framework is presented in Figure 1.1.

Purpose of optimization and robust optimization for engineering design

Optimization plays a key part in reducing the cost of testing and implementation. Currently, the real world optimization problems have a higher associated computational cost to evaluate the candidate design this is mainly due to expensive high-fidelity simulation models. As an example, for complex problems the cost of running a single evaluation can take hours to compute on a

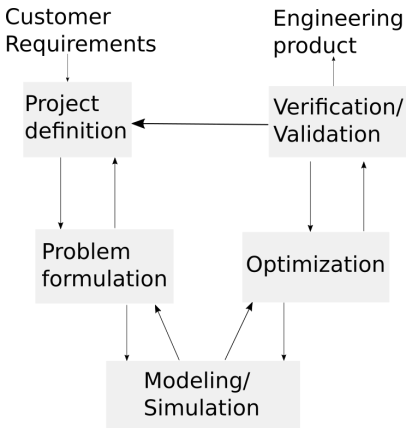


Figure 1.1: The virtual engineering workflow.

workstation. Hence the evaluation budget may be limited (small) during the optimization phase suggesting that the current multi-objective evolutionary algorithms will struggle to produce good results in optimum time.

For complex engineered products (such as in automotive and aerospace industry) optimization specifically the robust optimization is an important part of the engineering design. If in a real world application (e.g. design of a powertrain) the optimization process is implemented in the classical form, the system can be sensitive to noise and perturbations. Hence, a better design is the one where robustness has been taken into account. Salomon et al. (2016) states that a good robust performance design is the one where the required performance of the system is met without violating any constraints even in the presence of variations and uncertainties.

Role of virtual engineering in engineering design

Virtual engineering is the process where the complex engineered products are guided by computer models and simulations. These tools are used to estimate the performance of any design with objectives and constraints. When working on real world problems, the model or simulation can be considered as a black-box which is not easily amenable to analytical methods. Here a key role of virtual engineering is to make sure that the appropriate optimization framework is applied to solve the problem and visualise where the optimal solutions might occur.

Challenges of optimization in virtual engineering

When designing or redesigning a real world system one can be faced with numerous challenges such as availability and accuracy of the model and the required computational time. Majority of the time the initial models have low speed of interaction, complex features or low fidelity. The optimization techniques can be useful to improve and address these issues to enhance the performance of the overall system.

Through the process *robust design optimization* optimal robust solutions can be identified that can account not only for the perturbations but also for the uncertainties in the input decision variables. The design parameters may represent a portion of the overall system with certain assumptions about the optimal operating conditions of other interconnected subsystems. (Beyer & Sendhoff 2007*a*)

Kalsi et al. (1999) states that the virtual engineering technique in a complex system design is highly beneficial as multidisciplinary teams independently optimize their subsystems with limited information of the other team's subsystem(s). Simulations and surrogate models are used to apply these techniques in practice and approximate the objective functions. This helps in reducing the higher evaluation costs associated with these tasks.

1.2 Aims and Objectives

The aim of this thesis is to benchmark surrogate based algorithms from Liger optimization software and compare Liger with other optimization platforms available to industry practitioners in order to see if Liger is easier to use by non-specialists in optimization. The aim can be further subdivided into the following objectives:

- Critically compare Liger with other available workflow optimization software.
- Design, implement and test an existing indicator based multi-objective optimization algorithm in Tigon library.
- Create a benchmarking framework for the stochastic problems to compare sParEGO optimization algorithm with Monte Carlo based alternatives.

1.3 Research gaps and contributions

In comparison to the other available optimization platforms, Liger is easy to understand and use. Liger's capability to run on different operating systems and user friendly interface allows non-specialists in optimization to use state of the art algorithms to solve problems. The interactive decision-making capabilities makes it even more attractive to industry professionals. A key contribution to Tigon library is the implementation of an indicator based multi-objective optimization algorithm. This boosts the diversity of available algorithms in the current library in Liger. Researching the state of the art literature SMS-EMOA algorithm has been chosen due to its performance. The key contribution towards the field of optimization is the benchmarking of MC-ParEGO and sParEGO optimization algorithms on stochastic multi-objective optimization problems. The performance assessment of both algorithms will provide the scientific community and people in industry a framework from which they can decide which algorithm is best to use, to obtain Pareto optimal solutions, when dealing with complex optimization problems.

1.4 Description of the thesis

This section provides an overview of the thesis structure, first with a brief summary, followed by a more detailed explanation. Chapter 2 presents an overview of the optimization literature relevant to this research. This is broken down into stages to discuss types of problems and different algorithms currently used in practice. Chapter 3 shows a comparison between Liger and other optimization software followed by an implementation of the SMS-EMOA optimization algorithm in Tigon. Chapter 4 introduces a benchmarking analysis between sParEGO and MC based ParEGO optimization algorithms on a set of stochastic problems. The conclusion of this thesis is presented in Chapter 5 along with the exploration of potential directions for future research.

- Chapter 2 presents the literature review relevant to multi-objective optimization including stochastic problems that are relevant to real world optimization problems. This chapter starts by looking into the problem formulation when dealing with optimization problems. We then discuss about the methods for robust optimization, concluding with a review of the expensive multi-objective optimization.

- Chapter 3 presents the Liger optimization platform starting with an introduction into the concept and the architecture of the software. Moving forward Tigon library is explained and compared with relevant related optimization software. The chapter then shows how intuitive it is to implement SMS-EMOA optimization algorithm in Tigon library. The results obtained using Liger framework are then compared with the ones in Emmerich et al. (2005) paper.

- Chapter 4 introduces a benchmarking framework for testing MC based versions of ParEGO optimization algorithms and sParEGO algorithm on a set of stochastic multi-objective optimization problems. The chapter starts by introducing the implementation settings of the algorithms and the design options along with the performance metrics used. The results of the problems CODeM 1 to 6 are discussed along with the statistical significance.

- Chapter 5 summarises the key findings and discussions before moving onto the future research work.

Chapter 2

Literature Review

The virtual design approach extends across many research fields. In this work, the literature review concentrates on the optimization side of the design. While other areas of the virtual design, such as modelling, decision-making, and system formulation, contribute to the overall process, they are not part of the scope of this study. To reiterate, the focus of this review is on computationally expensive problems, aiming to discuss which optimization methods can provide better results.

The chapter starts by looking into the problem formulations and associated algorithms moving on to the black-box optimization. There will be a discussion on surrogate models and how efficient they are in solving stochastic multi-objective problems. Furthermore, a critical appraisal of the existing algorithms found in the literature to solve the proposed problems is also discussed.

2.1 Optimization

Starting with a model of the general system, the first task in optimization is to find a design, x, that satisfies the targeted objective. This design may or may not be optimal. Optimality refers to the design that maximizes the absolute performance. Beyer & Sendhoff (2007*a*) has described *optimization* the process of finding the right design parameters which optimize the objective function. The general formulation for an optimization problem is conventionally framed as a minimization problem, one of which can be noticed in Equation 2.1:

$$\min_{x} \quad f(\mathbf{x}) \tag{2.1}$$

where $\boldsymbol{x} = \{x_1, ..., x_n\}$ is a vector of decision variables in a domain $x_{min} \leq \boldsymbol{x} \leq x_{max}$ and $f(\boldsymbol{x})$ is the objective function. This function, $f(\boldsymbol{x})$, can be approximated by verbal descriptions, mathematical model, physical model or simulation. When noise or perturbations are not taken into account while developing the model the optimization process is called deterministic optimiza-

tion but when these are considered it becomes stochastic optimization. Stochastic optimization problems refers to the majority of the real world problems.

2.2 Problem formulation

2.2.1 Multi-objective deterministic optimization

The optimization problem of real world applications normally has more than one objective. Therefore, the system is modelled using a standard multi-objective optimization framework as follows:

$$\begin{aligned} \min_{x\in\Omega} \quad & \boldsymbol{f}(\mathbf{x}) = [f_1(\mathbf{x}), ..., f_m(\mathbf{x})] \\ \text{s.t.} \quad & g_i(\mathbf{x}) \leq 0, \quad i = 1, ..., I \\ & h_j(\mathbf{x}) = 0, \quad j = 1, ..., J \end{aligned} \tag{2.2}$$

here $\boldsymbol{f}$ is a performance vector formed by $f_k, k = \{1, ..., m\}$ objective functions, $\boldsymbol{x}$ is a vector of decision variables with $\boldsymbol{x} = \{x_1, ..., x_n\}$ in a domain $\Omega \subset \mathbb{R}^n$, $g_i(\mathbf{x})$ represents the set of inequality constraints and $h_j(\mathbf{x})$ represents the set of equality constraints. When solving the multi-objective optimization problem there is no a single solution which can optimize simultaneously all of the objectives, instead there is a set of trade-off solutions called Pareto set or Pareto optimal solutions. (Salomon, Avigad, Purshouse & Fleming 2016)

A solution $\boldsymbol{x} \in \Omega$ is Pareto optimal if and only if there is no solution $\boldsymbol{x}' \in \Omega$ for which the vector $\boldsymbol{u} = [f_1(\boldsymbol{x}), ..., f_m(\boldsymbol{x})]$ is dominated by the vector $\boldsymbol{v} = [f_1(\boldsymbol{x}'), ..., f_m(\boldsymbol{x}')]$. Vector $\boldsymbol{u} = [u_1, ..., u_m]$ dominate another vector $\boldsymbol{v} = [v_1, ..., v_m]$, $\boldsymbol{u} \preceq \boldsymbol{v}$, if and only if the statement: $\forall i \in \{1, ...m\}, u_i \leq v_i \wedge \exists i \in \{1, ..., m\} : u_i < v_i$ holds true. In other words if the vector $\boldsymbol{u}$ is no worse than the vector $\boldsymbol{v}$ in all objectives, but exist at least one objective from $\boldsymbol{u}$ which is better than the objective from the vector $\boldsymbol{v}$, then the vector $\boldsymbol{u}$ dominates the vector $\boldsymbol{v}$.

2.2.2 Multi-objective robust optimization

Taguchi (1986) was the first to design an optimization framework in three stages where uncertainties and perturbations were taken into account. The author is also known as, "the father of robust design", and his design framework is as follows :

- *Systems design*: here the general structure and the basic performance parameters are determined.
- *Parameter design*: is the stage to meet the required design performance, the optimization of the design parameters is also required.

- *Tolerance design*: is the step where a fine-adjusting of the design parameters is made after they are obtained from *Parameter design*.

The objective function of a real world application comprises of two types of parameters defined as: parameter $\mathbf{x}$ which is a control parameter and the noise factor $\mathbf{P}$ (e.g. environmental conditions such as temperature, pressure and production tolerances such as weight, length variations). These uncertainties make it more challenging to find the optimal operating point of a system. Taguchi (1986) work addresses this by accounting for performance variations more effectively than other optimization algorithms.(Beyer & Sendhoff 2007*a*)

The robust performance design can be defined as a method through which the required performances can be achieved by taking into account the uncertainties and perturbations. The formulation of a stochastic multi-objective optimization is as follows:

$$\begin{aligned} \min_{\mathbf{x}\in\Omega} \quad & \boldsymbol{f}(\mathbf{x},\mathbf{P}) = [f_1(\mathbf{x},\mathbf{P}),...,f_m(\mathbf{x},\mathbf{P})] \\ \text{s.t.} \quad & g_i(\mathbf{x},\mathbf{P}) \leq 0, \quad i = 1,...,I \\ & h_j(\mathbf{x},\mathbf{P}) = 0, \quad j = 1,...,J \end{aligned} \tag{2.3}$$

here $\boldsymbol{f}$ is a performance vector formed by $f_k, k = 1,...,m$ objective functions, $\mathbf{x}$ is a vector of decision variables with $\mathbf{x} = \{x_1,...,x_n\}$ in a domain $\Omega \subset \mathbb{R}^n$, $\mathbf{P}$ is an random variate vector, $g_i(\mathbf{x},\mathbf{P})$ is the uncertain set of inequality constraints and $h_j(\mathbf{x},\mathbf{P})$ is the uncertain set of equality constraints.(Salomon, Avigad, Purshouse & Fleming 2016)

The system random variates $\boldsymbol{f}(\mathbf{x},\mathbf{P})$ are evaluated by sampling the simulation, this will allow for the computation of the expected values, the standard deviation and the robustness indicator (e.g. six sigma, ninety percentile). In literature, methods such as Monte Carlo and Polynomial Chaos (PC) are used to compute the random variate.

Monte Carlo sampling is a statistical technique used to estimate numerical results by randomly sampling variables and observing the outcomes. It's widely applied in fields such as physics, finance, engineering, and machine learning to solve problems that might be deterministic in theory but are too complex to solve analytically. The first step in MC sampling is the random sampling where a large number of random inputs are generated using a specified probability distribution (e.g. uniform or normal). The second step is the evaluation of the identity function or model, this computes for each random variable a corresponding output. The final step is the aggregation of the results where the outputs are used to estimate the desired quantity such as the mean and variance.

Polynomial Chaos is a mathematical technique used for uncertainty quantification in computational models. It provides an efficient way to represent and propagate uncertainty in systems governed by differential equations. This method is particularly useful in engineering, finance, and physics, where model inputs are uncertain due to randomness or measurement errors. In essence PC represents a stochastic process as an expansion of orthogonal polynomials, these

polynomials serve as basis functions to approximate a random variable in terms of known probability distributions. The PC expansion of a stochastic function *f(X)* where *X* is a random variable, is given by:

$$f(X) \approx \sum_{i=1}^{N} c_i \Phi_i(X) \tag{2.4}$$

here c_i are deterministic coefficients to be computed,and $\Phi_i(X)$ are orthogonal polynomials associated with the probability distribution of X. When comparing PC with MC method we can say that PC has faster (exponential) convergence speed with a lower computational cost. The accuracy of MC simulations depend on sample size while PC has high accuracy for smoother functions. A disadvantage of PC is the complexity as it requires solving coefficients while MC is simple to implement.

To solve this type of stochastic problem a robustness metric indicator needs to be applied to the optimization problem. One of the approaches to handle robustness is to present the performance of a solution by computing the expected value as:

$$\min_{\mathbf{x}\in\Omega} \quad E[\boldsymbol{f}(\mathbf{x},\mathbf{P})] \tag{2.5}$$

where E is the expectation function computed over all evaluated objectives, the solution of this equation is the one that minimizes the mean performance.

Salomon et al. (2014) describes a common approach to handle robustness by minimizing the worst case performance as:

$$\min_{x\in\Omega} \arg\max_{\boldsymbol{P}} \quad \boldsymbol{f}(\mathbf{x},\mathbf{P}) \tag{2.6}$$

Ray et al. (2015) presented a solution to solve this type of robust multi-objective optimization through a six-sigma robust indicator. Considering only inequality constraints the algorithm solves the problem using a decomposition based method. The formulation of the problem can be expressed as:

$$\begin{aligned}
&\min_{(\mathbf{x},\mathbf{P})} \quad E[f_k(\mathbf{x},\mathbf{P})] \quad k = 1,...,m \\
&\max_{(\mathbf{x},\mathbf{P})} \quad f_{m+1}(\mathbf{x},\mathbf{P}) = \text{Min}(sigma_g, R_c) \\
&\text{s.t.} \quad sigma_g \equiv \text{Min}(\frac{E[g_i(\mathbf{x},\mathbf{P})]}{\sigma_{g_i(\mathbf{x},\mathbf{P})}}) \geqslant 0, \quad i = 1,...,I \\
&\qquad \mathbf{x}^{(L)} \leq \mathbf{x} \leq \mathbf{x}^{(U)}, \mathbf{P}^{(L)} \leq \mathbf{P} \leq \mathbf{P}^{(U)}
\end{aligned} \tag{2.7}$$

where: $\mathbf{x}$ is a vector of decision variables with $\mathbf{x} = \{x_1,...,x_n\}$, $\mathbf{P}$ is an n_p-dimensional vector random variate, $E[f_k(\mathbf{x},\mathbf{P})], k = 1,...,m$ are the expected values of the objective functions, $f_{m+1}(\mathbf{x},\mathbf{P})$ is the new objective function to account for constraints, $sigma_g$ is the ratio of the expected constraint value ($E[g_i(\mathbf{x},\mathbf{P})]$) and the standard deviation ($\sigma_{g_i(\mathbf{x},\mathbf{P})}$) of the constraint g, which measures how many standard deviations can fit between the constraint boundary and the given solution. R_c represents six-sigma quality ($R_c = 6$). This robust approach transforms the original objectives into new objectives where the focus is to minimize the mean of the original

objectives. An interesting fact is that the author solves the constraint problem by introducing the constraint as an extra objective, where the six-sigma quality is maximized.

2.2.3 Constraint handling

In real world the optimization problems encounter constraints, for example physical constraints such as: temperature, pressure, manufacturing costs and emissions, demanding consideration when building the system model. These constraints can be modelled as: inequality and/or equality constraints, box constraints, true constraints, soft constraints and hard constraints.

In the past few years evolutionary algorithms (EA) like Evolutionary Programming, Genetic Algorithm (GA), Evolutionary Strategies (ES) and Particle Swarm Optimization (PSO) have been applied successfully to solve the constrained multi-objective optimization problems. These algorithms offer flexibility and parallel search on the objective space (Engelbrecht 2007), occasionally this requires expensive function evaluations in order to find the optimal Pareto front.

One way to deal with the constraints in a multi-objective optimization problem is by transforming the problem into an unconstrained single-objective problem using a *weighted-sum* formulation presented below:

$$\begin{aligned} \min_{x\in\Omega} \quad & \sum_{i=1}^{m} \gamma_i \times f_i(\mathbf{x}) \\ & \sum_{i=1}^{m} \gamma_i = 1 \\ & \gamma_i > 0, \quad i = 1,...,m \end{aligned} \tag{2.8}$$

where $\boldsymbol{x}$ is a vector of decision variables with $\boldsymbol{x} = \{1,...,n\}$ in a domain $\Omega \subset \mathbb{R}^n$, $f_i(\boldsymbol{x}), i = \{1,...,m\}$ are the objective functions, and $\boldsymbol{\gamma}$ is the user defined weight vector, with the condition that the weights are strictly positive and the sum of all weights are equal to one. This creates a single objective function taking into account all the objectives. After this transformation any gradient based algorithm can be used to solve the problem.

Penalty based methods

Another method to handle constraints in the direct search community is by the use of penalties. Therefore the constrained-optimization problem is transformed into an unconstrained optimization problem. If the solution violates the constraints then a fitness value is added to the objective function based on this violation. The general formulation of such penalty function is given in:

$$\phi(\mathbf{x}) = f(\mathbf{x}) + \big[\sum_{i=1}^{I} r_i \times G_i + \sum_{j=1}^{J} c_j \times L_j\big] \tag{2.9}$$

here $\phi(\mathbf{x})$ is the new objective function, $f(\boldsymbol{x})$ is the unconstrained objective function, r_i and c_j are the penalty factors (positive constants), $G_i = max[0, g_i(\mathbf{x})]^{\beta}$ is a function based on inequality constraints $g_i(\mathbf{x})$ which maximizes the expected constraint satisfaction and $L_j = |h_j(x)|^{\gamma}$ is a function based on equality constraints $h_j(\mathbf{x})$, β and γ are constants usually with the value of 1 or 2. Coello Coello (2002) provide a comprehensive literature survey on constraints penalty methods.

Considering the minimum penalty rule, it is ideal to set the penalty value close to the threshold where feasible solutions can still be obtained. Significant deviations from this threshold can hinder the performance of direct search algorithms (e.g. EA), preventing them from delivering satisfactory results. Using a large penalty may overly restrict exploration, reducing the algorithm's ability to search the infeasible region at the start. Conversely, using a penalty that is too low risks wasting search time in the infeasible regions.

In many cases, the exact boundary between feasible and infeasible regions is unknown, making the implementation of the minimum penalty method challenging. Additionally, most problems where direct search algorithms are applied do not have constraints explicitly defined in algebraic form; instead, they are obtained from complex simulations.

Supplementary objectives

The idea is to transform a single objective constrained optimization problem into a multi-objective optimization problem (i.e the problem will have $I+1$ objective, where I is the number of constraints). Afterwards, any multi-objective optimization algorithm can be applied to solve the problem.

Surry & Radcliffe (1997) introduced the Constrained Optimization by Multi-Objective Optimization Genetic Algorithm (COMOGA), where the population is Pareto-ranked according to the constraint violation. The solution selection mechanism is based on either the fitness or tournament selection, depending on the rank. However, the results obtained with this method did not outperform those achieved using a penalty function.

Camponogara & Talukdar (1997) introduced a method where a single objective constrained problem is transformed in such a way that will result in two objective optimization problem. The first objective will be to optimize the original function and the second will be to minimize the equation:

$$f_{new}(\mathbf{x}) = \sum_{i=1}^{I} max[0, g_i(\mathbf{x})] \tag{2.10}$$

where $g_i(\mathbf{x})$ represents the inequality constraint. Runarsson & Yao (2005) proposed a similar method based on stochastic Pareto-ranking approach. Hernandez et al. (2004) proposed an approach named IS-PAES which is build up on Knowles and Corne (2000) Pareto Archive Evolution Strategy algorithm.

Dealing with constraints in optimization problem and treating them as objective is a promising future research path to follow. Depending on the problem this approach might be an efficient way, but the selection of a multi-objective algorithm to solve the problem needs to be made carefully. This is due to computational time the algorithm might have. The drawback of some of these algorithms is that they struggle with the population diversity, which is a common problem when using evolutionary multi-objective optimization techniques.

Feasibility preserving or restoring methods

Using EA for preserving and restoring feasibility is more refined than penalties but a major drawback is that requires special operators, the calculations are more complex when performing crossover and mutation. Sometimes, it might not be possible to transform an infeasible solution to a feasible one.

