Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Marius Aarsnes & Erling Wisløff

Informed Push & Pull Search for Solving Constrained Multi-Objective Optimisation

Master's thesis in Master of Science in Informatics
Supervisor: Pauline Haddow
July 2020

NTNU
Norwegian University of Science and Technology

Marius Aarsnes & Erling Wisløff

Informed Push & Pull Search for Solving Constrained Multi-Objective Optimisation

Master's thesis in Master of Science in Informatics
Supervisor: Pauline Haddow
July 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

In recent years, Constrained Multi-Objective Problems (CMOPs) have gained a considerable focus by the research community as the existence of constraints provide challenges the field is struggling to solve. To overcome this obstacle, research has delved into the development of artificial problems to simulate their real-world counterparts and extending Multi-Objective Evolutionary Algorithms (MOEAs) with Constraint-Handling Methods (CHMs) to solve these problems. In general, CHMs within the field are influenced by information extracted from the current population alone. Thus, little work gathers knowledge during the search to influence the optimisation of the CMOP. Multiple objectives in conjunction with constraints add difficulty in converging towards optimal solutions, as well as coverage of all solutions fulfilling the constraints. Recent advances within this field introduced a biphasic framework, called Push and Pull Search (PPS). The original PPS ignores constraints during the first phase of the search to focus on exploration of the whole search space and approximating the unconstrained Pareto-optimal Front (PF). During the second phase, constraints are considered and previously infeasible solutions are evolved to become feasible in an effort to approximate the constrained PF. Splitting the Constrained Multi-Objective Evolutionary Algorithm (CMOEA) into two phases to meet the challenges of exploration and exploitation, this thesis further explores the use of landscape knowledge to enhance the newly developed PPS framework. The effect on search performance, different problem characteristics behaviour, traversal through constrained space and bias towards certain parts of the search space, introduced by Boundary Search (BS), are analysed and discussed.

Sammendrag

Den siste tiden har forskning fokusert på Beskrankede Flerkriterie-Problemer (BFPer) ettersom beskrankninger byr på krevende utfordringer som må løses. For å overkomme denne utfordringen har nye kunstige testproblemer blitt utviklet for å simulere problemer fra den virkelige verden. I tillegg har Flerkriterie Evolusjonære Algorithmer (EAer) blitt utvidet med Beskrankningshåndtering-metoder (BHMer) for å løse disse problemene. Vanligvis er oppførselen til BHMer kun påvirket av informasjon om selve populasjonen. Det har har vært lite bruk av informasjon innhentet under søket for å påvirke optimeringen av BFPer. Flere kriterier sammen med beskrankninger øker utfordringen med å konvergere mot optimale løsninger i tillegg til å oppnå en høy dekningsgrad av løsninger som oppfyller beskrankningene. Nyere forskning har brakt frem et to-fase rammeverk kalt Push and Pull Search (PPS). PPS ignorerer beskrankninger under den første fasen av søket og fokuserer på å utforske en større del av søkerommet. Under den andre fasen av søket blir beskrankninger tatt med i beregningen og målet er da å finne optimale løsninger som ikke bryter beskrankningene til problemet. Denne avhandlingen utforsker bruken av informasjon innhentet under søket for å videreutvikle det nye rammeverket PPS. Herunder utforskes hvordan Grensesøk (GS) påvirker ytelse, håndtering av ulike problemkarakteristikker, traversering gjennom landskapet, og om hvordan visse deler av søkerommet blir foretrukket fremfor andre.

Preface

The following thesis constitutes the work of a master's thesis in Artificial Intelligence (AI) authored at the Norwegian University of Science and Technology (NTNU), in Trondheim, Norway. It was carried out at the Department of Computer and Information Science (IDI) in the period of 19.08.2019 - 17.07.2020.

We would like to thank our supervisor Pauline Haddow for her excellent guidance, insight and invaluable feedback helping us through the whole process. Also, we would like to extend our gratitude to our peers at the Complex, Robust, Adaptive, Bio-inspired solutions (CRAB) lab for their helpful input during gatherings.