Coello Coello (2002) provides a history of how these algorithms came into practice. These algorithms use special builders and representations to solve real world optimization problems. One example is given in Davidor (1991) where a robot trajectory is built using a varying genetic algorithm which introduces a special crossover operator named *analogous crossover*. The crossover and mutation probability is selected based on a distribution error. Bean (1994) introduced ”random keys encoding” which is a special representation to eliminate the necessity of the crossover and mutation operators in certain types of optimization problems. This method solve the optimization problems using a low computational budget. Zbigniew (1996) developed an algorithm called Genetic algorithm for Numerical Optimization for Constrained Problems (GENOCOP) which simplify the search space for EA by removing the equality constraints together with an equal number of problem variables. Kowalczyk (1997) used a constraint consistent method to eliminate a portion of the search space by checking variables for feasible solutions. Michalewicz & Nazhiyath (1995) presents a repair algorithm named GENOCOP III where the two sets of population are kept to influence the solution evaluation between sets. First population set contains search points which satisfy the constraints and the second population contains the feasible reference points. Xiao et al. (1997) presented a repair algorithm which was able to reconstruct an infeasible path, using a specific set of genetic operators, for a robot which was moving from a point A to a point B in the presence of obstacles.

An advantage of using repair algorithms is that an infeasible solution can be transformed at a low computational cost in a feasible one. However, this might not be always possible. The drawback of this approach is that it will always be problem dependent and special repair methods need to be designed for each particular system.

2.3 Multi-objective optimization algorithms

By using the classical optimization methods the multi-objective optimization problem is transformed into multiple single objective problem where a single Pareto-optimal solution is obtained for each iteration. Focusing on optimizing a single objective at a time can most likely result in many optimal solutions being overlooked. Multi-objective evolutionary algorithms (MOEAs) can be more beneficial as multiple Pareto-optimal solutions for each iteration can be obtained. MOEA's can help to reduce the overall computational budget. The algorithms found in literature can be divided in three main types: Pareto based methods, Indicator based and Decomposition based methods.

2.3.1 Pareto-based methods

In the last decade, EA such as Non-dominated Sorting Genetic Algorithm II (NSGA-II) introduced by Deb et al. (2002), Multi-Objective Genetic Algorithm (MOGA) created by Murata et al. (1995) and the Niched Pareto Genetic Algorithm (NPGA) developed by Horn et al. (1994) gained more popularity in solving multi-objective optimization problems due to their Pareto optimality. The Pareto front is composed of solutions which have the property that they can not improve in one objective without worsening the others.

A key feature shared by these three algorithms is that the fitness is assigned to the population based on a non-dominating sorting criterion. Additionally, these algorithms ensure that solution diversity within a single non-dominated Pareto optimal front is maintained. Another important property of these algorithms is that they allow the overall performance os a system to be represented by a single value, one example is through Hypervolume Indicator. Due to the ever increasing complexity of the engineering design problems there is a need to conduct further research and development to be able to solve complex problems in the future.

NSGA-II algorithm is an extension of the initial NSGA method, developed by Srinivas & Deb (1994). The new method adopts the same fitness-sharing techniques as proposed in the NSGA approach. The initial population is sorted by using a non-dominating criterion where through the binary tournament selection (SBX crossover, and polynomial mutation) a child population is generated. Furthermore, a combined set of parent and child solutions undergoes another round of non-dominating sorting, after which the best non-dominated solutions are selected to form the new parent population. All these steps are repeated until the stopping criteria is met. Deb et al. (2002) algorithm demonstrated superior performance compared to approaches like Pareto Archived Evolution Strategy (PAES) algorithm and Strength Pareto Evolutionary Algorithm (SPEA) on a range of test problems.

2.3.2 Indicator-based methods

The evolutionary multi-objective algorithms (EMOA) are widely used in the Pareto optimization techniques due to their applicability on a range of problems. Another advantage is that their approach is based on the population, assuring that an inherent set of feasible solutions is obtained. Also, making sure that the required results are obtained when it becomes difficult to acquire a precise Pareto set through the analytical methods. The main idea is to optimize the value of a specific indicator (either by minimizing or maximizing it). In a multi-objective framework each point is ranked in the population according to the indicator. After ranking, a subset of the top-performing points is selected for improvement, typically through reproduction or other techniques. Over successive iterations, an optimal set of points guided by the chosen indicator is gradually constructed. (Zitzler & Künzli 2004)

Various quality indicators are employed to assess the effectiveness of Pareto optimal front approximations. Zitzler & Künzli (2004) proposed the *hypervolume measure* also called the $\mathscr{S}metric$ and is among the most important indicator measures used in an optimizer due to its measure of the dominated space. This indicator provides information about the convergent of the solutions towards the Pareto front and their distribution along the front.

Beume et al. (2007) developed a method based on $\mathscr{S}metric$ indicator called $\mathscr{S}$ Metric Selection Evolutionary Multi-objective Optimization Algorithms (SMS-EMOA). Bader & Zitzler (2011) introduced HypE algorithm that uses MC simulation to approximate the exact hypervolume values. The authors stated that the indicator values are of lesser importance in comparison to the rank of the solutions induced by this indicator.

SMS-EMOA was designed to use only a finite number of points in order to cover a maximum hypervolume. Using this method diminishes the issues related to the selection of the right reference point. This algorithms borrows ideas from other methods such as NSGA-II. The strategies deployed by Knowles for the archiving were also taken into consideration in the development of this algorithm. The ranking criteria used in this method is based on the non-dominated sorting which is one of the most important characteristics of this steady-state algorithm. Another distinguishing feature of this algorithm is that the solution which contributes with the least hypervolume value in the worst ranked front is discarded.

The benchmark of SMS-EMOA shows superior performance when compared with others EMOA's. HypE was also tested against NSGA-II and SPEA2, but also against IBEA, and showed that the mean performance was better.

2.3.3 Decomposition-based methods

Another method to obtain Pareto optimal solutions when dealing with multi-objective optimization is by using decomposition-based algorithms. The idea is to split the problem into a number of single objective sub-problems using scalarization/aggregation functions such as

Tchebycheff (also known as l_∞ norm):

$$f(x) = \max_{i=1}^{k}(\lambda_i f_i(x)), \tag{2.11}$$

and Weighted sum:

$$f(x) = \sum_{i=1}^{k}(\lambda_i f_i(x)), \tag{2.12}$$

Recently, this class of optimizers gained popularity when Zhang & Li (2007) introduced an algorithm named Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D). The algorithm works similarly like other EA but with two distinguishing features. The first one is the use of decomposition where a weighting selection is chosen to compute the sub-problems, and the second one is the introduction of neighbourhoods. These steps are taken at the beginning of the algorithm, where each weight is associated with a set of neighbouring weights. During the algorithm's execution, neighbourhoods are used in the reproduction stage (when a child solution is generated) and at the stage "Update of Neighbouring Solutions". This stage looks at each solution to compare and replace it with the newly discovered point if the weighting is superior. The MOEA/D offers the lowest computational complexity at each iteration in comparison to NSGA-II and has been demonstrated by experiments that it has the ability to outperform NSGA-II and other similar algorithms by the use of a simple decomposition method.

Hughes (2007) introduced the Multiple Single Objective Pareto Sampling (MSOPS) framework which compared with MOEA/D uses multiple scalarization functions as sometimes this can be more advantageous. The selection of scalarization function is important as it sets out the feature of the optimizer. Ishibuchi et al. (2006) introduced a weighted sum (l_1 norm) scalarization function into NSGA-II due to the slighter impact on the approach direction towards the Pareto front. Knowles (2006) uses a modified Tchebycheff aggregated function named augmented Tchebycheff, which a small ρ value is multiplied with l_1 norm to reduce the weakly dominated solution to enter the population.

2.4 Methods for robust optimization

There are several approaches to solve the robustness in the single-objective optimization problem, Taguchi (1986) was one of the first researchers to account for robust solutions based on the measure of deviation between the objective value of a sample and its expected value using DOE (i.e design of experiments). Other researchers (e.g Parmere (1996), Branke(1998),Stagge (2001),etc.) developed many approaches based on Taguchi's method to find robust solutions. Jin and Sendhoff (2003) solved the robust single-objective optimization problem by transforming it into multi-objective. Deb & Gupta (2005) took into account robustness by defining two types of robust solutions, robust solution of Type I where instead of the objective function the expected value of the objective function is used and robust solution of Type II where the worst case scenario is used. The main disadvantage of all these algorithms is that they only consider

unconstrained single-objective optimization problems. Beyer & Sendhoff (2007*b*) provides a comprehensive survey into robust optimization mentioning that this study is dedicated to reducing the impact of uncertainty.

Robust optimization is described as an optimization problem where $\boldsymbol{\omega}$ is the uncertainty of the inputs, parameters and structure of the system. These uncertainties cause the objective functions to give a different output every time they are computed. Hence the output will be substitute for the $I(f)$ robust indicator as in:

$$\max_{x \in X} I(\mathbf{x}, \boldsymbol{\omega}). \tag{2.13}$$

Rambeaux et al. (2000) introduces a method called *Threshold Probability* which assesses the likelihood that the objective function will exceed a reference threshold q as:

$$I_{con}(f(\mathbf{x}, \boldsymbol{\omega}), q) = p(f(\mathbf{x}, \boldsymbol{\omega}) \leq q). \tag{2.14}$$

Mourelatos & Liang (2006) introduced an indicator where a combination of possible values were obtained from the robust values using variance, expectancy or a mix in between. This is known as *Aggregated Value* and can be expressed as follows:

$$\begin{aligned} I_{exp}(\mathbf{x}, \boldsymbol{\omega}) &= E[f(\mathbf{x}, \boldsymbol{\omega})] \\ I_{var}(\mathbf{x}, \boldsymbol{\omega}) &= var[f(\mathbf{x}, \boldsymbol{\omega})] \end{aligned} \tag{2.15}$$

and the bi-objective problem to be optimized becomes:

$$\min_{x \in X} [I_{exp}, I_{var}]. \tag{2.16}$$

Ehrgott et al. (2014) developed the $Worst - case$ scenario that can be described in a bounded domain as:

$$I_{wc}(\mathbf{x}, \boldsymbol{\omega}_s) = \max_{\boldsymbol{\omega} \in \boldsymbol{\omega}_s} f(\mathbf{x}, \boldsymbol{\omega}). \tag{2.17}$$

Gaussian process

Rasmussen (2003) describes Gaussian processes as a powerful statistical method used in modelling and making predictions about the data in an uncertain system. Simply said a GP is defined by a set of random numbers with a determined mean and a kernel (covariance) function that follows a joint Gaussian distribution, as follows:

$$f(x) \sim \mathsf{GP}(m(x), k(x, x')) \tag{2.18}$$

here $m(x)$ is the mean and $k(x, x')$ is the kernel. GPs are applied to regression and Bayesian optimization, to construct a Gaussian process the set of points needs to be known, through experiments or evaluations, in order to compute the mean and covariance matrix. The more points that we can find through experiments or function evaluations of the true model the better a GP can approximate the real model by reducing the uncertainty.

Robust multi-objective optimization methods

After dealing with robustness in single-objective optimization problems, researchers focused on extending the methods for multi-objective optimization problems. Deb & Gupta (2006) extended their method of robust solutions of type I and II to multi-objective optimization problems by using NSGA-II to find robust Pareto-optimal solutions. To apply the robust indicator the algorithm needs to compute for each solution from fifty to a hundred times more MC samples. Therefore, the main drawback of using this type of method is the practicability, due to high computational time required to compute the Pareto optimal front.

Paenke et al. (2006) presented a method which efficiently estimates a solution expected value and its variance by constructing local approximation models of the objective function value (e.g. basis functions and polynomial chaos). Using model based existing estimation techniques the algorithm outperforms the implicit and explicit averaging approaches, by doing an efficient sampling. The disadvantage of the algorithm is that it requires an expensive budget (e.g. time due to number of iterations) to run the simulation.

Lim et al. (2006) presented a IMORE (i.e inverse multi-objective robust evolutionary) method where information about the desired robustness is used. Using a multi-level optimization search the algorithm manages the cluster of uncertain events by modelling a family of nest sets. Achieving robust performance solutions using DOE to reduce the evaluation budget. The drawback of the method is that it can not handle constraints and still requires a large number of function evaluations to locate the Pareto optimal solutions. Therefore, it will be computationally prohibited if problems with expensive objective functions are considered.

2.5 Black-box optimization

Constrained black-box optimization refers to a problem where the optimization algorithm needs to simulate and evaluate the function to get solutions without having the information about the gradient of the function. Usually, these are complex problems which are computationally expensive to be solved. Figure 2.1 presents a typical system where a robust design optimization needs to be implemented in order to solve a black box problem.

The general idea here is to build reduced order models, less complex estimators for more complex systems. These reduced order models are either used as substitute evaluations in the conventional method or they are sown in to the method itself.

Some optimizers work with lower fidelity (cheaper) evaluations this refers to *surrogate modelling*. The types of surrogate models are for example: polynomials, Gaussian processes (GP), radial basis functions, neural networks and support vector machines. The method which is currently used more often in optimization of these type of problems is based on Gaussian processes.

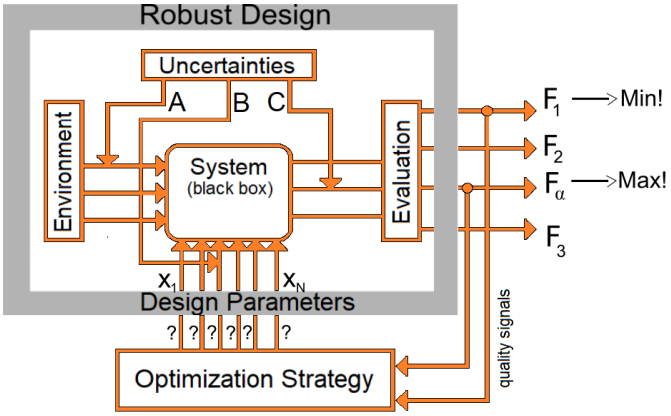


Figure 2.1: Robust system design of a black-box optimization framework. The system will account for three types of uncertainties: (1) The operating conditions, (2) Tolerance of design parameters, (3) Observed system performance.

Forrester & Keane (2009) presented a comprehensive overview into surrogate modelling, also known as meta-modelling. This involves creating simplified (low-cost) models to approximate more complex ones. This approach helps identify the optimal parameters for evaluating the true model, allowing for more efficient use of evaluations. Based on how the surrogates are used in the algorithm design these can be integrated or non-integrated.

Surrogate based algorithms

Jones et al. (1998) the first to introduce an Efficient Global Optimization (EGO) algorithm which combines Latin Hypercube Sampling (LHS) method and a Gaussian process model (i.e. DACE model - Design and Analysis of Computer Experiments) to be able to solve these types of black-box optimization expensive problems. To predict where it is better to sample next, the algorithm calculates the expected improvement (EI) function which provides the next best candidate design. On this new solution the algorithm will compute the true function value in order to update the GP model, which will provide a better approximation of the real world complex model.

In figure 2.2 the Gaussian process model fitted to 5 sample points (black) is shown, the model prediction is illustrated as a blue line and its error with a red line. Since the sampled points are evaluated using the expensive function evaluations the error at these points is zero. The rest of the function values follow a normal distribution of a random variate with the mean and the standard error given by the GP as follows: $F(x) \sim N(\hat{f}(x), \hat{\sigma}^2(x))$.

The EGO method works by first fitting a Gaussian process to the sampled points and then locates the point where the objective has the highest probability of improving over the current best design (i.e. f_{min}). The mathematical formulation of the EI can be expressed as in equation 2.17

$$E[I(\mathbf{x})] = (f_{min} - \hat{f})\Phi\left(\frac{f_{min} - \hat{f}}{\hat{\sigma}}\right) + \hat{\sigma}\phi\left(\frac{f_{min} - \hat{f}}{\hat{\sigma}}\right) \tag{2.19}$$

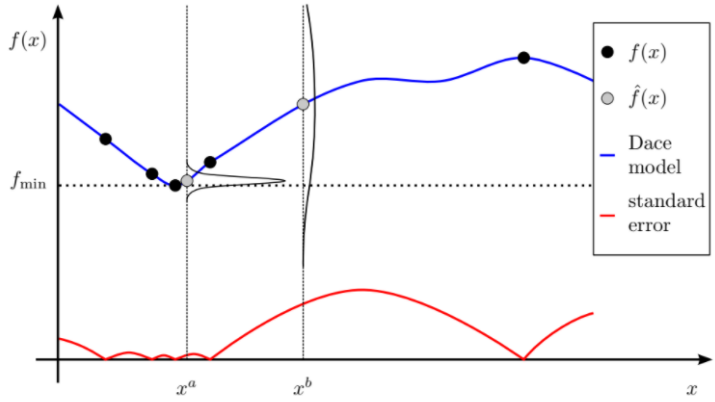


Figure 2.2: Interpolation of the evaluated points, presented in the figure with black and the Gaussian process model (blue line).The red line is the estimated error. The grey point are the new estimated sampled points where the EI is maximum. (Source:WP1 M4 (2015))

where: $\Phi()$ is the standard normal distribution, $\phi()$ is the density function, f_{min} is the best objective value found so far, $\hat{f} = \hat{f}(x)$ which is expected value of the new estimated point and $\hat{\sigma} = \hat{\sigma}(x)$ which is the standard deviation of the new estimated point. (Jones et al. 1998)

When maximizing the EI function the algorithm will search unexplored regions due to a larger prediction error. Therefore, this is the location where the maximum probability of improving the objective function value will be present. This procedure terminates when there is no other solution which can improve the EI or when the evaluation budget is reached.

Lim et al. (2007) presented a comparative analysis of more than a few mono-objective surrogate models which are used in the local stage of a memetic algorithm: Gaussian Processes, Extreme Learning Machines (ELMs), polynomial regression and Radial Basis Functions (RBFs), where the best approaches were Gaussian processes and polynomial regression. (Pavelski et al. 2016)

Guo et al. (2014) proposed an incremental extreme learning machine for online sequential learning problems and Hao & Liu (2014) used this method in a combination with a Differential Evolution algorithm to approximate the objective function for a discrete problem. In order to build the surrogate model the author chooses only the best individuals from a population. (Pavelski et al. 2016)

From all these methods Jones et al. (1998) algorithm presents a promising way to deal with expensive black-box optimization. The main disadvantages of this method is that it requires a large number of function evaluations, it is not able to deal with constraints or with multi-objective optimization problems. Therefore, many researchers tried to modify EGO to improve it and cope with different types of problems. This work will focus on the ones allowing to solve multi-objective optimization problems.

Knowles (2006) extended EGO to solve multi-objective optimization problem and intro-

duced the algorithm named ParEGO. The algorithms combines the EGO method and a evolutionary algorithm to be able to provide the Pareto optimal front for the multi-objective optimization.

In order to solve the multi-objective optimization problem Knowles propose to decompose the objective functions into multiple single-objective function using a set of reference directions as can be seen in the figure 2.3.

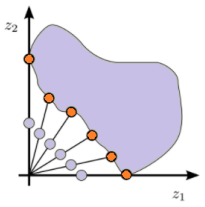


Figure 2.3: Objective Decomposition in Reference Directions.

For each reference direction now the algorithm will perform the EGO procedure of fitting a Gaussian process model. To build the surrogate model ParEGO sums all reference directions using the augmented Tchebycheff function presented in equation 2.18 below

$$f_\lambda = \max_{j=1}^{k}(\lambda_j f_j(x)) + \rho \sum_{j=1}^{k} \lambda_j f_j(x) \tag{2.20}$$

where: λ is a weight factor, ρ is a small positive number (usually 0.05) and f is the objective function.

The ParEGO method has been compared against alternative algorithms such as NSGA-II, Bin_MOPS (binary search algorithm), TOMO (Tau oriented multi-objective optimizer) and it outperform most of them on the set of test problems presented in the original paper as well as in others. One of the reasons behind it might be due to the specific design of ParEGO which can cope with a limited budget (function evaluations).

Another algorithm presented by Zapotecas Martínez & Coello Coello (2013) argues that Gaussian Process models might be in some cases computationally expensive and introduced a surrogate-assisted MOEA/D based on RBFs neural networks to create the surrogate model. A multi-objective evolutionary algorithms which uses meta-learning to select the best surrogate model is used in Pilát & Neruda (2012). Pavelski et al. (2016) proposed an algorithm called ELMOEA/D which uses ELM as surrogate model, the mechanism is similar with Pilát & Neruda (2012) but uses only one machine learning technique to construct the surrogate model.

Akhtar & Shoemaker (2016) introduced an algorithm called GOMORS which uses RBFs which iteratively builds the surrogate model as an approximation of the expensive objective function. Wu & Kozlowski (2017) uses a Bayesian framework where a polynomial chaos expansion computes the surrogate model of a nuclear reactor. Lu & Chen (2008) proposed self adaptive velocity particle swarm optimization (SAVPSO) to solve COPs.

ParEGO can outperform alternative algorithms due to its efficiency of approximating the expensive black-box functions and because it was designed in such a way that it will only use a limited budget of function evaluations.

Research on various aspects of the ParEGO algorithm has been conducted, Horn et al. (2015) included the use of LCB instead of EI, Hakanen & Knowles (2017) used the integration

of decision maker preferences inside the algorithm, Trautmann et al. (2017) introduced the selection of scalarizing norms,Davins-Valldaura et al. (2017) showed how the application of dual Kriging improved the algorithm performance.

Dealing with constraints in surrogate based framework

A recent study conducted by Tutum et al. (2015) has suggested a modification of the EI to handle constraints where the best feasible objective function is selected instead of the mean objective value. One main drawback of this method is that it has only been applied to single-objective optimization problems.

Hussein & Deb (2016) introduced a method to generate a surrogate Gaussian process model for the optimization problem in a decomposition-based evolutionary algorithm with a penalty-based method to handle constraints. The algorithm presents fast features, computationally speaking, due to the interconnection of multiple surrogate models. This algorithm handles constraints directly through modelling of the selection function. The idea behind is that when the solution **x** is feasible the unconstrained problem is solved. Otherwise, on the objective function value it will add to some extent a fitness based on the overall constraint violation. For the test problems which the authors presented in the paper when compared to the MOEA/D-EGO method the algorithm manages to approximate the Pareto optimal front. But when the constraint problems are presented the algorithm does not manage to find solutions which approximate the Pareto front. This algorithm has been tested on a small set of a unconstrained test problems, and few bi-objective problems with associated constraint functions ranging from 2 to 6 in numbers.(Duro et al. 2023)

Chugh et al. (2016) proposed an extension to surrogate based evolutionary multi-objective optimization algorithm, this is named as K-RVEA, for dealing with constraints. The author used penalty based function approach and decomposed the original problem into a number of sub-problems using reference direction vectors. The multiple sub-problems are then solved simultaneously to generate the Pareto set of solutions. The objective functions are surrogated but the constraints are considered inexpensive. This approach has been tested on benchmark problems with the objective functions from 3 to 10 and the associated constraints functions from 1 to 10. In real world problems this algorithm will require a large budget as predominantly the constraint function will also be expensive to evaluate and will require a surrogate model to be built instead.(Duro et al. 2023)

The problem with these methods is that the handling of the constraints creates big cliffs in the function that they are trying to estimate. Therefore, the algorithm is not able to estimate the function. The main reason for this inability is that the GP hyper parameters do not allow to estimate anything which is vertical in the landscape, due to the discontinuities. The authors do not elaborate on why the algorithm does not work, only mentioning that the problems they are trying to solve are hard problems. In conclusion, there are no specific algorithms which can

deal with constraint expensive black-box optimization problems.

Moving forward, the approaches that have used the probability of feasibility (PF) are discussed. PF is often multiplied with an infill criterion such as the probability of improvement (PI) or the expected improvement (EI) for handling constraints. These will be analysed later on.

Emmerich (2005), Emmerich & Klinkenberg (2008) and Emmerich et al. (2006) discuss methods to integrate infill strategies into a multi-objective evolutionary algorithm, amongst them were PI, lower confidence bound and EI. To determine the EI they used the hypervolume indicator, and named the method hypervolume-based EI. In order to deal with the constraint functions both PI and EI had been multiplied by the PF. To estimate the hypervolume the authors used MC sampling approach. This has been tested on many unconstrained single-objective and multi-objective problems, but also on a multipoint airfoil optimization problem having three objectives and 10 constraints.(Duro et al. 2023)

Couckuyt et al. (2014) highlights that if hypervolume indicator is used to determine the PI (hypervolume-based PI) the algorithm scales better with the objective functions and the number of Pareto solutions. When compared with Emmerich et al. (2006) produces better results. The benchmark has only been conducted for unconstrained multi-objective problems. (Duro et al. 2023)

Audet et al. (2000) presented a surrogate model based method to deal with constants where a bi-objective problem was introduced to deal with EI as presented in equation 2.19

$$\begin{aligned} \max \quad & E[I(\mathbf{x})] \\ \text{s.t.} \quad & EV_i(\mathbf{x}) \geq t_{EV}, \quad i = 1, ..., m \end{aligned} \tag{2.21}$$

where: EV_i is expected violation for each constraint function given by the equation 2.20

$$E[V_i(\mathbf{x})] = -\hat{g}_i \Phi\left(-\frac{\hat{g}_i}{\hat{\sigma}_{g_i}}\right) + \hat{\sigma}_{g_i} \phi\left(-\frac{\hat{g}_i}{\hat{\sigma}_{g_i}}\right) \tag{2.22}$$

where $\Phi()$ is the standard normal distribution of the constraint function, $\phi()$ is the density function of the constraint, $\hat{g}_i$ is the expected constraint function value, σ_{g_i} is expected value of the standard deviation of the constraint.

2.6 Research gap

By conducting a thorough analysis of the literature a number of gaps can be identified. By analysing the multi-objective algorithms available it can be seen that they have been implemented in different software such as: C, C++, Java and Pyhton. It will be interesting to see a comparison of how these packages perform, how good is the library in terms of running and developing new optimization algorithms and how easy it is to use the graphical user interface (GUI).

One area within the literature where a novel contribution can be made is the comparison of surrogate based optimizers such as ParEGO with one that can already deal with stochastic problems such as sParEGO. In order for the analysis to take place ParEGO can be validated using MC sampling or any other robustness metric to see if it can outperform other robust algorithms.

Chapter 3

Multi-objective optimizer design in Tigon and Liger

3.1 Introduction

Several optimization software tools are available for addressing real-world multi-objective optimization problems. Majority of these are commercial packages which are complex to use and require the user to buy a licence. Real-world problems often involve multiple conflicting objectives, necessitating the use of multi-objective optimization algorithms to find trade-off solutions. Effectively utilizing these tools and configuring algorithms to solve such problems require a deep understanding of how the algorithms work. Only after this one can find the algorithm which is best suited to the specific problem at hand, highlighting the need for expertise in optimization. (Duro, Yan, Giagkiozis, Giagkiozis, Salomon, Oara, Sriwastava, Morison, Freeman, Lygoe et al. 2021)

In Industry this is a clear challenge for engineers and non-specialists in optimization. To address this Liger was developed to be used as an alternative tool, as an open source optimization environment with an easy learning curve for non-experts in industry. Liger is a visual programming language, such as Simulink in Matlab, through which a user can interact to create an optimization work-flow by drag and drop method. Giagkiozis et al. (2013) introduced Liger environment for the first time at GECCO conference in 2013.

Duro, Oara, Sriwastava, Yan, Salomon & Purshouse (2021) reinstated Liger software in 2021 GECCO conference showing how this platform is specifically designed to be user-friendly for industry professionals without specialized expertise in optimization. This optimization software introduces a novel optimization library called Tigon based on a concept of design patterns, where the construction of optimization algorithms is done using simple reusable operator nodes. The library offers a diverse selection of multi-objective evolutionary algorithms which cover different paradigms in evolutionary computation; and supports a wide variety of real-world and

benchmark optimization problems.

Another benefit of using Liger is the ability of using more than one programming language within a single optimization model, for example the user can use Matlab or Python to import an optimization problem or to save the Pareto set. Furthermore, Liger's functionality can be seamlessly extended through plug-ins, providing access to state-of-the-art visualization tools and managing the graphical user interface (GUI). The GUI in Liger is a visual programming language through which a user can create an optimization work-flow. Finally, the user-driven interactive capabilities allow the decision-maker to interact with the optimization process at any point during execution. (Duro, Yan, Giagkiozis, Giagkiozis, Salomon, Oara, Sriwastava, Morison, Freeman, Lygoe et al. 2021)

3.2 The concept and architecture

In essence Liger is a software that can run on different operating systems and provides an easy to use interface for non-specialists in optimization. The software makes use of open source libraries and has interactive decision-making capabilities. The rapid prototyping capability of an optimization algorithm can be achieved through the operators available, library of algorithms and data management utility. Figure 3.1 shows the Liger architecture which consists primarily of two key components: the Tigon optimization library and the GUI.

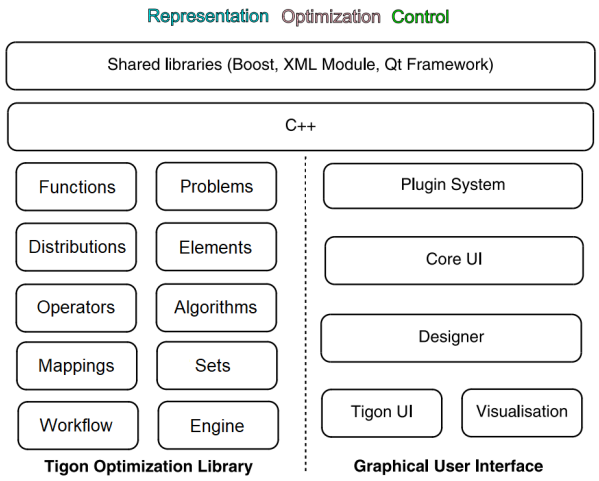


Figure 3.1: Liger high level architecture.

Each small block in the diagram represents a specialized component responsible for a specific functionality within the software. The diagram also illustrates the hierarchical dependencies among these components, where each component relies on the one below and to its left within each row. For instance, *Distributions* is a foundational component of Tigon, upon which all other components depend, while *Engine* is the highest-level component, relying on all others. Notably, there are no dependencies between Tigon and the GUI, meaning the Tigon library

operates independently and can be replaced with a different optimization library if needed. This separation also allows for the development of alternative front-ends for Tigon. (Duro, Yan, Giagkiozis, Giagkiozis, Salomon, Oara, Sriwastava, Morison, Freeman, Lygoe et al. 2021)

The general architecture and underlying framework used to extend Liger's GUI is built on a subset of QtCreator, an integrated development environment developed by Nokia to support the Qt library. This choice was primarily driven because it provides numerous pre-existing low-level utilities that have already been thoroughly tested and refined. Consequently, Liger also shares several architectural features with QtCreator.

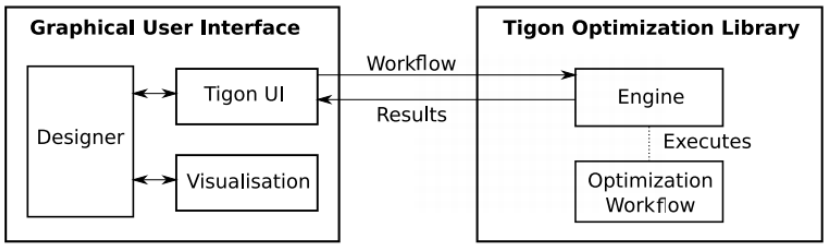


Figure 3.2: Interaction between GUI and Tigon library. (Duro et al. 2021)

Figure 3.2 illustrates the interface between the GUI and the Tigon library. The user constructs a workflow through the GUI and controls when to initiate the optimization process. When the process begins, the GUI will send the workflow to the optimization library via the Tigon UI plug-in. Upon receiving the workflow, the optimization library triggers its engine to process the workflow operators. After each iteration of the optimization algorithm, it communicates the results back to the Tigon UI, allowing the visualization plugin to update plots in real-time as the optimization progresses. (Duro et al. 2021)

3.3 Tigon library

Tigon is designed with a component-based architecture and implemented in a C++ object oriented paradigm to enhance its flexibility and re-usability. It offers users a collection of base classes that can be leveraged to interface with complex problem structures, create custom operators, develop new algorithms, and construct intricate optimization workflows. As shown in Figure 3.3 Tigon depends on shared libraries such as Boost, the XML Module, and the standard C++ library. Although, its core functionality does not rely on the Qt Framework, the current Tigon test suite does, development efforts are under way to reduce external library dependencies. Duro et al (2020) shows that a key aspect of Tigon is the use of decorator design patterns to set up an a workflow as suggested by Gamma (1995).

This structural pattern allows new functionality to be dynamically added to an existing object at runtime, treating operators as decorators. This approach introduces a novel paradigm for EA, enabling flexible composition and modification of workflows. Additionally, the decorator pattern adheres to the single responsibility principle as described in Winter (2014), who stipu-

lates that a class should have only one reason to change, enhancing maintainability. In Tigon, each task within the optimization process is handled by a dedicated operator, ensuring clear separation of responsibilities. More details of Tigon library is provided by Duro, Yan, Giagkiozis, Giagkiozis, Salomon, Oara, Sriwastava, Morison, Freeman, Lygoe et al. (2021).

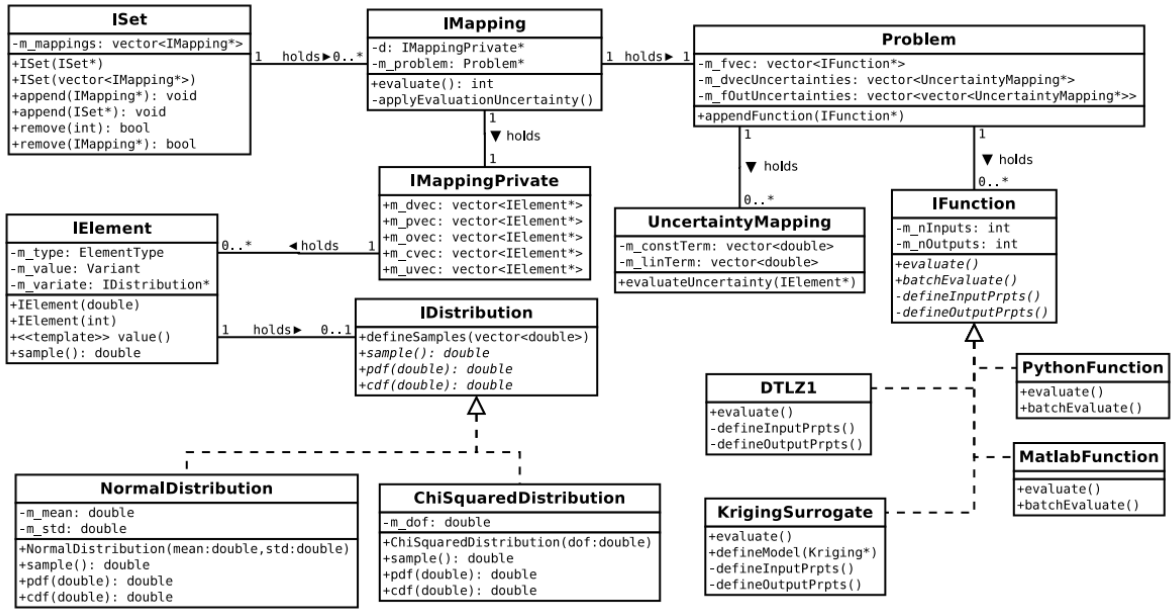


Figure 3.3: The relationship between clases in Tigon. (Duro et al. 2020)

In the above diagram each class is divided into three sections: the class name at the top, attributes in the middle section (e.g. prefixed with *m_*), and the class operations (methods) are at the bottom. The connection between classes represent two types of relationships: association and realization. Association, indicated by a plain line without an arrow shows that one class contains an object of another. For example, the class ISet can hold zero or more instances of the class IMapping, which will be stored in the vector m_mappings. Realization, represented by a line with an arrow indicates that one class (the subclass) implements the interface of another (the superclass). For example, IDistribution is a superclass that provides a generic interface for probability distributions, while NormalDistribution is a subclass implementing this interface, making it a more specialized version of IDistribution. (Duro et al. 2020)

Figure 3.4 shows how a simple optimization workflow can be created for a benchmark problem (e.g. DTLZ1 test problem) with a multi-objective optimization algorithm, in this case SMS-EMOA. The optimization is run for a number of iterations (function evaluations, e.g. 5000 function evaluations) and the results can be seen in the visualization nodes, these iterations only took couple of seconds to run. To create a workflow first an empty project needs to be created by clicking on *File/New/Liger Optimization Workflow*. Now on the left side we can open the tabs, select the nodes that will create the workflow and drag and drop them in the middle page as seen in Figure 3.4. All nodes depicted in the figure are connected sequentially, with MSN (Green) serving as the starting node and MEN (Red) as the final node. The intermediate nodes include:

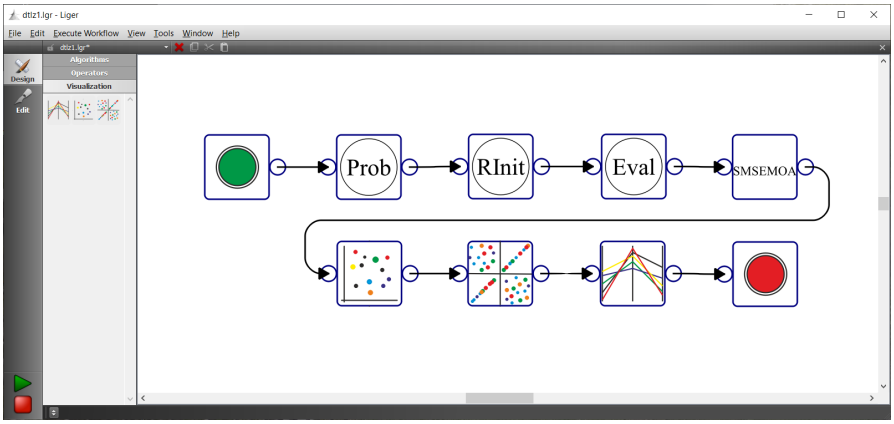


Figure 3.4: A optimization workflow in Liger GUI.

Prob, which loads the optimization problem; RInit, responsible for initializing a random population of solutions; Eval, which evaluates the optimization model; NSGA-II, executing the optimization algorithm. (Duro, Yan, Giagkiozis, Giagkiozis, Salomon, Oara, Sriwastava, Morison, Freeman, Lygoe et al. 2021)

The final five nodes correspond to visualization tools: a scatter plot, a matrix scatter plot, a parallel coordinates plot, an objective reduction plot and a glyphs plot. In Figure 3.5 the results of these nodes can be seen.

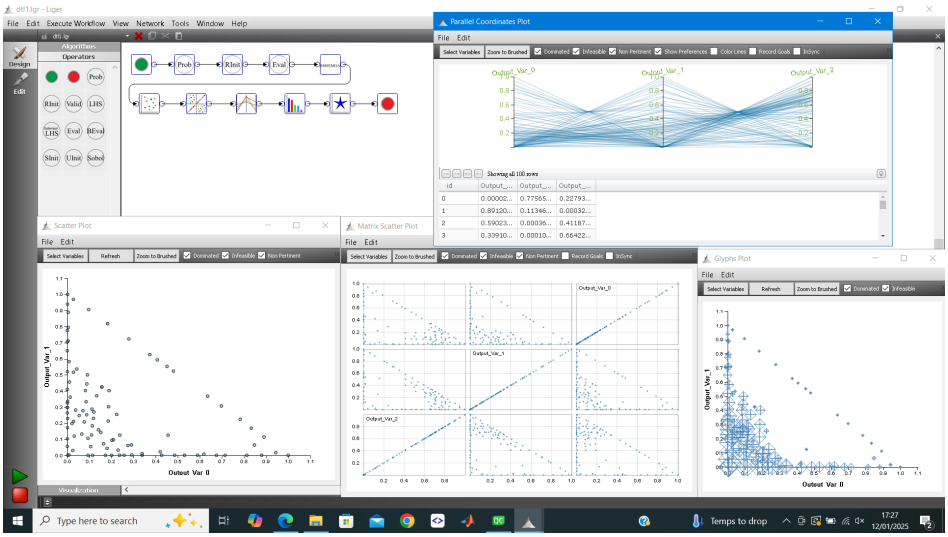


Figure 3.5: Visualization nodes in Liger GUI.

3.4 Related work

Zhou et al. (2011) research highlights numerous MOEAs that can be found in the literature. Several barriers hinder researchers and practitioners from using available optimizers. For example, sometimes the developers do not provide the source code or the test framework, and even when they do, multiple steps are required before the optimization process can begin. To

address these challenges various software libraries have been developed over the past years.

This discussion focuses on libraries with open-source licenses. One notable example is PISA (2013) which is a modular framework that includes a library of MOEAs, optimization problems and performance assessment tools. PISA uses a C interface and divides the optimization process into two distinct modules: one specific to the optimization problem and the other for handling processes independent of the problem meaning that it primarily relates to MOEA selection. However, the communication between these modules depends on text files, leading to significant time spent on disk input-output operations. This reliance on hard drive access reduces PISA's suitability for computationally intensive tasks compared to libraries that utilize faster memory resources. (Duro et al. 2021)

Another library is ParadisEO-MOEO developed by Liefooghe et al. (2011). This object-oriented framework, implemented in C++, is specifically designed for developing MOEAs and serves as an extended version of the Evolving Objects (EO) library developed by Keijzer et al. (2001). The EO library offers a collection of generic components that enable the construction of flexible evolutionary strategies, promoting extensive code reuse while allowing users to add, extend, and customize existing functionalities. However, despite its flexibility, EO is primarily intended for advanced users, as noted by its creators. Additionally, neither PISA nor ParadisEO-MOEO includes a graphical user interface, which may limit accessibility for less experienced practitioners.

Klempous et al. (2013) have developed Heuristic Lab (HL) a framework designed for heuristic and evolutionary algorithms. HL features a modular architecture based on a plugin system, enabling easy extension of its capabilities. It follows good software engineering principles and offers a comprehensive graphical user interface. However, HL only includes a classical MOEA (specifically NSGA-II) which has limitations when addressing multi-objective problems involving four or more objectives as presented in Purshouse & Fleming (2007). This limitation arises because NSGA-II's selection mechanism based on Pareto dominance becomes less effective as the number of objectives increases. Additionally, HL is implemented in C#, a language subject to Microsoft's software patents. Although, open source C# implementations allow cross-platform compatibility, there remains a risk that Microsoft could enforce its patents, potentially restricting the implementation availability. (Duro, Yan, Giagkiozis, Giagkiozis, Salomon, Oara, Sriwastava, Morison, Freeman, Lygoe et al. 2021)

In Java a number of optimization libraries have been implemented, such as: ECJ by Luke (2017), Opt4J by Lukasiewycz et al. (2011), EvA2 by Kronfeld et al. (2010), JCLEC-MOEA by Ramírez et al. (2015), MOEA Framework by Hadka et al. (2015), and jMetal by Durillo & Nebro (2011) and Nebro et al. (2015). One significant advantage of Java is its portability across different operating systems. However, its reliance on a Java Virtual Machine (JVM) makes it generally less suitable for computationally intensive tasks due to lower runtime performance compared to libraries written in C or C++ languages which run directly on the processor. Among

the libraries mentioned, ECJ, JCLEC-MOEA, and jMetal do not include a GUI, and ECJ's multi-objective optimization capabilities are limited, offering only two classical Pareto-based MOEAs: NSGA-II and SPEA2. More recently, PlatEMO was developed by Tian et al. (2017) for the MATLAB platform. However, this dependency on MATLAB requires users to purchase a license. (Duro et al. 2021)

3.5 Implementation of SMS-EMOA algorithm in Tigon library

Emmerich et al. (2005) introduced a state of the art algorithm called $\mathscr{S}$ metric selection Evolutionary Multiobjective Optimization Algorithm (SMS-EMOA). This is a Hypervolume-based EMOA where a hypervolume indicator ($\mathscr{S}$ metric) combined with a non-dominating sorting criteria is used as a selection method to guide the search towards the Pareto front balancing the diversity and convergence in the population. The key strength of SMS-EMOA is its use of the hypervolume indicator to select the solution that maximizes the dominated region of the objective space. This provides a natural measure of diversity in the population and helps ensure that the solutions are spread out across the Pareto front. The algorithm has been tested on standard benchmark problems found in literature and on real world applications showing promising results. As there were no indicator based optimization algorithms in Tigon library this algorithm has been selected and implemented in Liger software due to its notable performance.