Marius Aarsnes & Erling Wisløff
Trondheim, July 17, 2020

Contents

List of Figures

List of Tables

Glossary

ϵCH ϵ-Constraint-Handling. 16, 26, 35, 36, 43, 44, 48, 79, 94, 106, 107

$\mathbf{FR_c}$ CMOP Feasibility Rate. xiii, 19, 58–60, 63, 72

$\mathbf{FR_p}$ Population Feasibility Ratio. x, 19, 20, 37, 42, 48, 53, 68, 74, 86, 93–96, 99, 104

ACD Active Constraints Detection. xiii, 32, 33, 43–45, 49, 64, 88, 104, 107, 110

ACO Ant Colony Optimisation. 3

AI Artificial Intelligence. iii, 3

BFP Beskranket Flerkriterie-Problem. ii

BHM Beskrankningshåndtering-metode. ii

BiS Binary Search. ix, x, 29, 31, 35, 36, 39, 41, 42, 49, 53, 54, 56–58, 62, 63, 69, 71, 73, 75–80, 82, 83, 89, 90, 92–101, 104–109

BiS-E Push Binary Pull Epsilon. ix, x, xiii, xiv, 36, 75–84, 89, 92–94, 96, 115–118

BiS-IE Push Binary Pull Improved Epsilon. ix–xi, xiii, xiv, 36, 57, 75–84, 89–94, 96–100, 110, 115–118

BS Boundary Search. i, 1, 2, 5, 18, 23, 28–30, 35, 55, 58, 60, 65–67, 70, 73, 89, 96, 105–110

CD Crowding Distance. x, xi, 21, 67, 68, 74, 95, 96, 98–100, 104

CHM Constraint-Handling Method. i, 1, 23–26, 28, 35, 36, 43, 44, 48, 53, 62, 73, 84, 85, 89, 95, 96, 109

CMO Constrained Multi-Objective Optimisation. 5, 23

CMOEA Constrained Multi-Objective Evolutionary Algorithm. i, 4, 5, 23, 24, 26, 39

CMOP Constrained Multi-Objective Problem. i, 1, 2, 4, 14, 16, 19, 23, 24, 28, 29, 42, 48, 59, 61, 105, 108, 109

CRAB Complex, Robust, Adaptive, Bio-inspired solutions. iii

CSOP Constrained Single-Objective Optimisation Problem. 14

DE Differential Evolution. ix, 3, 8, 9, 59

DFR Direct Fitness Replacement. 18

EA Evolutionary Algorithm. 1, 3, 7, 8, 18, 42

EA Flerkriterie Evolusjonær Algoritme. ii

EC Evolutionary Computation. 5

FLA Fitness Landscape Analysis. 5, 18, 28, 105, 109

GS Grensesøk. ii

HV Hypervolume. ix, x, xiii, xiv, 20, 21, 26, 37, 48, 59, 60, 63, 65, 68, 69, 72–87, 89, 91, 93, 98, 105, 116, 117

IϵCH Improved ϵ-Constraint-Handling. ix, 24, 26, 35, 36, 43, 44, 48, 53–55, 59, 61, 62, 79, 91, 92, 94, 106–108

IDI the Department of Computer and Information Science. iii

IFR Indirect Fitness Replacement. 18

IGA Island Genetic Algorithm. 3

IGD Inverted Generational Distance. ix–xi, xiii, xiv, 20, 37, 48, 57, 59, 60, 63, 65, 67–94, 96–105, 115, 116, 118

LRQ Literature Review Question. 5

MOEA Multi-Objective Evolutionary Algorithm. i, 1, 3, 5, 18, 24, 36, 37, 39, 43, 44, 47, 48, 62

MOEA/D Multi-Objective Evolutionary Algorithm based on Decomposition. ix, 12, 13, 37, 39, 43, 44, 48, 52, 59–61, 91