The basics of SMS-EMOA can be observed in Algorithm 1 pseudo-code below. The procedure starts with a random initial population where a non dominating ranking criteria is applied (ideas borrowed from the well known NSGA-II algorithm). A new solution is generated by means of randomised variation operators. This new solution will try to enter the population if it will contribute more in term of hypervolume value compared to the solution from the worst ranked front. This steady state algorithm keeps the population size constant, hence both dominated and non-dominated solutions will be kept in the archive.(Beume et al. 2007)

Emmerich et al. (2005) explained that the runtime of SMS-EMOA depends on the hypervolume calculations, and for problems with more than two objective the Fleisher algorithms can be used to calculate the contributing hypervolume for each solution. The authors provide a modified reduce procedure that allows the algorithm to have a smaller run time complexity. The $\mathscr{S}metric$ was originally introduced by Zizler (1998) and explained that this is a measure of the *size of dominated space* as follows:

$$S(M) := \Lambda(\{\bigcup_i a_i \mid m_i \in M\}) = \Lambda(\{\bigcup_{m \in M} x \mid m \prec x \prec x_{ref}\}). \tag{3.1}$$

where the Lebesque measure Λ of the uniod hypercubes a_i is defined by a non-dominated point and a reference point: m_i and x_{ref} respectively. This metric inclines towards convex regions but has a major drawback due to the computational time needed to compute the $\mathscr{S}$ values. The reduce part of the algorithm discards one solution form the worst rank if in the population

Algorithm 1 SMS-EMOA algorithm

procedure MAIN ALGORITHM LOOP
    $P_0 \leftarrow$ init()
    $t \leftarrow 0$
repeat:
    $q_{t+1} \leftarrow generate(P_t)$
    $P_{t+1} \leftarrow \textbf{\textit{Reduce}}(P_t \cup \{q_{t+1}\})$
    $t \leftarrow t+1$
until: *termination condition fulfilled.*
Reduce *loop*:
    $\{\mathscr{R}_1, ..., \mathscr{R}_v\} \leftarrow$ fast-nondominated-sort(Q)
    $r \leftarrow argmin_{s \in \mathscr{R}_v}[\Delta_{\mathscr{S}}(s, \mathscr{R}_v)]$.
return $Q \backslash \{r\}$

$|\mathscr{R}_v \geq 1|$ there is an individual $s \in \mathscr{R}_v$ that minimises the following:

$$\Delta_{\mathscr{S}}(s, \mathscr{R}_v) := \mathscr{S}(\mathscr{R}_v) - \mathscr{S}(\mathscr{R}_v \diagdown \{s\}). \tag{3.2}$$

here the value of $\Delta_{\mathscr{S}}(s, \mathscr{R}_v)$ can be understand as the unique contribution of s to $\mathscr{S}metric$.

The reason of using Tigon library to implement SMS-EMOA is to show how easily an optimization algorithm can be configured using block operators and tags but also to contribute and add different types of optimization algorithms in the library. Therefore, using this object oriented feature of the software other members of the research team can use specific block operators from a range of algorithms, present in the library, and create new optimization methods.

Figure 3.6 illustrates the workflow of the algorithm implemented through operators and tags in Tigon. A random populations is assigned to the main optimization set in which each individual in the population will have a tag with the corresponding fitness. Using the non-dominating criteria borrowed from the NSGA-II algorithm the population is ranked.

Moving forward the operator *SMS-EMOA Reduce* arranges the population into sets based on their ranks in order to apply the hypervolume indicator. Furthermore, the solution from the last ranked set which has the lowest hypervolume value is removed from the population.

In the fist iteration, the *SMS-EMOA Reduce* operator is not evaluated in order to create a child solution and add it into the population. The child solution is created based on the *SBX crossover* and the *Polynomial Mutation* operators after the population passes through a *Tournament filtration*. The *Merge for Next Iteration* operator appends the new created solution to the main optimization set and sends it to be evaluated by the objective function. After this point the the iteration of the algorithms is incremented until the stopping criteria is met.

To test the Tigon implementation of the algorithm the *DTLZ 2* problem formulation which is presented in the Beume et al. (2007) paper was used. In figure 3.8 the result of the final

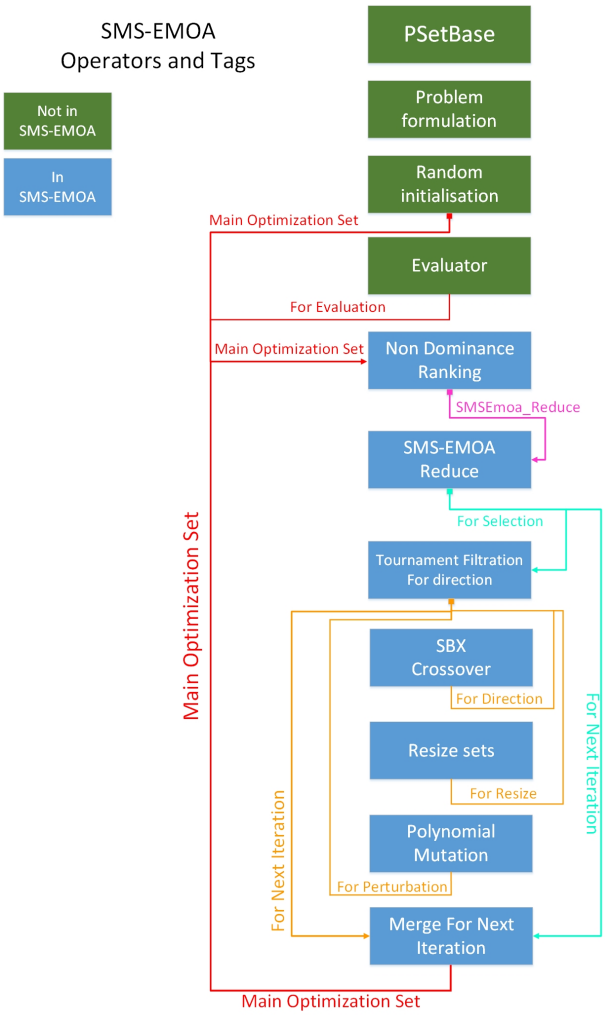


Figure 3.6: The SMS-EMOA algorithm workflow.

population is shown. It can be seen that the solutions found approximate exactly the Pareto-optimal front. Therefore, the result found are similar with the ones in the paper's authors, confirming that the implementation of the algorithm is similar with the pseudo code presented in the publication.

3.6 Results

The SMS-EMOA was tested on few benchmarking problems from the literature. The results shown here are aimed at comparability with the Emmerich et al. (2005) paper and the frequently cited papers of Deb et al. (2003*a*) and Deb et al. (2003*b*) where was ε-MOEA was developed. The variation operators are kept the same as in the original paper for comparability i.e. SBX crossover and polynomial mutation. The parameter settings for the ZDT test functions were: the initial population was set to 100 solutions, the decision vector was set to 10, the number of objective was set to 2 and the total budget was set to 20000 function evaluations. The algorithms was run five times and only "successful" runs were used to calculated the metric

values. For the DTLZ test problems the initial populations stayed the same at 100 solutions, the decision vector was set to 7 for DTLZ1 and 12 for DTLZ2 for 3 objective functions, the total budget was set to 30000 functions evaluations.

Table 3.1: Results of several EMOA on ZDT1, ZDT2 and ZDT4 test functions.

Problem	Algorithm	Convergence measure		S metric	
		Average	Standard deviation	Average	Standard deviation
ZDT1	NSGA-II	0.00054898	$6.62e-05$	0.8701	$3.85e-04$
	C-NSGA-II	0.00061173	$7.86e-05$	0.8713	$2.25e-04$
	SPEA2	0.00100589	$12.06e-05$	0.8708	$1.86e-04$
	e-EMOA	**0.00039545**	$1.22e-05$	0.8702	$8.25e-04$
	SMS-EMOA	0.00044394	$2.88e-05$	**0.8721**	$2.26e-04$
	SMS-EMOA dp	0.00044872	$4.35e-05$	**0.8721**	$1.59e-04$
	SMS-EMOA Tigon	0.00044394	$2.88e-05$	**0.8721**	$1.07e-05$
ZDT2	NSGA-II	**0.00037851**	$1.88e-05$	0.5372	$3.01e-04$
	C-NSGA-II	0.00040011	$1.91e-05$	0.5374	$4.42e-04$
	SPEA2	0.00082852	$11.38e-05$	0.5374	$2.61e-04$
	e-EMOA	0.00046448	$2.47e-05$	0.5383	$6.39e-05$
	SMS-EMOA	0.00041004	$2.34e-05$	**0.5388**	$3.60e-05$
	SMS-EMOA dp	0.00041923	$2.94e-05$	**0.5388**	$1.77e-05$
	SMS-EMOA Tigon	0.00041004	$2.34e-05$	**0.5388**	$1.23e-05$
ZDT4	NSGA-II	0.00639002	0.0043	0.8613	0.00640
	C-NSGA-II	0.00618386	0.0744	0.8558	0.00301
	SPEA2	0.00769278	0.0043	0.8609	0.00536
	e-EMOA	0.00259063	0.0006	0.8509	0.01537
	SMS-EMOA	**0.00251878**	0.0014	**0.8677**	0.00258
	SMS-EMOA dp	0.00289158	0.0019	0.8660	0.00324
	SMS-EMOA Tigon	0.00245872	0.0013	0.8595	0.0259

A run is considered successful if the solution set is well-distributed across the entire true Pareto front. The performance of the algorithm is evaluated using the $\mathscr{S}metric$ alongside the convergence metric, as outlined in Deb et al. (2003*b*). For each point in the solution set, the Euclidean distance to the nearest point on the Pareto front in the solution space is calculated. The convergence metric represents the arithmetic mean of these distances.

The results presented in Table 1 where the outcomes of both SMS-EMOA variants (the

original and the Tigon implementation) can be seen to be similar, and close to each other for the ZDT4 test problem. This indicates a favourable result, as the quality of SMS-EMOA is preserved while reducing algorithmic complexity and runtime. In the following discussion, the results for each test function are analysed individually without differentiating further between the SMS-EMOA variants.

The Pareto front of ZDT1 test problem can be observed in Figure 3.7, this is a smooth convex front and SMS-EMOA has ranked best $\mathscr{S}metric$ value, and provides the best convergence and spread of the Pareto front. The ZDT4 test problem can be observed in Figure 3.9 where it is noticed that the Pareto front is equivalent to ZDT1 problem, here SMS-EMOA provided the best results.

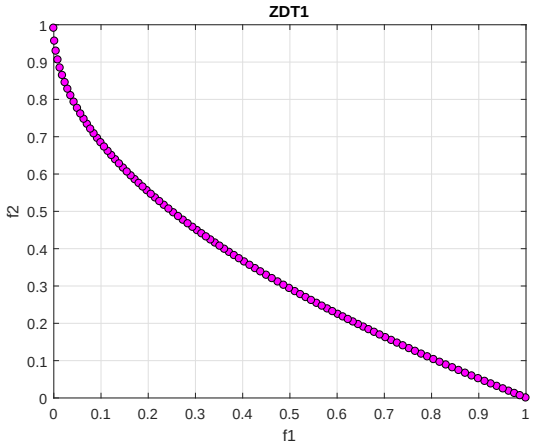


Figure 3.7: Pareto front approximated by the SMS-EMOA algortihm for the ZDT1 function .

Figure 3.8 shows the Pareto front of the ZDT2 test problem, this is a smooth concave front again SMS-EMOA provided the best result, covering most of the hypervolume. Based on the values provided by Deb et al. (2003*b*) it is assumed that all algorithms, performed as SMS-EMOA, transitioning through the second-best front with the majority of solutions on the best front.

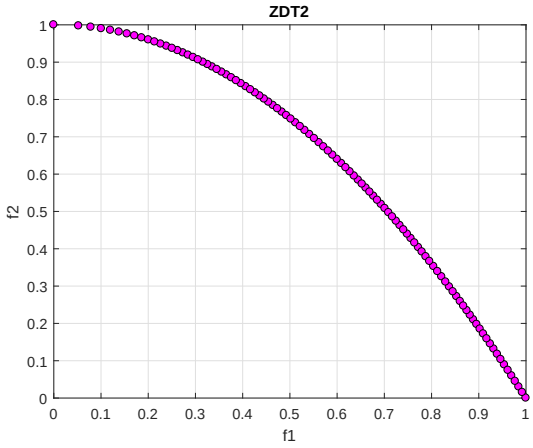


Figure 3.8: Pareto front approximated by the SMS-EMOA algortihm for the ZDT2 function .

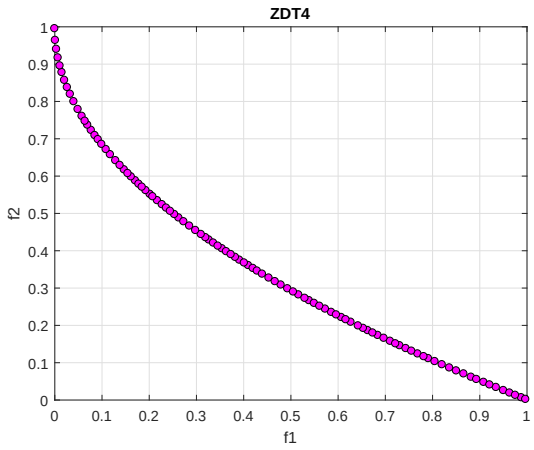


Figure 3.9: Pareto front approximated by the SMS-EMOA algortihm for the ZDT4 function .

The results of the DTLZ test functions are presented in Table 2. For easier comparison the values published in Beume et al. (2007) are copied here. The abbreviation ”NC” stands for ”not computed”, meaning that there are missing values in the literature. It is observed that the implementation of SMS-EMOA algorithm in Liger provided similar results to that of the authors. The Pareto front for DTZL1 function can be seen in Figure 3.10 and the Pareto front for the DTLZ2 test function can be seen in Figure 3.11.

The results demonstrate that SMS-EMOA significantly outperformed the previously mentioned EMOAs in terms of the $\mathcal{S}metric$ value. A similar trend is observed in the convergence metric, with the exception of the ε-MOEA applied to DTLZ1, which performs better than SMS-EMOA. As with the ZDT test functions, the comparison between the SMS-EMOA variants reveals no notable differences, confirming that the hypervolume-based selection method is effective across different frameworks, this can also be confirmed by looking at the statistical comparisons between the two from Table 3. (Beume et al. 2007)

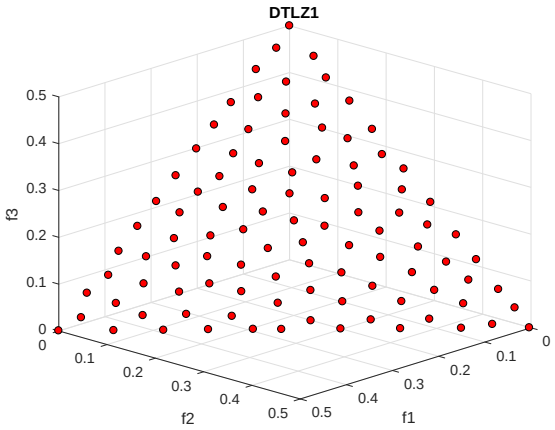


Figure 3.10: Pareto front approximated by the SMS-EMOA algortihm for the DTLZ1 function .

Using the two-sample t test it can be concluded that there is no significant difference between the Pareto sets in the referred paper and the ones obtained using Liger. These results

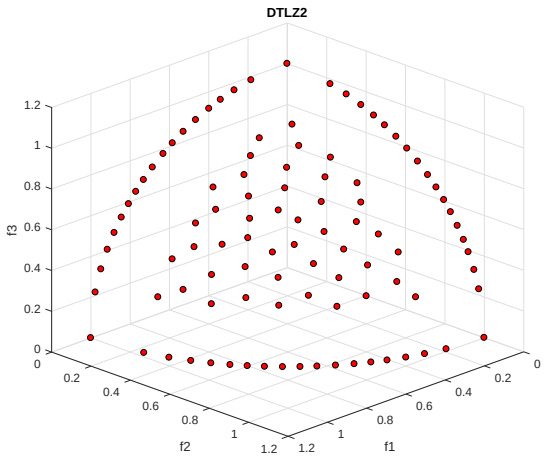


Figure 3.11: Pareto front approximated by the SMS-EMOA algortihm for the DTLZ2 function.

are shown in 4.2 and based on the values given by conventional criteria are not statistically significant.

Benchmarking demonstrates that SMS-EMOA generally outperforms NSGA-II, SPEA2, and ε-MOEA in solving two- and three-dimensional problems. In addition to its superior performance on test problems, this algorithm has also been successfully applied to a real-world case study providing good results. A challenge with SMS-EMOA is the computational cost associated with calculating the $\mathscr{S}metric$. While this burden is manageable for two and three dimensional problems it can become prohibitive as the number of dimensions increases. A limitation to Liger here is the necessity of further implementation of DTLZ3,DTLZ4 and ZDT3,ZDT6 test functions .

Table 3.2: Results of several EMOA on DTLZ1 and DTLZ2 test functions.

Problem	Algorithm	Convergence measure		S metric	
		Average	Standard deviation	Average	Standard deviation
DTLZ1	SPEA2	0.0033377	$3.54e-02$	0.315981	$6.98e-04$
	e-EMOA	**0.00245**	$9.52e-05$	0.298487	*NC*
	SMS-EMOA	0.0029175	$1.72e-04$	0.316930	$5.30e-05$
	SMS-EMOA dp	0.0028909	$1.06e-04$	0.316936	$8.38e-05$
	SMS-EMOA Tigon	0.0029175	$1.72e-04$	**0.316982**	$1.41e-05$
DTLZ2	C-NSGA-II	0.00986	$8.8e-04$	*NC*	*NC*
	SMS-EMOA	**0.0063652**	$3.20e-04$	0.757911	$4.49e-05$
	SMS-EMOA dp	0.0065383	$5.12e-04$	**0.757994**	$4.74e-05$
	SMS-EMOA Tigon	**0.0063652**	$3.20e-04$	0.757949	$6.61e-05$

Table 3.3: p-values obtained by the two-sample t test. The p value (significance level of 5%) indicates rejection of the null hypothesis that two samples being compared have equal medians. A h value of zero supports that there is not enough evidence to reject the null hypothesis.

Test Problems	***t* test values**		
	p	h	confidence interval
ZDT1	1	0	-2.33e-04 to 2.33e-04
ZDT2	1	0	-3.92e-05 to 3.92e-05
ZDT4	0.50111	0	-3.5e-02 to 1.86e-02
DTLZ1	0.0668	0	-4.55e-06 to 1.08e-04
DTLZ2	0.3187	0	-4.44e-05 to 1.20e-04

Chapter 4

Comparison of sParEGO with simpler versions of ParEGO algorithm for robust multi-objective optimization

This chapter covers the surrogate based optimization algorithms from Liger software. These algorithms have been successfully applied to solve real world optimization problems. The aim here is to compare sParEGO optimization algorithm with MC based alternatives on a range of surrogate based benchmarking problems.

4.1 sParEGO optimization algorithm

Duro et al. (2019) work extended the Knowles (2006) algorithm to deal with the uncertainty in real-world optimization problems. The authors proposed an algorithm named sParEGO which uses a novel uncertainty quantification approach to examine the robustness of a candidate solution without having to compute an expensive sampling technique.

sParEGO is a surrogate-based multi-objective optimization algorithm designed for addressing stochastic multi-objective problems (MOP). It shares many characteristics with ParEGO, in particular its ability to approximate expensive MOP with a limited budget. The key concept behind sParEGO is the uncertain distribution in the objective space of each candidate solution. This is not determined through traditional uncertainty quantification methods (such as MC sampling) instead, each solution is evaluated once and the distribution is approximated using the performance of the nearby solutions (neighbours). A pseudo-code for sParEGO is provided in Algorithm 2.

The general working principles are outlined as follows: firstly, it normalizes the decision and objective space to non-dimensional units using the upper and lower boundary of the i^{th} decision variable. After this the nadir and ideal vectors are calculated using:

Algorithm 2 sParEGO Pseudo-code

procedure MAIN ALGORITHM LOOP
  $\mathscr{D} \leftarrow$ set of all reference direction vectors
  $\mathscr{X} \leftarrow$ generate initial set of solutions using n_{init} and δ_{pert}
  $\mathscr{Z} \leftarrow f(\mathscr{X})$
while: *stopping criteria not met* ***do***
  Shuffle the set $\mathscr{D}$
  for all $\boldsymbol{d} \in \mathscr{D}$ ***do***
  update ideal and nadir vectors
  $\mathscr{S} \leftarrow$ calculate scalar fitness value of all solutions
  $\mathscr{I} \leftarrow$ **RobustnessApproximation**$(\mathscr{X}, \mathscr{S}, \delta)$
  model $\leftarrow$ fit a Surrogate model to the indicator values $\mathscr{I}$ using n_{max}
  $\mathbf{x}^{new} \leftarrow$ maximise the expected improvement based on *model*
  $\mathbf{x}^{pert} \leftarrow$ add a neighbour to x^{new} using δ_{pert}
  $\mathscr{X} \leftarrow \mathscr{X} \cup \{x^{new}, x^{pert}\}$
  $\mathscr{Z} \leftarrow \mathscr{Z} \cup \{f(x^{new}), f(x^{pert})\}$
  end for
end while
RobustnessApproximation*$(\mathscr{X}, \mathscr{S}, \delta)$*
  for all $\boldsymbol{x}_i \in \mathscr{X}$ ***do***
  update the neighbourhood $\mathscr{N}(x_i)$ for a given δ
  end for
  $\mathscr{I} \leftarrow \emptyset$
  for all $\boldsymbol{x}_i \in \mathscr{X}$ ***do***
  approximate the distribution of S_i
  calculate robustness indicator $I[S_i]$
  $\mathscr{I} \leftarrow \mathscr{I} \cup I[S_i]$
  end for
  return $\mathscr{I}$
end procedure

$$\begin{aligned} \widetilde{x_i} &= (x_i - x_i^l)/(x_i^u - x_i^l), i = 1, ..., n_x \\ \widetilde{z_j} &= (z_j - z_j^l)/(z_j^u - z_j^l), j = 1, ..., n_x \end{aligned} \tag{4.1}$$

These normalized values are applied to all operations within the algorithm. Prior to evaluating a candidate design, another normalization to its original dimension is required. sParEGO breaks down the overall MOP into several single-objective problems by utilizing a set of reference direction vectors, which steer the search towards various regions of the Pareto front. The complete set of direction vectors is represented by $\mathscr{D}$, and constructed using a Simplex Lattice design as given by:

$$\mathscr{D} = \left\{ \mathbf{d} = [d_1, ..., d_{n_z}] | \sum_{j=1}^{n_z} d_j = 1 \wedge d_j \in \right\} \frac{0}{h}, \frac{1}{h}, ..., \frac{h}{h} \Bigg\} \quad \text{for} \quad \text{all} \quad j \Bigg\} \tag{4.2}$$

where h is a parameter that defines the divisions corresponding to each objective. The direction vectors are selected sequentially from the set $\mathscr{D}$, one at a time. Then the vectors in set $\mathscr{D}$ are shuffled when the optimizer has explored all the direction vectors. This is done to ensure that no bias arises from repeatedly using the same sequence of direction vectors. (Duro et al. 2019)

A scalar fitness value for each solution is computed using the same scalarizing function as in ParEGO, this is the weighted Tchebycheff as give in:

$$s = \max_{1 \leq i \leq n_z} \{w_i z_i\} \tag{4.3}$$

where each objective z is mapped to a scalar value using the weight w. The robustness indicator for these solutions are estimated and stored in the set $\mathscr{I}$ in order to construct the surrogate model. A search is then performed on the model in order to find a solution, x^{new} that optimizes the given robustness indicator using the expected improvement. This step is similar to the one applied in ParEGO optimization algorithm. A new solution, x^{pert} will be generated by sampling x^{new} to ensure that each generated solution has at least one nearby neighbour. The new solutions will have their performance evaluated and recorded in $\mathscr{Z}$. The algorithm then returns to Line 5, and the process repeats until the stopping criterion is met.

The robustness metric for the final set is estimated using the procedure outlined in Line 18. The initial step to compute this involves identifying all nearby solutions for each candidate. To achieve this, the concept of a neighbourhood is defined. Two solutions are considered neighbours only when their normalized decision-space distance is within a user-specified neighbourhood distance. The statistical properties of a candidate's performance are then approximated using the performance data from its neighbouring solutions. Finally, by taking into consideration the statistical performance the robustness indicator values are calculated. In practice the ninety percentile is usually proposed. (Duro et al. 2019)

The key distinction between sParEGO and ParEGO lies in how they handle the evaluation function's outcome. sParEGO algorithm treats this as a realization of a random variable. Meaning that the scalarized function value cannot be directly used to build the surrogate model and

instead a utility indicator is employed. For each direction vector the surrogate model is used to identify a design that optimizes a specified robustness indicator. The main objective here is to avoid repeated sampling of every candidate design to determine its statistical properties in the objective space. The measures of central tendency and dispersion are estimated using the information from the other evaluated candidate designs (neighbouring solutions).

To approximate the central tendency and account for variation, the performance of a candidate design can be determined from the performance of neighbouring designs if x_i and x_j are considered neighbours. Two solutions are considered neighbours when the normalized Euclidian distance is smaller or equal to δ as shown in:

$$\|x_i - x_j\|_2 \leq \delta \tag{4.4}$$

Duro et al. (2019) shows that for a solution x_i with a scalar fitness value s_i, the statistical properties of this fitness are approximated by first defining the neighbourhood $\mathcal{N}(x_i)$ of the solutions as in:

$$\mathcal{N}(x_i) = \{x_j \in X | \quad \|x_i - x_j\|_2 \leq \delta\} \tag{4.5}$$

Next the approximated mean $\mu_{s,i}$ of the scalar fitness function for x_i is calculated as:

$$\mu_{s,i} = \frac{1}{\varsigma_i} \sum_{x_j \in \mathcal{N}(x_i)} v_j s_j \tag{4.6}$$

here the overall neighbourhood size ς_i is smaller than $|\mathcal{N}(x_i)|$ as shown in:

$$\varsigma_i = \sum_{x_j \in \mathcal{N}(x_i)} v_j \tag{4.7}$$

where v_j it's a larger weight given to the solutions that are closer to x_i and can be derived as:

$$v_j = \frac{\delta - \|x_i - x_j\|_2}{\delta} \tag{4.8}$$

Now that the expected mean is known the expected variance can be determined using the Equation 4.9:

$$\sigma^2_{s,i} = \frac{1}{\varsigma_i} \sum_{x_j \in \mathcal{N}(x_i)} v_j (s_j - \mu_{s,i})^2 .1 \tag{4.9}$$

After the estimation of the statistical properties of the scalar fitness function, the random variable $S(x)$ is assumed to follow a normal distribution with a computed mean and variance. The robustness indicator is determined based on this distribution for a specified confidence level c (where $c \in [0, 100]$). This indicator denoted as $I_c[S]$ corresponds to the c^{th} percentile of the normal distribution with mean μ_s and variance σ^2_s. In the current iteration, this value defines the solution's fitness. (Duro et al. 2019)

To compare the performance of sParEGO optimization algorithm a modified ParEGO algorithm is required. Using MC sampling method two algorithm variants are obtained. One modification done to ParEGO algorithm is that when the optimization budget has been exhausted

the final population will undergo a MC sampling, using 100 replication, in order to compute the ninety percentile robustness measure. Another modification can be made in Tigon in such a way that at the end of each iteration the new candidate solution will be re-evaluated $N = 10$ times using MC sampling and then take the empirical ninety percentile (i.e. the second worst in the ordered set) as a measure of robustness. This value of performance is fixed for the duration of the optimizer (until it will get validated post-run). This second modification is referred in this work as MC-ParEGO (a MC version of ParEGO). All three algorithms had their final performance set validated using 100 MC replications. MC-ParEGO's evaluation budget includes the N replications i.e. only 100 candidate solutions are evaluated altogether. Each solution is evaluated 10 times. Hence, this constitutes a budget of 1,000 function evaluations, which is comparable to that of ParEGO.

4.2 CODeM Test problems

To compare the performance of these algorithms a benchmark test was conducted on a set of stochastic problems using the Salomon, Purshouse, Giaghiozis & Fleming (2016) toolbox. This toolbox can generate scalable robust multi-objective optimization problems, which have been named in Liger CODeM1 to CODeM6 test problems. The test procedure for these 3 algorithms was designed to see which of them is more effective to identifying robust trade-off solutions. This in return can help practitioners to identify an algorithm which is best suited for real world situations.

A hypothesis testing approach is adopted to evaluate the performance of sParEGO algorithm in comparison with the two version of ParEGO algorithm. A specified computational budget for multi-objective optimization problems is also defined. The first hypothesis is that, in highly stochastic problems with smooth regions near the Pareto front the ParEGO algorithm may select seemingly superior solutions that lack robustness, while sParEGO's convergence is not compromised. The second hypothesis will be that MC-ParEGO algorithm will outperform sParEGO algorithm's convergence for lower budget problems. This is because the sParEGO algorithm features will require a larger computational budget to produce similar results.

To test the two hypotheses six variants of the CODeM toolkit are used. These test problems are based on WFG test suite developed by Huband et al. (2006). The problem features available in this suite are as follows: the WFG4 test problem is separable, multimodal and concave; the WFG6 test problem is non-separable, multimodal and concave; the WFG8 test problem is non-separable, multimodal, concave and with bias. CODeM6 test problem is based on DTLZ1 as this problem is separable, affine and unimodal. All the test problems have two objectives and five decision variables and for reproducibility the parameter settings are kept the same as in Salomon, Purshouse, Giaghiozis & Fleming (2016).

Inside this CODeM toolkit all operators work with normalized decision and objective

spaces, the random objective vector is also normalized. When a candidate solution enters the uncertainty generator it is normalized using:

$$\begin{aligned} \widetilde{x_i} &= \frac{x_i - x_{i,lb}}{x_{i,ub} - x_{i,lb}}, i = 1, ..., n_x \\ \widetilde{z_j} &= \frac{z_j - z_j^\star}{z_j^{\star\star} - z_j^\star}, j = 1, ..., n_z \end{aligned} \tag{4.10}$$

here $x_{i,lb}$ and $x_{i,ub}$ are the lower and upper bound for the i^{th} decision variable, and $\mathbf{z}^\star$ and $\mathbf{z}^{\star\star}$ are the ideal and anti-ideal vectors. To describe the vectors in the objective space this toolkit uses polar representation where $\mathbf{z}$ can be described by its magnitude and direction as given by:

$$\mathbf{z} = z\hat{\mathbf{z}} \tag{4.11}$$

where z is the vector's Euclidean distance and the direction of $\hat{\mathbf{z}}$ is defined on $n_z - 1$ simplex.(Salomon, Purshouse, Giaghiozis & Fleming 2016)

Salomon, Purshouse, Giaghiozis & Fleming (2016) describes how the uncertainty generator produces an objective vector with inherent uncertainty consisting of three key components. The first one is the univariate distribution function for the radial component; the second is the maximum perturbation distance for the random direction vector; and the third one is the curvature norm for the perturbation in the perpendicular direction. These components can remain fixed or adapted based on the characteristics of the candidate solution. For instance, in problems with simple geometric structures the perturbation norm may stay constant. On other problems with more complex objective spaces this perturbation may vary depending on the direction.

The authors also show how an uncertain objective vector radial component can be defined as a PDF $f(\rho)$ over the interval $[0,1]$, and consists of n basic parametric distribution functions as shown in:

$$f(\rho) = \sum_{i=1}^{n} w_i f_i(\rho), \quad where \quad \sum_{i=1}^{n} w_i = 1 \tag{4.12}$$

These three basic distributions available to choose from the toolkit are uniform distribution, triangular distribution and smooth peak distribution. These are all defined in a similar fashion according to their position and locality. The position refers to where the distribution function resides on the interval and the locality describes how concentrated the PDF is in this region. (Salomon, Purshouse, Giaghiozis & Fleming 2016)

The parameters defining the uncertain objective vector are derived from the deterministic characteristics of the candidate solution in both the design and objective space. These characteristics serve as a kernel of the uncertain objective vector, denoted by $\mathbf{\Psi}$ as given by:

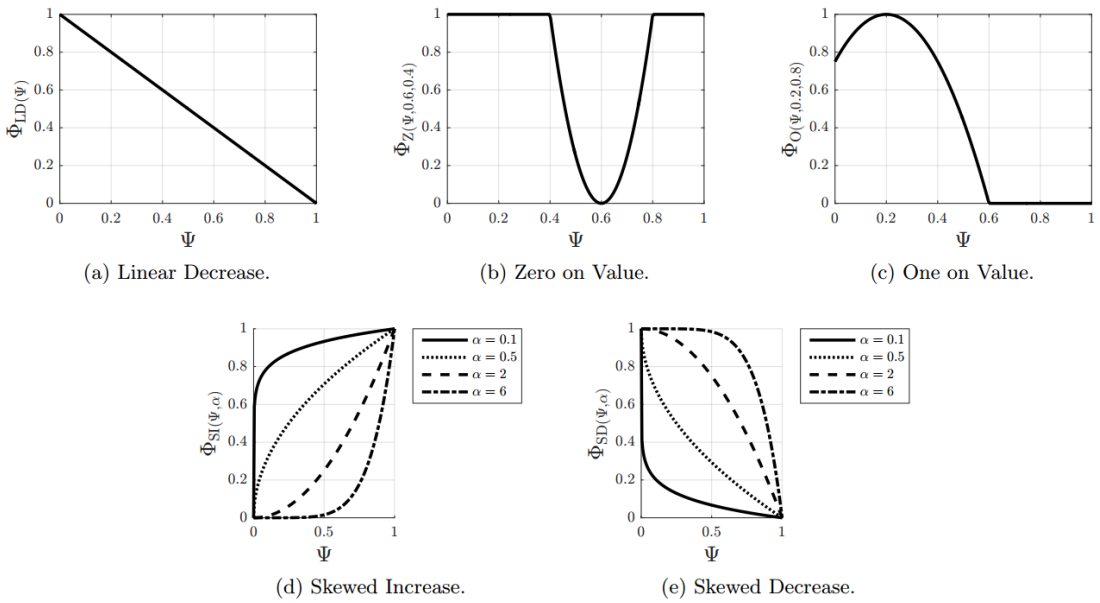
$$\begin{aligned} \mathbf{\Psi} = [\Psi_r, \Psi_s, \Psi_{x,1}, ..., \Psi_{x,n_x}, \Psi_{z,1}, ..., \Psi_{z,n_z}], \\ 0 \leq \Psi_i \leq 1, \quad i = 1, ..., n_x + n_z + 2. \end{aligned} \tag{4.13}$$

where Ψ_r is the remoteness parameter, Ψ_s is the symmetry parameter, $\mathbf{\Psi}_x$ is the decision vector parameter and $\mathbf{\Psi}_z$ is the objective vector parameter. This toolkit offers a set of functional operators to help and manipulate the kernel parameters which can be seen in Figure 4.1. All functions

accept a value of $\Psi \in [0,1]$ and transform it into $\Phi(\Psi) \in [0,1]$ which can be visualised in Figure 4.2. (Salomon, Purshouse, Giaghiozis & Fleming 2016)

Function	Description	Parameters		Definition
Linear Decrease	Function values decrease at a constant rate.			$\Phi_{\mathrm{LD}}(\Psi) = 1 - \Psi$
Skewed Increase	Function values increase at a changing rate. Sensitive to changes of high Ψ values when $0 < \alpha < 1$ and low values when $\alpha > 1$.	$\alpha > 0$	Skewness	$\Phi_{\mathrm{SI}}(\Psi, \alpha) = \Psi^{\alpha}$
Skewed Decrease	Function values decrease at a changing rate. Sensitivity as for Φ_{SI}.	$\alpha > 0$	Skewness	$\Phi_{\mathrm{SD}}(\Psi, \alpha) = 1 - \Psi^{\alpha}$
Zero on Value	Parapolic function of width β with a minimum $(\alpha, 0)$. Function values equal to 1 outside $\alpha \pm \beta/2$.	$\alpha \in [0,1]$ $\beta > 0$	Zero value Width	$\Phi_{\mathrm{Z}}(\Psi, \alpha, \beta) = \min\left(\frac{4(\Psi-\alpha)^2}{\beta^2}\ ,\ 1\right)$
One on Value	Parapolic function of width β with a maximum $(\alpha, 1)$. Function values equal to 0 outside $\alpha \pm \beta/2$.	$\alpha \in [0,1]$ $\beta > 0$	One value Width	$\Phi_{\mathrm{O}}(\Psi, \alpha, \beta) = \max\left(1 - \frac{4(\Psi-\alpha)^2}{\beta^2}\ ,\ 0\right)$

Figure 4.1: Transformations function in CODeM toolkit.(Salomon, Purshouse, Giaghiozis & Fleming 2016)



(a) Linear Decrease. (b) Zero on Value. (c) One on Value.

(d) Skewed Increase. (e) Skewed Decrease.

Figure 4.2: Transformations function.(Salomon, Purshouse, Giaghiozis & Fleming 2016)

CODeM1, CODeM2 and CODeM3 are problems generated by the toolkit by modifying WFG4 test problem. CODeM1 problem is a deterministic multimodal problem with a concave Pareto front such as in the original test problem. It has uncertainty in the radial direction away from the objective space. Solutions that are close to the deterministic front will have larger uncertainty. CODeM2 test problem has the same concave deterministic Pareto front but the uncertainty is now on the perpendicular direction away for the objective space. CODeM3 test problem will keep the same concave Pareto front but has uncertainty in both radial and perpendicular directions. Radial uncertainty is lower as you get closer to the deterministic front and

in regions with similar values of the objectives. Perpendicular uncertainty is constant, except in regions where the first objective is close to 0.9.

CODeM4 test problem is composed of a deterministic multimodal problem with a concave Pareto front, as given in WFG6 problem, and uncertainty in both radial and perpendicular directions. Radial uncertainty is constant for all solutions and Perpendicular uncertainty tends to be reduced for solutions characterized by objective vectors with similar component values.

CODeM5 test problem is a deterministic non-separable problem with concave Pareto front, as in WFG8 test problem, and has uncertainty in both radial and perpendicular directions. Here the uncertainty is proportional to the first decision variable.

CODeM6 problem setup involves a deterministic and multimodal problem whose Pareto front lies on a plane, as in DTLZ1 test problem. The 100 scaling factor is removed from the distance function and uncertainty in both radial and perpendicular directions are added. The feasible objective space is bonded between $\sum_{j=1}^{n_z} z_j = 0.5$ and $\sum_{j=1}^{n_z} z_j = 1.125(n_x - n_z + 1) + 0.5$.

The radial probability function in CODeM6 is a uniform distribution with the deterministic value equal to the lower bound. The locality of the distribution has been set to be inversely correlated to the optimality of the deterministic vector. Hence, solutions that are far away from the deterministic Pareto front $(i.e. \Psi_r \geq 0.5)$ will have no radial uncertainty (maximum locality) and as they get closer to the Pareto front the locality decreases and the degree of radial uncertainty increases as given by:

$$f(\rho) = f_u\Big(\rho, \Psi_r, 0.9 + 0.1\Phi_z(\Psi_r, 0, 1)\Big) \tag{4.14}$$

On the directional distribution the perturbation norm p is set to one, and the radius δ is small for high symmetry. This increases near the boundaries of the objective space as suggested in:

$$\delta = 0.02 + 0.1\Phi_{LD}(\Psi_s) \tag{4.15}$$

The purpose of this is to make the identification of the boundary solutions more challenging, in order to mimic real world scenarios. (Salomon, Purshouse, Giaghiozis & Fleming 2016)

Experimental setting

For all three algorithms the number of reference direction vectors are set to 10, the population size n_{init} is set to 20, the optimization budget is set to 1000 function evaluations, and the surrogate size is set to 30 solutions. All test problems have 2 objective and 5 decision variables, and for MC validation the sampling size was set to 100. In sParEGO algorithm the same parametric settings were kept as given by Duro et al. (2019), $\delta = 0.1\sqrt{n_x}$, $\delta_{pert} = \frac{\delta}{2}$ and the confidence level c for the robustness indicator $I_c[S]$ is set to 0.9 (ninety percentile). To test the statistical significance all the algorithms have had to run for 21 times. At the end of these 21 runs the hypevolume measures are computed and the results can be observed in Table 4.1. The reference point, to compute the hypervolume measure, for CODeM5 was set to $[3.5, 4.5]$ and

for the rest of the test problems was set to $[2.5, 4.5]$. These points were calculated looking at the nadir point for each run and a constant was added to make sure all solutions were captured.

Table 4.1: Hypervolume indicator results for CODeM test problems.

Test Problems	**ParEGO Hypervolume**				**MC-ParEGO Hypervolume**				**sParEGO Hypervolume**			
	Median	Mean(Std)	Max	Min	Median	Mean(Std)	Max	Min	Median	Mean(Std)	Max	Min
CODeM1	4.1848	4.1887(0.1511)	**4.5446**	3.8970	4.1772	4.1868(0.1478)	4.5245	3.9003	**4.4352**	**4.3174(0.1083)**	4.4928	4.1384
CODeM2	4.1313	4.1152(0.1584)	4.3641	3.8671	4.1018	4.1193(0.1613)	4.3320	3.8320	**4.2615**	**4.2736(0.1203)**	**4.4743**	4.0705
CODeM3	4.1542	4.1659(0.0938)	4.3508	4.0000	4.1830	4.1858(0.1008)	**4.4341**	4.0268	**4.2437**	**4.2419(0.1131)**	4.4181	3.9256
CODeM4	**3.8198**	3.8288(0.0715)	3.9418	3.6942	3.8108	**3.8329(0.01761)**	**3.9637**	3.6985	3.5870	3.5883(0.0958)	3.8125	3.3752
CODeM5	7.9805	7.9917(0.4427)	8.8352	6.9556	8.0041	7.9560(0.3514)	8.5948	7.2169	**8.3933**	**8.3726(0.4899)**	**9.11398**	7.4640
CODeM6	10.7358	10.7222(0.08141)	10.8203	10.5575	**10.7390**	10.7193(0.0885)	10.8212	10.4712	10.7366	**10.7225(0.0794)**	**10.8580**	10.5559

4.3 Results

Figure 4.1 shows the median hypervolume indicator for the 21 runs on CODeM1 test problem. It can be seen that sParEGO algorithm although it provides better performance compared with the two modified versions of ParEGO the convergence up to 450 function evaluations is worse compared to that of the two versions of ParEGO. The validated ParEGO and MC-ParEGO algorithms provided similar performance. In the first 400 iterations it can be seen that MC-ParEGO manages to converge faster to the Pareto front providing better hypervolume value. Figure 4.5 and Figure 4.6 shows the different empirical attainment functions (EAFs) of the non-dominated solution given by the three algorithms. Looking at Figure 4.5 it can be said that sParEGO provides a better spread of solutions in the left side of the Pareto front but has less convergence on the right side of the front.

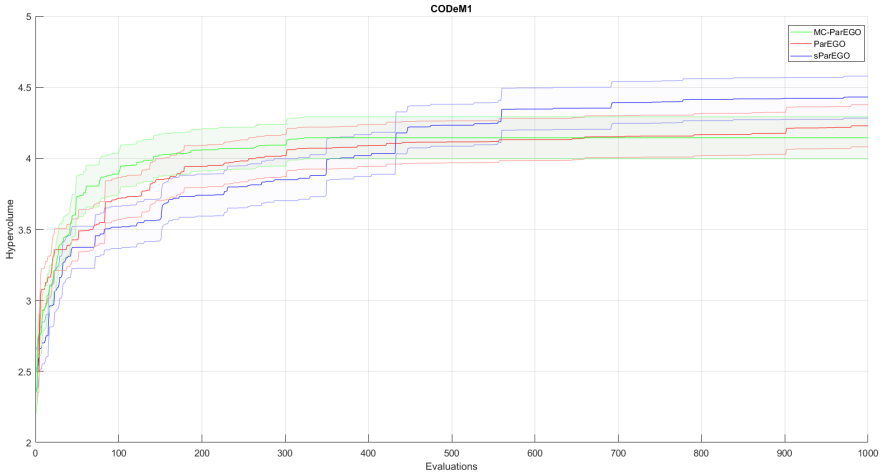


Figure 4.3: CODeM1 median run Hypervolume indicator.