MOO Multi-Objective Optimisation. 3–5, 10, 19

MOP Multi-Objective Optimisation Problem. 5, 10, 11, 13, 14, 19, 23, 62

NTNU the Norwegian University of Science and Technology. iii

OBL Opposition Based Learning. 3, 4

PBPS Push Binary Pull Search. 36, 57, 65–67, 105

PF Pareto-optimal Front. i, ix, xi, xiv, 2, 11, 12, 14–16, 18–21, 23–26, 28–30, 38–41, 43–46, 51–57, 59–63, 65–68, 71–73, 75–77, 79, 80, 82, 83, 85–89, 91–93, 95–108, 118

PGA Parallel Genetic Algorithm. 3

PPS Push and Pull Search. i, ii, ix, 1–4, 6, 23–26, 35–37, 39, 41, 44, 47–49, 51–53, 56, 57, 59, 61, 62, 64–67, 70, 71, 75, 76, 82, 83, 85–87, 89, 94–96, 105, 106, 108, 109

PPS-E Push Pull Epsilon. ix, x, xiii, xiv, 35, 65, 75–77, 80–89, 92–95, 106–108, 115–118

PPS-IE Push Pull Improved Epsilon. ix–xi, xiii, xiv, 35, 57, 75–77, 80–87, 89–101, 103, 106, 108, 110, 115–118

PPS-RSS Push Pull Reduced Search Space. x, xi, xiii, xiv, 36, 65–67, 75, 83–89, 94, 95, 101, 103, 106–108, 115–118

PS Pareto-optimal Set. 11, 12

RQ Research Question. 2, 65–67, 70, 73, 74, 105–107, 109

RSS Reduced Search Space. xiii, 32, 35, 36, 43, 44, 48, 49, 58, 60, 64, 71, 88, 89, 94, 96, 101–110

SCM Supply Chain Management. 3

SD Standard Deviation. 57, 68, 77, 79, 80, 82, 85, 88, 92

SOP Single-Objective Optimisation Problem. 4, 7, 10, 14, 23

Chapter 1

Introduction

This chapter presents the background and motivation for this thesis in section 1.1. Section 1.2 explains the goal and research questions. The research method is elaborated in section 1.3. Finally, the structure of this thesis is outlined in section 1.5.

1.1 Background and Motivation

When designing the layout of a new city, constructing aeroplanes or scheduling trains it is paramount to consider every option carefully before making a decision. This is no simple task as there is seldom one *best* option but rather a set of possible solutions with varying trade-offs. Often, certain restrictions are evident such as time, budget or resources complicating things further. Such problems are called Constrained Multi-Objective Problems (CMOPs) and finding the set of desired solutions to these problems is inherently hard.

An option to solve CMOPs is by applying Evolutionary Algorithms (EAs). EAs are based on *the Darwinian theory of evolution*. These algorithms evolve a population over generations to produce better and better solutions for a problem. EAs lend themselves well to solving problems not solvable by deterministic polynomial algorithms [Back, 1996]. Specifically, one class of EAs called Multi-Objective Evolutionary Algorithms (MOEAs) has proved to be capable of solving CMOPs by introducing some form of Constraint-Handling Method (CHM).

Recently, Fan et al. [2019b] proposed a two-stage framework called Push and Pull Search (PPS) to solve CMOPs. The framework splits the search into two stages. The first phase (*Push stage*) ignores constraints, allowing the population to roam unhindered by infeasible regions. During the second phase (*Pull stage*), constraints are considered. The first stage lends itself to gathering landscape information as the population is unhindered by infeasible regions, enabling it to cover a larger part of the search space. This information may then be used during the second stage to approximate the optimal solutions.

The following work is situated within the field of biologically inspired optimisation and explores the use of landscape information to handle constraints. The proposed approach enhances the promising framework PPS used to solve CMOPs. Landscape information is used to perform Boundary Search (BS) to search around the boundary between feasible and infeasible regions of the problem.