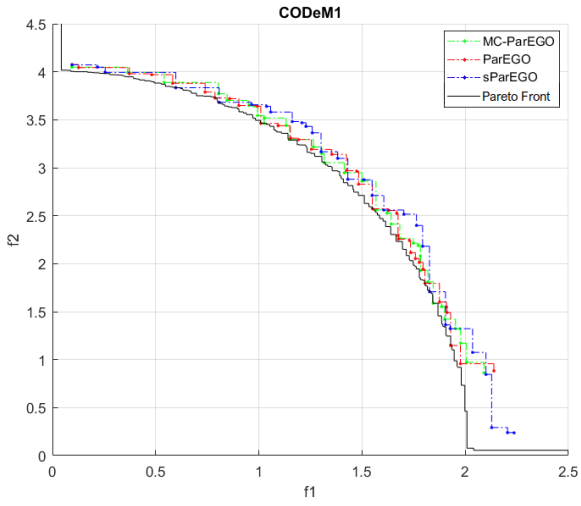


Figure 4.4: CODeM1 median run non dominated population.

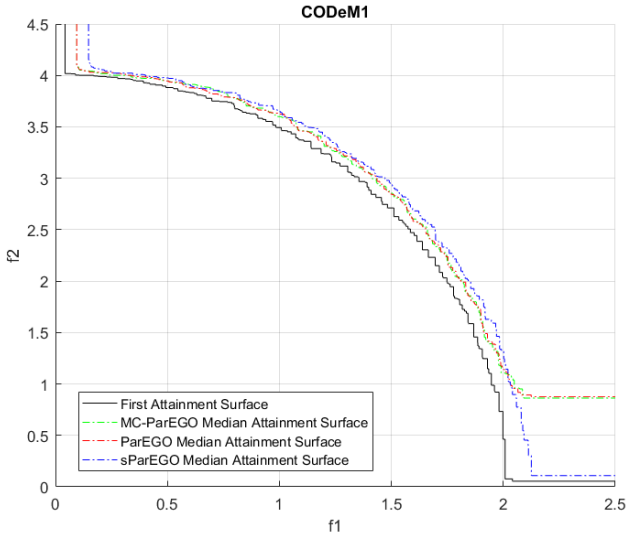
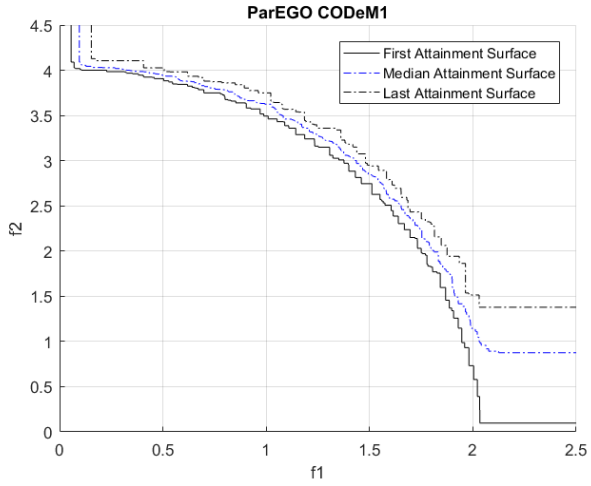


Figure 4.5: CODeM1 median attainment surfaces.

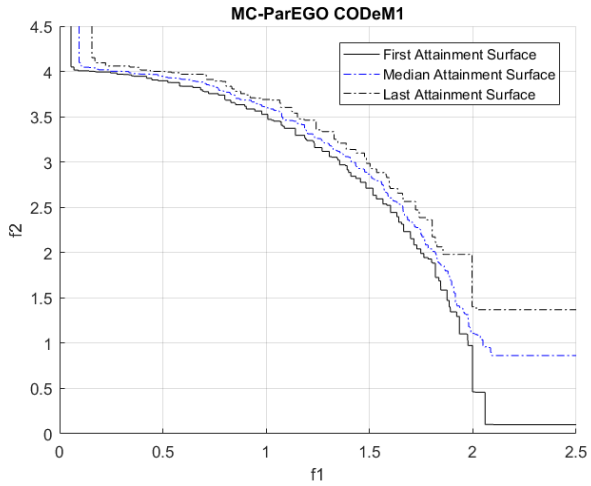
In Figure 4.7 is shown the median hypervolume indicator for CODeM2 test problem. Here, the algorithms perform in a similar fashion as in the first test problem. This is because of sParEGO's stochastic properties which makes the algorithm to struggle in the beginning. Hence, the converge to the front is slower in the first part of the run. Once the run passes the 500 function evaluations point the algorithms manages to obtain better spread of the entire Pareto front. This is also confirmed by looking at the Figure 4.8 and Figure 4.9. Here the non-dominated solutions and the median attainment surface shows that ParEGO and MC-ParEGO manage to provide better convergence towards the Pareto front on the top left side but sParEGO provides better spread of the front on the bottom right side.

Figure 4.11 shows the median hypervolume indicator values for CODeM3 test problem. Here the sParEGO algorithm provides solutions that struggle with convergence and only on the last 250 function evaluations these solutions converge to the Pareto front outperforming the two modified versions of ParEGO. This can be down to sParEGO's robustness indicator which normally will need a higher budget when regions of high uncertainty are discovered. Nevertheless, when looking at Figure 4.12 and Figure 4.13 it can be seen that sParEGO algorithm provides the same performance as in the first two test problems. It manages to get better spread of the Pareto front on the lower right side.

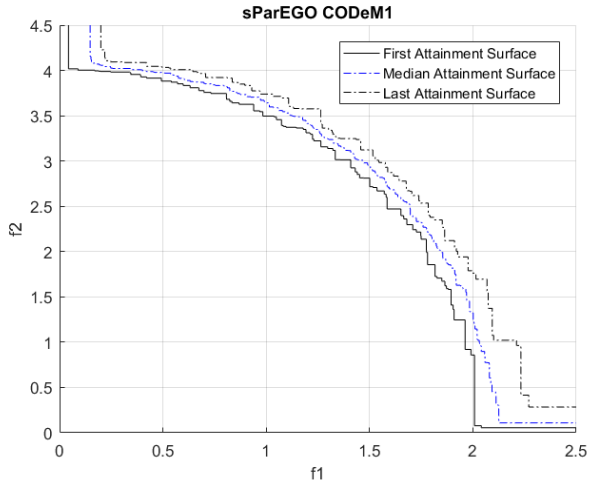
In Figure 4.15 the median hypervolume value can be observed for CODeM4 test problem. It can be noticed that sParEGO algorithm provides better convergence in the first 400 function evaluations than ParEGO. But overall sParEGO provides the worse convergence towards the Pareto front, this can also be observed in Figure 4.16. Looking at the attainment surface in Figure 4.17, it can be seen that both versions of ParEGO provide very similar results and that sParEGO only manages to converge better towards the edges of the Pareto front. The MC-ParEGO algorithm provides the best performance with the fastest convergence ratio.



(a)



(b)



(c)

Figure 4.6: CODeM1 EAF plots for each individual algorithm.

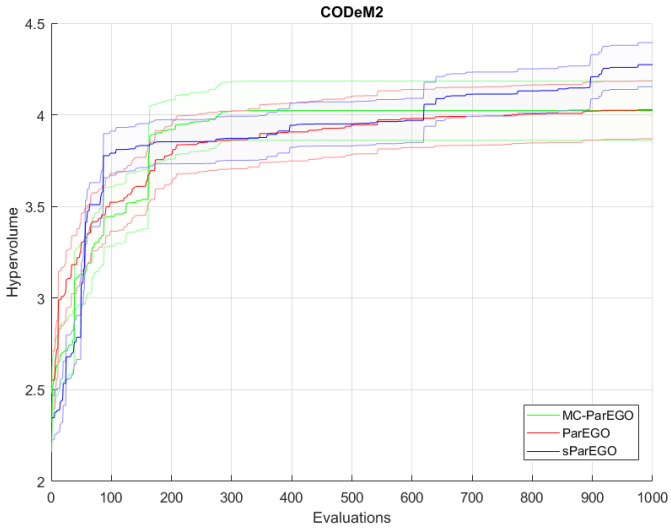


Figure 4.7: CODeM2 median run Hypervolume indicator.

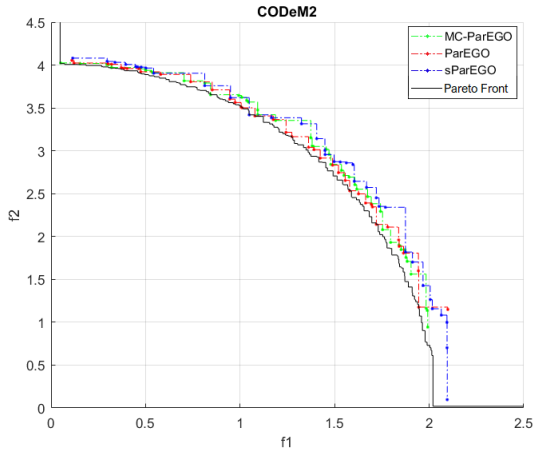


Figure 4.8: CODeM2 median run non dominated population.

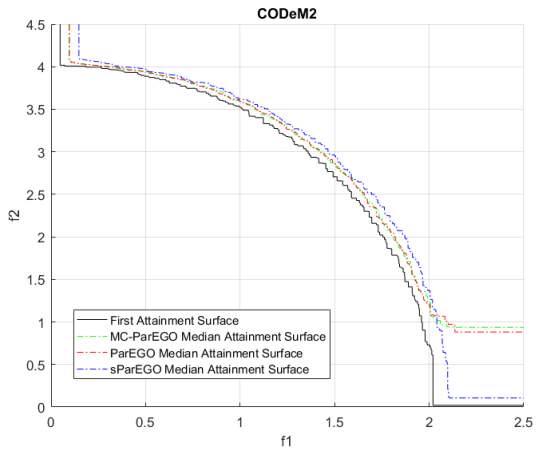
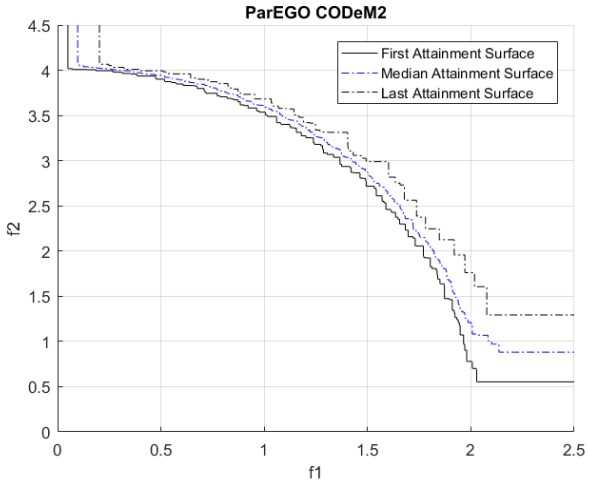
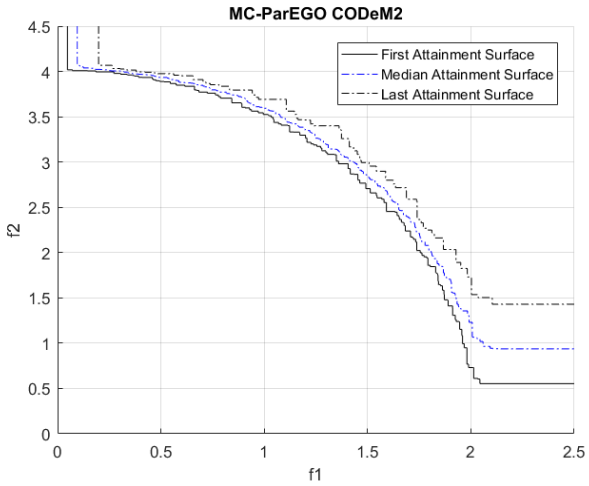


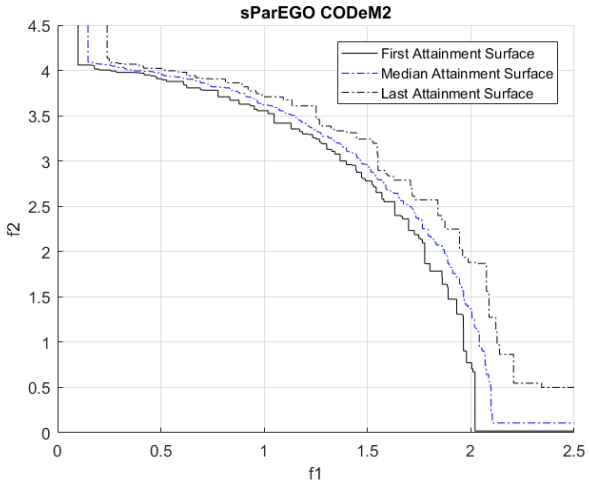
Figure 4.9: CODeM2 median attainment surfaces.



(a)



(b)



(c)

Figure 4.10: CODeM2 EAF plots for each individual algorithm.

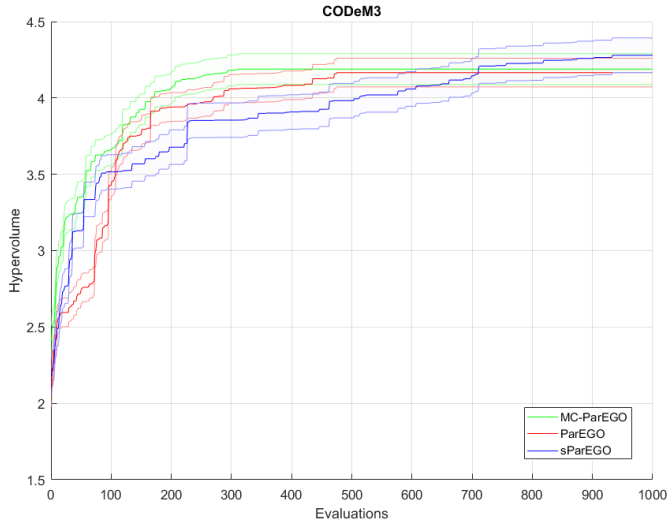


Figure 4.11: CODeM3 median run Hypervolume indicator.

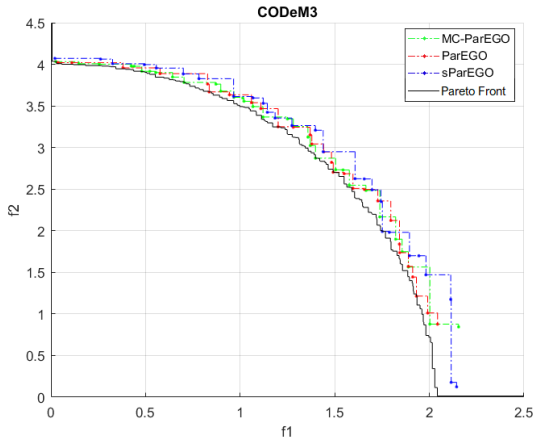


Figure 4.12: CODeM3 median run non dominated population.

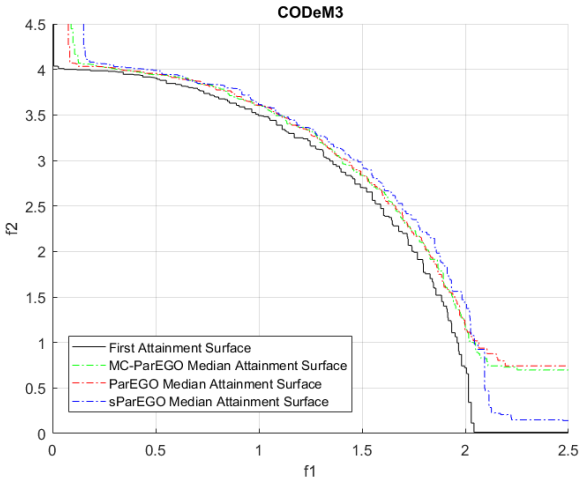
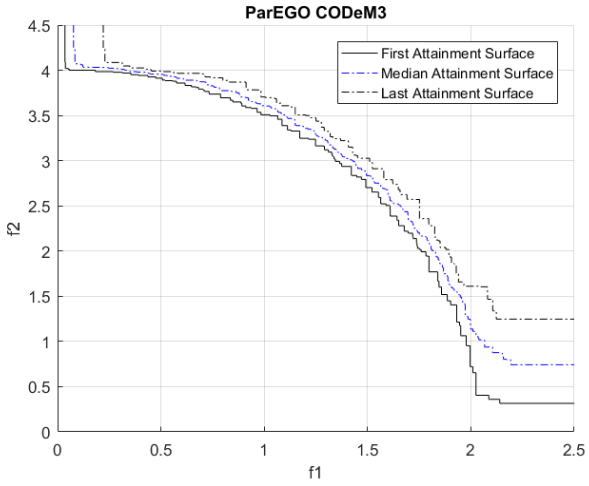
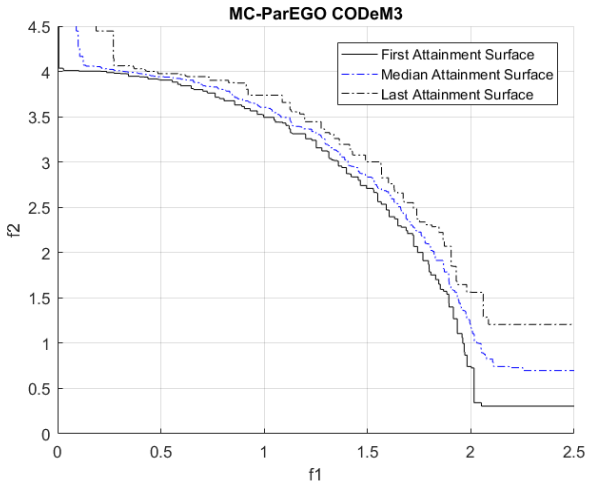


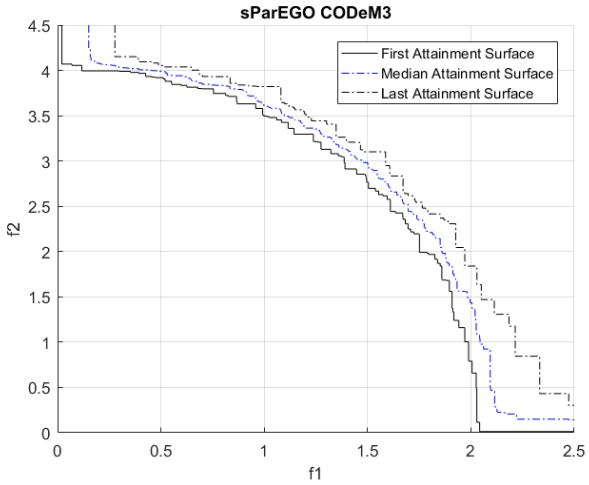
Figure 4.13: CODeM3 median attainment surfaces.



(a)



(b)



(c)

Figure 4.14: CODeM3 EAF plots for each individual algorithm.

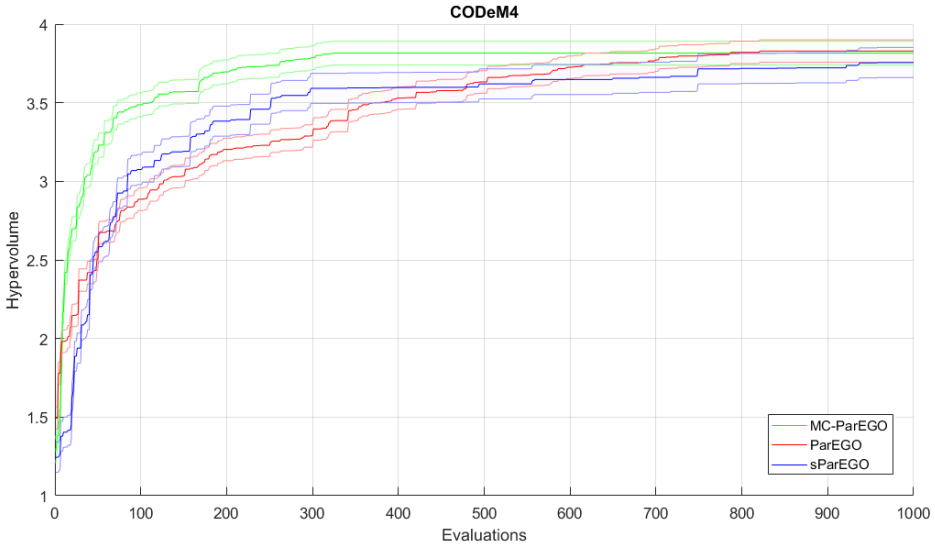


Figure 4.15: CODeM4 median run Hypervolume indicator.

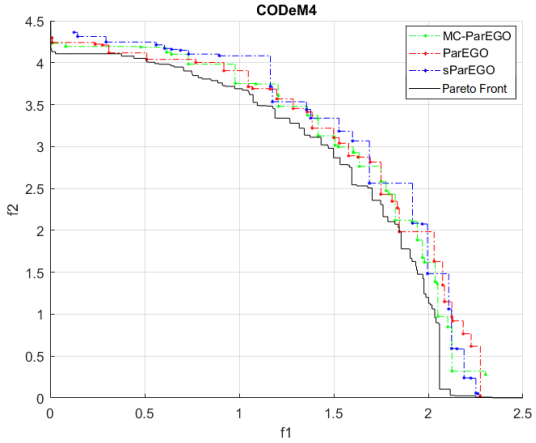


Figure 4.16: CODeM4 median run non dominated population.

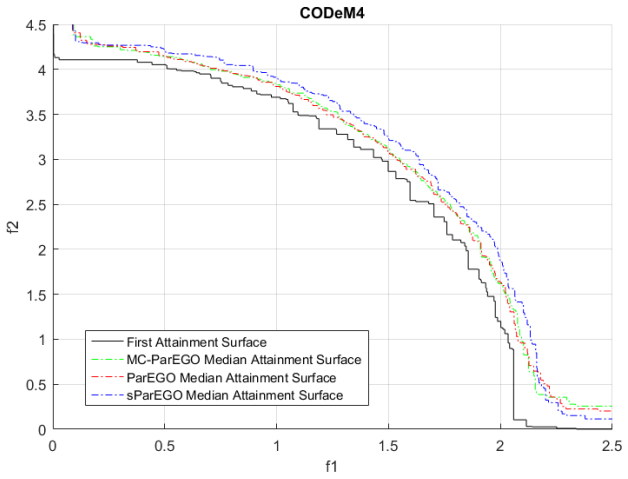
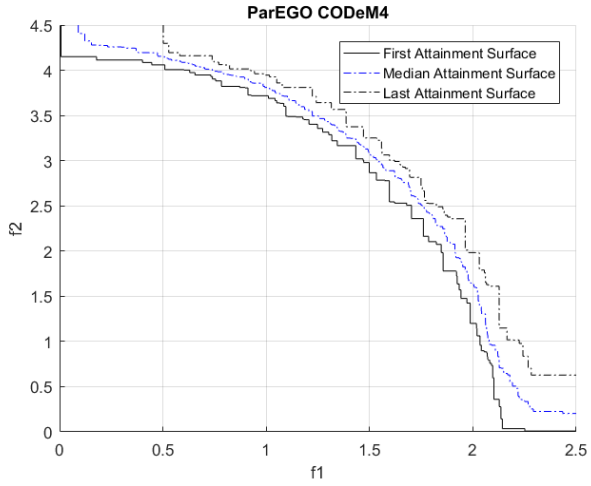
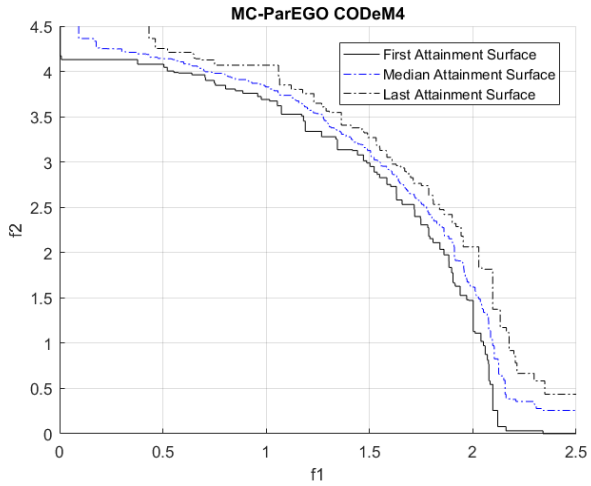


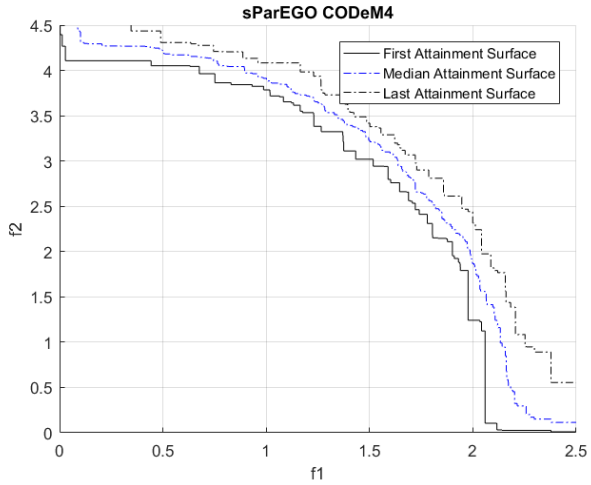
Figure 4.17: CODeM4 median attainment surfaces.



(a)



(b)



(c)

Figure 4.18: CODeM4 EAF plots for each individual algorithm.

In Figure 4.19 the median hypervolume indicator for CODeM5 test problem can be observed. An interesting change in this is that sParEGO algorithm provides promising results within the first half of the optimization run. Looking at Figure 4.20 it can be noticed that sParEGO has better exploration over the bottom right of the Pareto front but MC-ParEGO provides better convergence in the middle of the front. This is also confirmed by looking at the attainment surface in Figure 4.21.

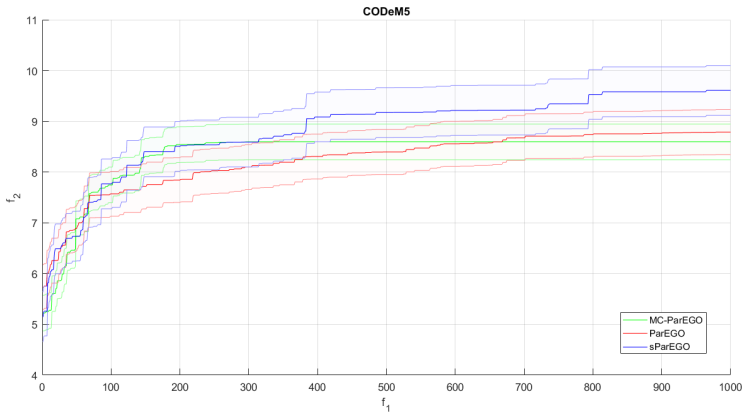


Figure 4.19: CODeM5 median run Hypervolume indicator.

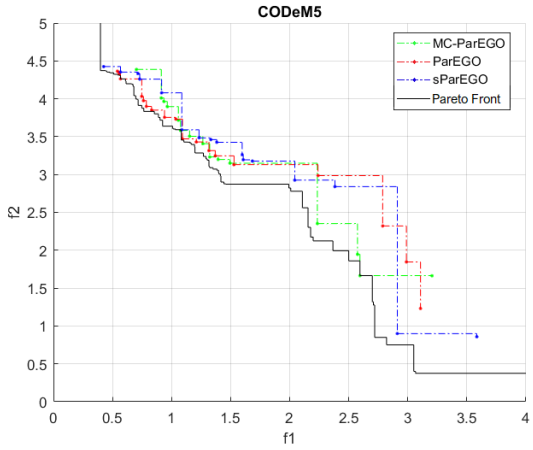


Figure 4.20: CODeM5 median run non dominated population.

Figure 4.23 shows the median hypervolume indicator for CODeM6 test problem. Here it can be seen that both versions of ParEGO obtain better performance for the majority of the optimization run. sParEGO shows slower convergence compared with the other two algorithms but towards the end of the optimization run sParEGO manages to provide better robust solutions. Looking at Figure 4.24 and the attainment surface in Figure 4.25 it can be seen that sParEGO converges faster towards the edges of the Pareto front but with similar solutions as the other 2 algorithms in the middle of the front.

Table 4.2 presents the statistical comparison of the three algorithms by looking at the two sample *t* test. On CODeM1 test problem by conventional criteria the difference between the two

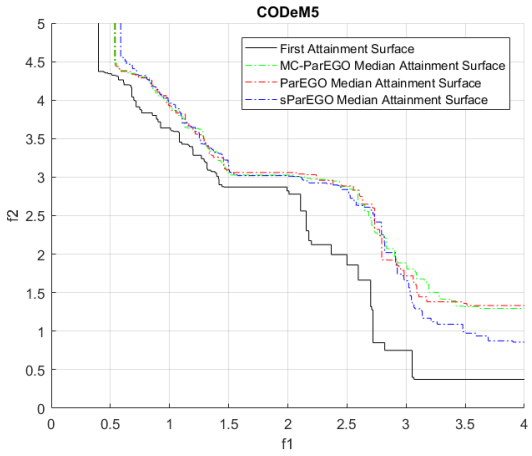


Figure 4.21: CODeM5 median attainment surfaces.

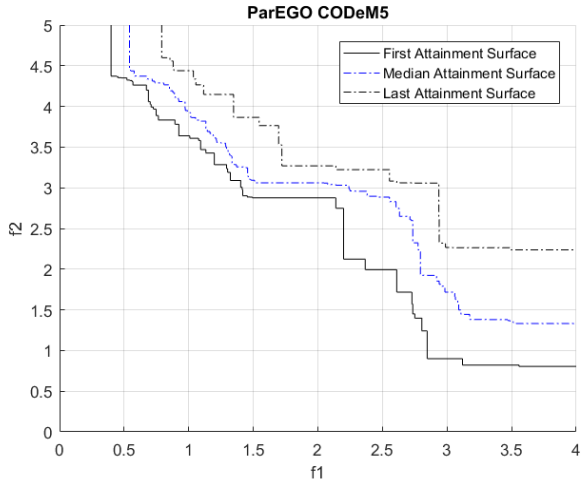
Table 4.2: p-values obtained by the two-sample t test. The p value (significance level of 5%) indicates rejection of the null hypothesis that two samples being compared have equal medians. A h value of zero supports that there is not enough evidence to reject the null hypothesis.

Test Problems	**MC-ParEGO vs ParEGO**	**ParEGO vs sParEGO**	**MC-ParEGO vs sParEGO**
CODeM1	0.9673	0.0030	0.0022
CODeM2	0.9342	0.0008	0.0011
CODeM3	0.5116	0.0227	0.0975
CODeM4	0.85181	0.0001	0.0001
CODeM5	0.6080	0.0001	0.0001
CODeM6	0.8678	0.5146	0.6370

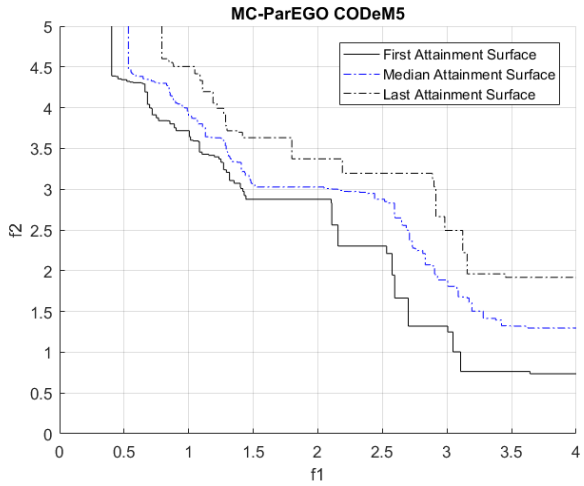
versions of ParEGO algorithms is considered not to be statistically significant. When compared with sParEGO both versions of ParEGO are considered to be statistically significant. This is also confirmed if the Willcox Ranksum is computed as follows: when comparing ParEGO with MC-ParEGO the statistical values are $p = 1, h = 0$ and $ranksum = 451$; when ParEGO is compared with sParEGO the values are $p = 0.003, h = 1$ and $ranksum = 333$; and when looking at sParEGO vs. MC-ParEGO the values are $p = 0.0028, h = 1$ and $ranksum = 332$.

For CODeM2 test problem by conventional criteria the difference between the two versions of ParEGO is considered not to be statistically significant. When sParEGO is compared with both versions of ParEGO the results are considered to be statistically significant. This can also be confirmed when the Willcox Ranksum is computed as follows: when comparing ParEGO with MC-ParEGO the statistical values are $p = 0.8209, h = 0$ and $ranksum = 461$; when ParEGO is compared with sParEGO the values are $p = 0.0017, h = 1$ and $ranksum = 326$; and when looking at sParEGO vs. MC-ParEGO the values are $p = 0.0018, h = 1$ and $ranksum = 327$.

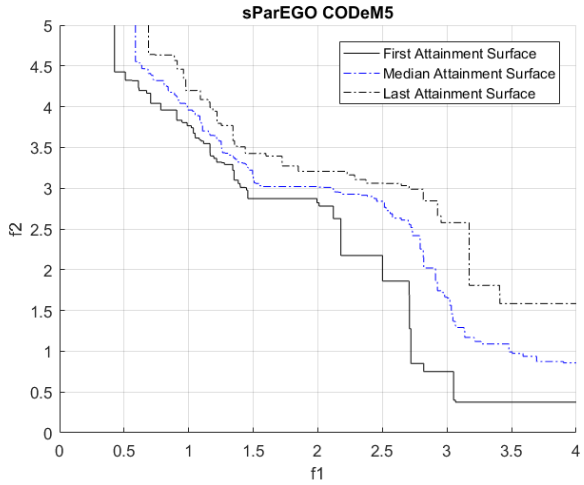
For CODeM3 test problem by conventional criteria the difference between the two versions



(a)



(b)



(c)

Figure 4.22: CODeM5 EAF plots for each individual algorithm.

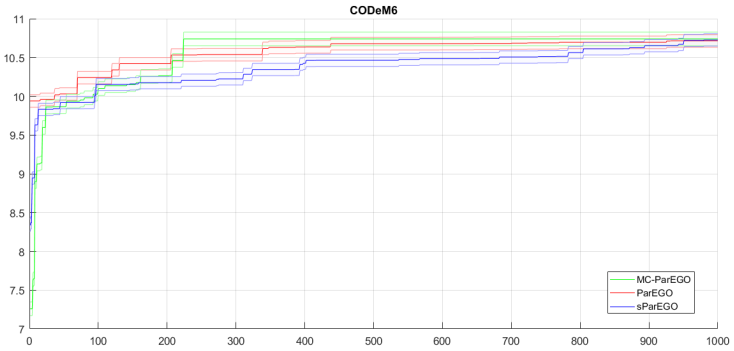


Figure 4.23: CODeM6 median run Hypervolume indicator.

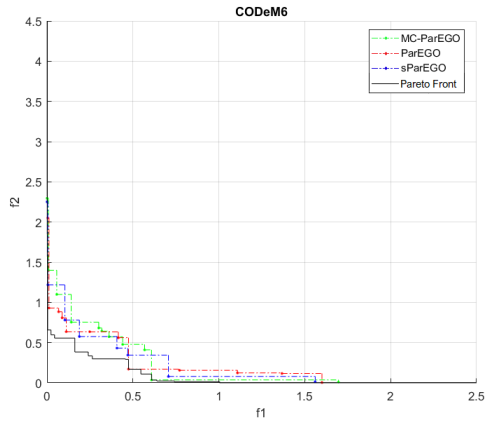


Figure 4.24: CODeM6 median run non dominated population.

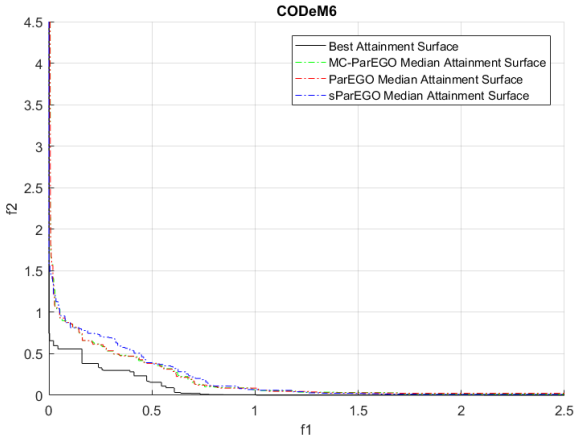
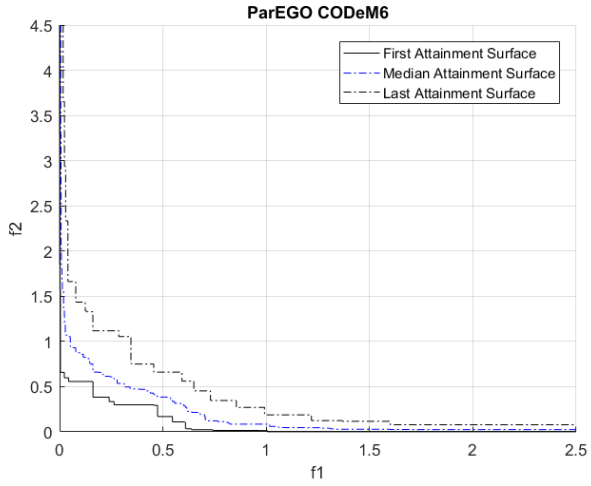
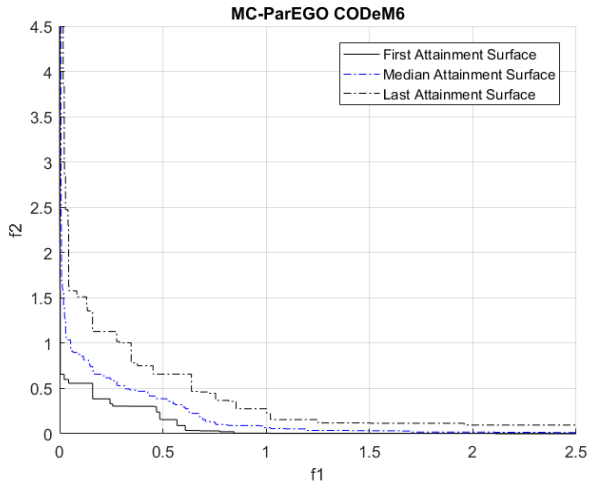


Figure 4.25: CODeM6 median attainment surfaces.

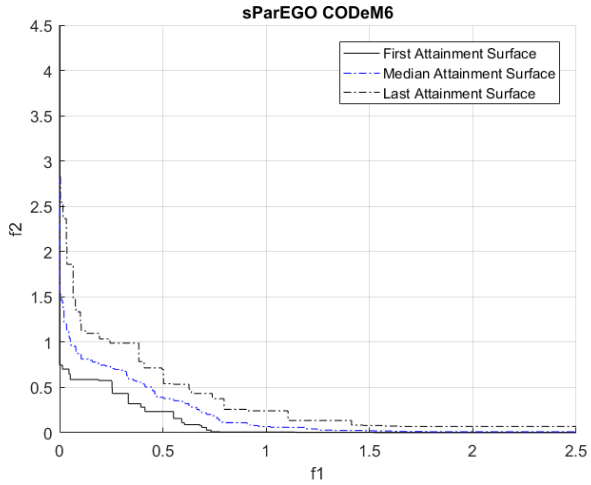
of ParEGO is considered not to be statistically significant. The same applies when comparing sParEGO and MC-ParEGO rezults. When comparing sParEGO with ParEGO results these are considered to be statistically significant. These are also confirmed if the Willcox Ranksum is computed as follows: when comparing ParEGO with MC-ParEGO the statistical values are $p = 0.7436, h = 0$ and $ranksum = 465$; when ParEGO is compared with sParEGO the values are $p = 0.0096, h = 1$ and $ranksum = 348$; and when looking at sParEGO vs. MC-ParEGO the



(a)



(b)



(c)

Figure 4.26: CODeM6 EAF plots for each individual algorithm.

values are $p = 0.0325, h = 1$ and $ranksum = 366$.

For CODeM4 test problem by conventional criteria the difference between the two versions of ParEGO is considered not to be statistically significant. When the results of sParEGO are compared with both versions of ParEGO it can be seen that they are extremely statistically significant. This is also confirmed if the Willcox Ranksum is computed as follows: when comparing ParEGO with MC-ParEGO the statistical values are $p = 0.9799, h = 0$ and $ranksum = 450$; when ParEGO is compared with sParEGO the values are $p = 1.1092e-07, h = 1$ and $ranksum = 663$; and when looking at sParEGO vs. MC-ParEGO the values are $p = 1.6733e-07, h = 1$ and $ranksum = 660$.

For CODeM5 test problem by conventional criteria the difference between the two versions of ParEGO is considered not to be statistically significant. When comparing sParEGO with both versions of ParEGO the results are considered to be extremely statistically significant. This is also confirmed if the Willcox Ranksum is compute as follows: when comparing ParEGO with MC-ParEGO the statistical values are $p = 0.6873, h = 0$ and $ranksum = 435$; when ParEGO is compared with sParEGO the values are $p = 0.10714, h = 0$ and $ranksum = 387$; and when looking at sParEGO vs. MC-ParEGO the values are $p = 00325, h = 1$ and $ranksum = 366$.

For CODeM6 test problem by conventional criteria the difference between the two versions of ParEGO and sParEGO is considered not to be statistically significant. Hence, all three algorithms are seen to provide the same performance. This is also confirmed if the Willcox Ranksum is computed as follows: when comparing ParEGO with MC-ParEGO the statistical values are $p = 0.9198, h = 0$ and $ranksum = 447$; when ParEGO is compared with sParEGO the values are $p = 0.5131, h = 0$ and $ranksum = 478$; and when looking at sParEGO vs. MC-ParEGO the values are $p = 0.3924, h = 0$ and $ranksum = 486$.

4.4 Discussion

Looking at these results it can be concluded that sParEGO outperformed both variants of ParEGO on 4 out of 6 test problems. Only on 3 off these instances sParEGO algorithm provided Pareto optimal solutions that are statistical significant. As it can be seen in the figures above sParEGO manages to provide better approximation of the Pareto front by providing better diversity in the final population. By looking at these results it can be said that in some cases if the problem at hand has a lower optimization budget (e.g 250 or 500 function evaluations) it is better to apply a Monte Carlo sampling method in order to obtain robust Pareto optimal sets.

Chapter 5

Conclusion and Future Work

The aim of this research was to benchmark the surrogate based multi-objective optimization algorithms available in the Tigon library using the Liger optimization software. This helped in testing the effectiveness of algorithms on robust optimization problems. The main focus was to analyse the sParEGO algorithm performance and compare this with Monte Carlo based alternatives on a variety of optimization problems. This work also presented how easy it was to use Liger optimization platform to implement a new indicator based multi-objective optimization algorithm, namely SMS-EMOA, from literature into the Tigon library.

One key contribution in this work is the implementation of the SMS-EMOA algorithm in Tigon, this was successfully verified using the set of problems from the original paper. For a fair comparison the same set-up configuration was kept as in the paper. It can be seen that Liger provides an easy to use framework where users can test optimization problems from within the Tigon library or by importing their own functions. The reason why Liger was used in this work is because of its re-usability and easy learning curve. This makes it ideal for industry and non-experts in the field of optimization. SMS-EMOA is now available to practitioners in the latest version (v1.4.0) that was released in 2021 under the LGPL licence. This is available as an open-source code in GitHub under: https://github.com/ligerdev/liger.

The second contribution is the comparative analysis of benchmarking tests, which showed that sParEGO algorithm achieved a slightly better statistical performance compared to the MC-based alternatives. This advantage stems from sParEGO's specialized capability to assess the robustness of a candidate solution based on neighbouring solutions. However, due to sParEGO's robustness indicator and parameter settings it can be seen that on some problems with a limited optimization budget (in terms of function evaluations) it is better to use Monte Carlo sampling based alternatives. In this work MC-ParEGO algorithm showed promising robust Pareto optimal solutions. As described above in Chapter 4 the benchmarking problems were divided into four types of test problems.

CODeM6 is based on the DTLZ1 test problem which is an affine, separable and unimodal

problem. This is a relatively simple problem for which all algorithms performed well and statistically it can not be said that one outperformed the other.

CODeM1 to CODeM3 problems are based on the WFG4 test problem which is a separable, multimodal and concave problem. The uncertainty presented here is radial, perpendicular or in both directions. On these types of problems sParEGO algorithm outperforms the other versions of ParEGO algorithm, indicating that its approach to estimating robustness is performing well.

CODeM4 is based on the WFG6 test problem which is a non-separable, multimodal and concave problem. This problem is complex and has uncertainty in both directions. In this case the MC-ParEGO algorithm outperforms sParEGO. The key distinction here lies in the non-separability of the problem.

CODeM5 is based on WFG8 test problem which is a non-separable, multimodal, concave and biased problem. This problem has greater complexity compared to the previous ones as it has uncertainty in both directions. For this the sParEGO is seen to exhibit the best performance.

To gain insight into these findings, first consider how the features of the problems interact with the components of the algorithms. sParEGO infers the uncertainty of a solution by computing variability in the neighbourhood. This variability arises from changes to the objective function values (i.e. the shape of the landscape) or from uncertainty. For smooth problems, the variability in the neighbourhood is more likely to arise from uncertainty. In these conditions (i.e. CODeM1-3), sParEGO's uncertainty estimation works well. However, for problems where the landscape is not smooth (i.e. due to non-separability), the local variability in objective values may be high and this will act to confound sParEGO's estimates. This outcome is seen for CODeM4. Interestingly, it is not observed in case of CODeM5 (which is non-separable but also contains bias). Further work is required to understand the complex interaction of bias, non-separability and multimodality for this problem. In all cases, MC-ParEGO does not confuse uncertainty with variance in objective values across the search space. Its estimates are of good quality (subject to the number of samples) but this sampling reduces the budget to explore the space and therefore it will take longer to converge towards the Pareto front.

In conclusion, this study considered which algorithm out of the three would be better to implement, ParEGO (not accounting for uncertainty), Monte Carlo or sParEGO. It is highly recommended to take uncertainty into consideration. In terms of MC vs sParEGO, more in-depth research is required which is not in the scope of this work. The benchmarking conducted has shown that sParEGO generally performs better but the scope of this benchmarking was limited. It can be said that as the problem complexity increases (e.g. bias) it provides a satisfying outcome for sParEGO algorithm. However, if the problem is non-separable then it is better to implement Monte Carlo and search for robust performance. This is because the idea of neighbourhood is more prohibited and since sParEGO was designed to assess neighbouring solutions for robustness and relies on having neighbourhood evaluations. The results indicate that the sParEGO algorithm's performance is conditioned on the neighbourhood.

In terms of the limitations, the current stochastic test problems, CODeM1 to CODeM6 are not sufficiently comprehensive to avoid confounding when trying to test hypotheses about performance. There is a need to extend towards more problem instances with greater variety of features and more systematic adjustments. Upon examining the artificial structures provided by the WFG and DTLZ toolkits, it becomes evident that many optimization algorithms have been specifically designed to perform well on these types of problems. An implicit bias that is often overlooked, as these benchmarking problems are not representative of the real world problems.

5.1 Ideas for future work

- It would be beneficial to incorporate more modern benchmarking methodologies, e.g. COCO platform introduced by Hansen et al. (2016) – http://numbbo.github.io/coco/, in Tigon in order to have more instances of each problem and avoid training biases.
- Test whether these algorithms can solve unseen problem instances. Also, consider generalization of classes of problems (e.g classes of unseen problems).
- Another avenue of future work could be to extend surrogate based algorithms to deal with constrained multi-objective optimization problems in order to solve challenging stochastic real world optimization problems.
- Investigate robustness extensions for other types of multi-objective Bayesian optimizers as discussed by Daulton et al. (2022) or by Garrido-Merchán & Hernández-Lobato (2019).
- Even though naive Monte Carlo sampling has been used in this benchmarking, one area of improvement would be to implement better sampling techniques in Tigon (e.g as discussed in Dong & Nakayama (2020)). Then benchmark ParEGO's robust versions against the sParEGO algorithm in order to assess the performance and see which algorithm provides a better approximation of the robust Pareto front.

Bibliography

Akhtar, T. & Shoemaker, C. A. (2016), 'Multi objective optimization of computationally expensive multi-modal functions with RBF surrogates and multi-rule selection', *Journal of Global Optimization* **64**(1), 17–32.

Audet, C., Denni, J., Moore, D., Booker, A. & Frank, P. (2000), A surrogate-model-based method for constrained optimization, *in* '8th symposium on multidisciplinary analysis and optimization', p. 4891.

Bader, J. & Zitzler, E. (2011), 'Hype: An algorithm for fast hypervolume-based many-objective optimization', *Evolutionary computation* **19**(1), 45–76.

Bean, J. C. (1994), 'Genetic algorithms and random keys for sequencing and optimization', *ORSA journal on computing* **6**(2), 154–160.

Beume, N., Naujoks, B. & Emmerich, M. (2007), 'Sms-emoa: Multiobjective selection based on dominated hypervolume', *European Journal of Operational Research* **181**(3), 1653–1669.

Beyer, H. G. & Sendhoff, B. (2007*a*), 'Robust optimization - A comprehensive survey', *Computer Methods in Applied Mechanics and Engineering* **196**(33-34), 3190–3218.

Beyer, H.-G. & Sendhoff, B. (2007*b*), 'Robust optimization–a comprehensive survey', *Computer methods in applied mechanics and engineering* **196**(33-34), 3190–3218.

Camponogara, E. & Talukdar, S. N. (1997), A genetic algorithm for constrained and multiobjective optimization, *in* '3rd Nordic workshop on genetic algorithms and their applications (3NWGA)', pp. 49–62.

Chugh, T., Sindhya, K., Miettinen, K., Hakanen, J. & Jin, Y. (2016), On constraint handling in surrogate-assisted evolutionary many-objective optimization, *in* 'Parallel Problem Solving from Nature–PPSN XIV: 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings 14', Springer, pp. 214–224.

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

Couckuyt, I., Deschrijver, D. & Dhaene, T. (2014), 'Fast calculation of multiobjective probability of improvement and expected improvement criteria for pareto optimization', *Journal of Global Optimization* **60**, 575–594.

Daulton, S., Cakmak, S., Balandat, M., Osborne, M. A., Zhou, E. & Bakshy, E. (2022), Robust multi-objective bayesian optimization under input noise, *in* 'International Conference on Machine Learning', PMLR, pp. 4831–4866.

Davidor, Y. (1991), 'A genetic algorithm applied to robot trajectory generation', *Handbook of genetic algorithms* pp. 144–165.

Davins-Valldaura, J., Moussaoui, S., Pita-Gil, G. & Plestan, F. (2017), 'Parego extensions for multi-objective optimization of expensive evaluation functions', *Journal of Global Optimization* **67**, 79–96.

Deb, K. & Gupta, H. (2005), 'Searching for robust Pareto-optimal solutions in multi-objective optimization', *Evolutionary multi-criterion optimization* pp. 150–164.
URL: *http://www.springerlink.com/index/49PXTAQNWEKW6EYB.pdf*

Deb, K. & Gupta, H. (2006), 'Introducing Robustness in Multi-Objective Optimization', *Evolutionary Computation* **14**(4), 463–494.
URL: *http://www.mitpressjournals.org/doi/pdf/10.1162/evco.2006.14.4.463*

Deb, K., Mohan, M. & Mishra, S. (2003*a*), 'A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions', *KanGAL report* **2003002**, 1–18.

Deb, K., Mohan, M. & Mishra, S. (2003*b*), Towards a quick computation of well-spread pareto-optimal solutions, *in* 'Evolutionary Multi-Criterion Optimization: Second International Conference, EMO 2003, Faro, Portugal, April 8–11, 2003. Proceedings 2', Springer, pp. 222–236.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. (2002), 'A fast and elitist multiobjective genetic algorithm: NSGA-II', *IEEE Transactions on Evolutionary Computation* **6**(2), 182–197.

Dong, H. & Nakayama, M. K. (2020), 'A tutorial on quantile estimation via monte carlo', *Monte Carlo and Quasi-Monte Carlo Methods: MCQMC 2018, Rennes, France, July 1–6* pp. 3–30.

Durillo, J. J. & Nebro, A. J. (2011), 'jmetal: A java framework for multi-objective optimization', *Advances in engineering software* **42**(10), 760–771.

Duro, J. A., Oara, D. C., Sriwastava, A. K., Yan, Y., Salomon, S. & Purshouse, R. C. (2021), Component-based design of multi-objective evolutionary algorithms using the tigon optimization library, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference Companion', pp. 1531–1539.

Duro, J. A., Ozturk, U. E., Oara, D. C., Salomon, S., Lygoe, R. J., Burke, R. & Purshouse, R. C. (2023), 'Methods for constrained optimization of expensive mixed-integer multi-objective problems, with application to an internal combustion engine design problem', *European Journal of Operational Research* **307**(1), 421–446.

Duro, J. A., Purshouse, R. C., Salomon, S., Oara, D. C., Kadirkamanathan, V. & Fleming, P. J. (2019), Sparego–a hybrid optimization algorithm for expensive uncertain multi-objective optimization problems, *in* 'Evolutionary Multi-Criterion Optimization: 10th International Conference, EMO 2019, East Lansing, MI, USA, March 10-13, 2019, Proceedings 10', Springer, pp. 424–438.

Duro, J. A., Yan, Y., Giagkiozis, I., Giagkiozis, S., Salomon, S., Oara, D. C., Sriwastava, A. K., Morison, J., Freeman, C. M., Lygoe, R. J. et al. (2021), 'Liger: A cross-platform open-source integrated optimization and decision-making environment', *Applied Soft Computing* **98**, 106851.

Ehrgott, M., Ide, J. & Schöbel, A. (2014), 'Minmax robustness for multi-objective optimization problems', *European Journal of Operational Research* **239**(1), 17–31.

Emmerich, M. (2005), 'Single-and multi-objective evolutionary design optimization assisted by gaussian random field metamodels', *University of Dormund* .

Emmerich, M., Beume, N. & Naujoks, B. (2005), An emo algorithm using the hypervolume measure as selection criterion, *in* 'International Conference on Evolutionary Multi-Criterion Optimization', Springer, pp. 62–76.

Emmerich, M. & Klinkenberg, J.-w. (2008), 'The computation of the expected improvement in dominated hypervolume of pareto front approximations', *Rapport technique, Leiden University* **34**, 7–3.

Emmerich, M. T., Giannakoglou, K. C. & Naujoks, B. (2006), 'Single-and multiobjective evolutionary optimization assisted by gaussian random field metamodels', *IEEE Transactions on Evolutionary Computation* **10**(4), 421–439.

Engelbrecht, A. P. (2007), *Computational intelligence: an introduction*, John Wiley & Sons.

Forrester, A. I. & Keane, A. J. (2009), 'Recent advances in surrogate-based optimization', *Progress in aerospace sciences* **45**(1-3), 50–79.

Gamma, E. (1995), 'Design patterns: elements of reusable object-oriented software', *Person Education Inc* .

Garrido-Merchán, E. C. & Hernández-Lobato, D. (2019), 'Predictive entropy search for multi-objective bayesian optimization with constraints', *Neurocomputing* **361**, 50–68.

Giagkiozis, I., Lygoe, R. J. & Fleming, P. J. (2013), 'Liger: an open source integrated optimization environment', pp. 1089–1096.

Guo, L., Hao, J.-h. & Liu, M. (2014), 'An incremental extreme learning machine for online sequential learning problems', *Neurocomputing* **128**, 50–58.

Hadka, D. et al. (2015), 'Moea framework: a free and open source java framework for multiobjective optimization', *Version* **2**, 74.

Hakanen, J. & Knowles, J. D. (2017), On using decision maker preferences with parego, *in* 'Evolutionary Multi-Criterion Optimization: 9th International Conference, EMO 2017, Münster, Germany, March 19-22, 2017, Proceedings 9', Springer, pp. 282–297.

Hansen, N., Auger, A., Mersmann, O., Tušar, T. & Brockhoff, D. (2016), 'Coco: A platform for comparing continuous optimizers in a black-box setting. arxiv e-prints', *arXiv preprint arXiv:1603.08785* **172**.

Hao, J.-h. & Liu, M. (2014), A surrogate modelling approach combined with differential evolution for solving bottleneck stage scheduling problems, *in* '2014 world automation congress (WAC)', IEEE, pp. 120–124.

Horn, D., Wagner, T., Biermann, D., Weihs, C. & Bischl, B. (2015), Model-based multiobjective optimization: taxonomy, multi-point proposal, toolbox and benchmark, *in* 'International Conference on Evolutionary Multi-Criterion Optimization', Springer, pp. 64–78.

Horn, J., Nafpliotis, N. & Goldberg, D. E. (1994), A niched pareto genetic algorithm for multiobjective optimization, *in* 'Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence', Ieee, pp. 82–87.

Huband, S., Hingston, P., Barone, L. & While, L. (2006), 'A review of multiobjective test problems and a scalable test problem toolkit', *IEEE Transactions on Evolutionary Computation* **10**(5), 477–506.

Hughes, E. J. (2007), Msops-ii: A general-purpose many-objective optimiser, *in* '2007 IEEE Congress on Evolutionary Computation', IEEE, pp. 3944–3951.

Hussein, R. & Deb, K. (2016), 'A Generative Kriging Surrogate Model for Constrained and Unconstrained Multi-objective Optimization', *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference - GECCO '16* pp. 573–580.
URL: *http://dl.acm.org/citation.cfm?doid=2908812.2908866*

Ishibuchi, H., Doi, T. & Nojima, Y. (2006), Incorporation of scalarizing fitness functions into evolutionary multiobjective optimization algorithms, *in* 'Parallel Problem Solving from Nature-PPSN IX: 9th International Conference, Reykjavik, Iceland, September 9-13, 2006, Proceedings', Springer, pp. 493–502.

Jones, D. R., Schonlau, M. & Welch, W. J. (1998), 'Efficient global optimization of expensive black-box functions', *Journal of Global optimization* **13**, 455–492.

Kalsi, M., Hacker, K. & Lewis, K. (1999), A comprehensive robust design approach for decision trade-offs in complex systems design, *in* 'International Design Engineering Technical Conferences and Computers and Information in Engineering Conference', Vol. 19715, American Society of Mechanical Engineers, pp. 1343–1354.

Keijzer, M., Merelo, J. J., Romero, G. & Schoenauer, M. (2001), Evolving objects: A general purpose evolutionary computation library, *in* 'International Conference on Artificial Evolution (Evolution Artificielle)', Springer, pp. 231–242.

Klempous, R., Nikodem, J., Jacak, W. & Chaczko, Z. (2013), *Advanced methods and applications in computational intelligence*, Vol. 6, Springer Science & Business Media.

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

Kowalczyk, R. (1997), Constraint consistent genetic algorithms, *in* 'Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC'97)', IEEE, pp. 343–348.

Kronfeld, M., Planatscher, H. & Zell, A. (2010), The eva2 optimization framework, *in* 'International Conference on Learning and Intelligent Optimization', Springer, pp. 247–250.

Liefooghe, A., Jourdan, L. & Talbi, E.-G. (2011), 'A software framework based on a conceptual unified model for evolutionary multiobjective optimization: Paradiseo-moeo', *European Journal of Operational Research* **209**(2), 104–112.

Lim, D., Ong, Y.-S., Jin, Y. & Sendhoff, B. (2007), A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation, *in* 'Proceedings of the 9th annual conference on Genetic and evolutionary computation', pp. 1288–1295.

Lim, D., Ong, Y. S., Jin, Y., Sendhoff, B. & Lee, B. S. (2006), 'Inverse multi-objective robust evolutionary design', *Genetic Programming and Evolvable Machines* **7**(4), 383–404.

Lu, H. & Chen, W. (2008), 'Self-adaptive velocity particle swarm optimization for solving constrained optimization problems', *Journal of Global Optimization* **41**(3), 427–445.

Lukasiewycz, M., Glaß, M., Reimann, F. & Teich, J. (2011), Opt4j: a modular framework for meta-heuristic optimization, *in* 'Proceedings of the 13th annual conference on Genetic and evolutionary computation', pp. 1723–1730.

Luke, S. (2017), Ecj then and now, *in* 'Proceedings of the genetic and evolutionary computation conference companion', pp. 1223–1230.

Michalewicz, Z. & Nazhiyath, G. (1995), Genocop iii: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints, *in* 'Proceedings of 1995 IEEE International Conference on Evolutionary Computation', Vol. 2, IEEE, pp. 647–651.

Mourelatos, Z. P. & Liang, J. (2006), 'A methodology for trading-off performance and robustness under uncertainty'.

Murata, T., Ishibuchi, H. et al. (1995), Moga: multi-objective genetic algorithms, *in* 'IEEE international conference on evolutionary computation', Vol. 1, IEEE Piscataway, pp. 289–294.

Nebro, A. J., Durillo, J. J. & Vergne, M. (2015), Redesigning the jmetal multi-objective optimization framework, *in* 'Proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation', pp. 1093–1100.

Paenke, I., Branke, J. & Jin, Y. (2006), 'Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation', *IEEE Transactions on Evolutionary Computation* **10**(4), 405–420.

Pavelski, L. M., Delgado, M. R., Almeida, C. P., Gon??alves, R. A. & Venske, S. M. (2016), 'Extreme Learning Surrogate Models in Multi-objective Optimization based on Decomposition', *Neurocomputing* **180**, 55–67.
URL: *http://dx.doi.org/10.1016/j.neucom.2015.09.111*

Pilát, M. & Neruda, R. (2012), Meta-learning and model selection in multi-objective evolutionary algorithms, *in* '2012 11th International Conference on Machine Learning and Applications', Vol. 1, IEEE, pp. 433–438.

PISA, A. (2013), 'Platform and programming language independent interface for search algorithms'.

Purshouse, R. C. & Fleming, P. J. (2007), 'On the evolutionary optimization of many conflicting objectives', *IEEE transactions on evolutionary computation* **11**(6), 770–784.

Rambeaux, F., Hamelin, F. & Sauter, D. (2000), 'Optimal thresholding for robust fault detection of uncertain systems', *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal* **10**(14), 1155–1173.

Ramírez, A., Romero, J. R. & Ventura, S. (2015), An extensible jclec-based solution for the implementation of multi-objective evolutionary algorithms, *in* 'proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation', pp. 1085–1092.

Rasmussen, C. E. (2003), Gaussian processes in machine learning, *in* 'Summer school on machine learning', Springer, pp. 63–71.

Ray, T., Asafuddoula, M., Singh, H. K. & Alam, K. (2015), 'An Approach to Identify Six Sigma Robust Solutions of Multi/Many-Objective Engineering Design Optimization Problems', *Journal of Mechanical Design* **137**(5), 051404.
URL: *http://mechanicaldesign.asmedigitalcollection.asme.org/article.aspx?doi=10.1115/1.4029704*

Salomon, S., Avigad, G., Fleming, P. J. & Purshouse, R. C. (2014), 'Active robust optimization: Enhancing robustness to uncertain environments', *IEEE Transactions on Cybernetics* **44**(11), 2221–2231.

Salomon, S., Avigad, G., Purshouse, R. C. & Fleming, P. J. (2016), 'Gearbox design for uncertain load requirements using active robust optimization', *Engineering Optimization* **48**(4), 652–671.
URL: *http://dx.doi.org/10.1080/0305215X.2015.1031659*

Salomon, S., Purshouse, R. C., Giaghiozis, I. & Fleming, P. J. (2016), A toolkit for generating scalable stochastic multiobjective test problems, *in* 'Proceedings of the Genetic and Evolutionary Computation Conference 2016', pp. 597–604.

Srinivas, N. & Deb, K. (1994), 'Muiltiobjective optimization using nondominated sorting in genetic algorithms', *Evolutionary computation* **2**(3), 221–248.

Taguchi, G. (1986), *Introduction to quality engineering: designing quality into products and processes*.

Tian, Y., Cheng, R., Zhang, X. & Jin, Y. (2017), 'Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum]', *IEEE Computational Intelligence Magazine* **12**(4), 73–87.

Trautmann, H., Rudolph, G., Klamroth, K., Schütze, O., Wiecek, M., Jin, Y. & Grimme, C. (2017), *Evolutionary Multi-Criterion Optimization: 9th International Conference, EMO 2017, Münster, Germany, March 19-22, 2017, Proceedings*, Vol. 10173, Springer.

Tutum, C. C., Deb, K. & Baran, I. (2015), 'Constrained efficient global optimization for pultrusion process', *Materials and Manufacturing Processes* **30**(4), 538–551.
URL: *http://dx.doi.org/10.1080/10426914.2014.994752*

Winter, R. J. (2014), 'Agile software development: Principles, patterns, and practices: Robert c. martin with contributions by james w. newkirk and robert s. koss'.

Wu, X. & Kozlowski, T. (2017), 'Inverse uncertainty quantification of reactor simulations under the Bayesian framework using surrogate models constructed by polynomial chaos expansion', *Nuclear Engineering and Design* **313**, 29–52.

Xiao, J., Michalewicz, Z., Zhang, L. & Trojanowski, K. (1997), 'Adaptive evolutionary planner/navigator for mobile robots', *IEEE transactions on evolutionary computation* **1**(1), 18–28.

Zapotecas Martínez, S. & Coello Coello, C. A. (2013), Moea/d assisted by rbf networks for expensive multi-objective optimization problems, *in* 'Proceedings of the 15th annual conference on Genetic and evolutionary computation', pp. 1405–1412.

Zbigniew, M. (1996), 'Genetic algorithms+ data structures= evolution programs', *Comput Stat* pp. 372–373.

Zhang, Q. & Li, H. (2007), 'Moea/d: A multiobjective evolutionary algorithm based on decomposition', *IEEE Transactions on evolutionary computation* **11**(6), 712–731.

Zhou, A., Qu, B.-Y., Li, H., Zhao, S.-Z., Suganthan, P. N. & Zhang, Q. (2011), 'Multiobjective evolutionary algorithms: A survey of the state of the art', *Swarm and evolutionary computation* **1**(1), 32–49.

Zitzler, E. & Künzli, S. (2004), Indicator-based selection in multiobjective search, *in* 'International conference on parallel problem solving from nature', Springer, pp. 832–842.

Zizler, E. (1998), 'Multiobjective optimization using evolutionary algorithms: A comparative study', *Parallel Problem Solving from Nature* pp. 292–301.