1.2 Goals and Research Questions

This section describes the goal and the Research Questions (RQs) of this thesis. The research goal of this thesis is as follows:

Goal *Investigate how landscape information can increase the performance of PPS when solving CMOPs.*

The population moves freely, unaffected by infeasible regions, during the Push stage. Therefore, this stage naturally lends itself to exploration and information gathering. The information gathered can then be utilised during the Pull stage to approximate the desired solutions. Thus, efforts are focused on investigating which information is useful for solving CMOPs, how this information should be gathered and finally how it should be applied during the Pull stage to benefit PPS. An increase in convergence, coverage and efficiency is sought. If this is not achieved, the goal is to understand why the proposed improvements are not beneficial to the framework.

The following research questions are explored:

RQ1 *What landscape information extracted during the evolutionary search can benefit PPS to increase convergence to and coverage of the constrained Pareto-optimal Front (PF)?*

RQ2 *How do different problem characteristics affect the performance of BS?*

RQ3 *How does the use of BS affect the traversal to the constrained PF through infeasible space?*

RQ4 *How can BS introduce a bias towards certain areas of the objective space?*

1.3 Research Method

The research method applied has been an analytical process. A structured literature review was conducted to identify potentially useful landscape information. Further, methods to incorporate this information into PPS were analysed to identify how the performance of the framework could be improved.

The knowledge accumulated from the literature review was then used to justify the selection of landscape information and how to utilise it. A visualisation and logging tool was developed to monitor the performance of the model. Results from both the literature review and the preliminary testing justified the design decisions of the algorithmic model. An experimental plan was developed and executed to answer the RQs described in section 1.2. The results of the experiments were analysed and discussed. Next, an assessment was performed of the extent to which the research goal was answered. Finally, the contributions of this thesis were elaborated together with future work to further research the use of landscape information in solving CMOPs.

1.4 Research Process

The research process was divided into three distinct phases:

1. The initial literature search to find a topic for the master thesis.
2. Creating a structured literature review protocol to find relevant literature and reduce the scope of the search.
3. Conducting a structured literature review of the literature found and placing this work in context of the state-of-the-art.

These phases are discussed below.

1.4.1 Initial Literature Search

Due to an open project description and little to no restrictions put upon the work, the initial literature search was the process of finding various topics related to biologically inspired Artificial Intelligence (AI). To do so, search engines such as Google Scholar and IEEEXplore were used. Also, recent work of known researchers such as Kalyanmoy Deb and Carlos A. Coello Coello was reviewed. Through an iterative process of discovering new research topics in different sub-fields a research objective emerged. This process is illustrated in figure 1.1 and further elaborated below.

The initial idea was to research the field of Multi-Objective Optimisation (MOO). Inspired by Baug et al. [2019], the search delved into the use of adaptive Parallel Genetic Algorithm (PGA) to solve bin-packing problems. A novel approach using Ant Colony Optimisation (ACO) on top of Island Genetic Algorithm (IGA) was proposed. The idea was to evolve the island topology by strengthening and weakening connections between sub-populations based on the impact of migration. After some research the approach seemed too complex as simpler ones have already produced good results.

The search shifted focus to finding a suitable application area. Norsk Kylling[1] was introduced as a potential collaborator. This gravitated the search towards literature regarding biologically-inspired AI in Supply Chain Management (SCM). Reading the literature evolved a better understanding of Differential Evolution (DE), a simple EA not burdened by extensive parameter-setting which has achieved good results. Exploring applications of DE, a paper by Jevne et al. [2012] inspired the search to focus on DE used to solve portfolio optimisation. It was realised that a specific application area was not as motivating or important. Instead, it was preferred to focus more on techniques for solving MOOs.

The search returned to the field of MOO. The research uncovered that most approaches to MOEAs use a "convergence first, diversity second" approach [Liu et al., 2017]. In addition, two techniques were further investigated: Opposition Based Learning (OBL) [Talukder et al., 2016] and PPS Fan et al. [2019b]. Using OBL together with PPS to alter the balance between convergence and diversity was pursued. Most research on OBL address the enhancement of convergence, therefore,

[1] Norsk Kylling

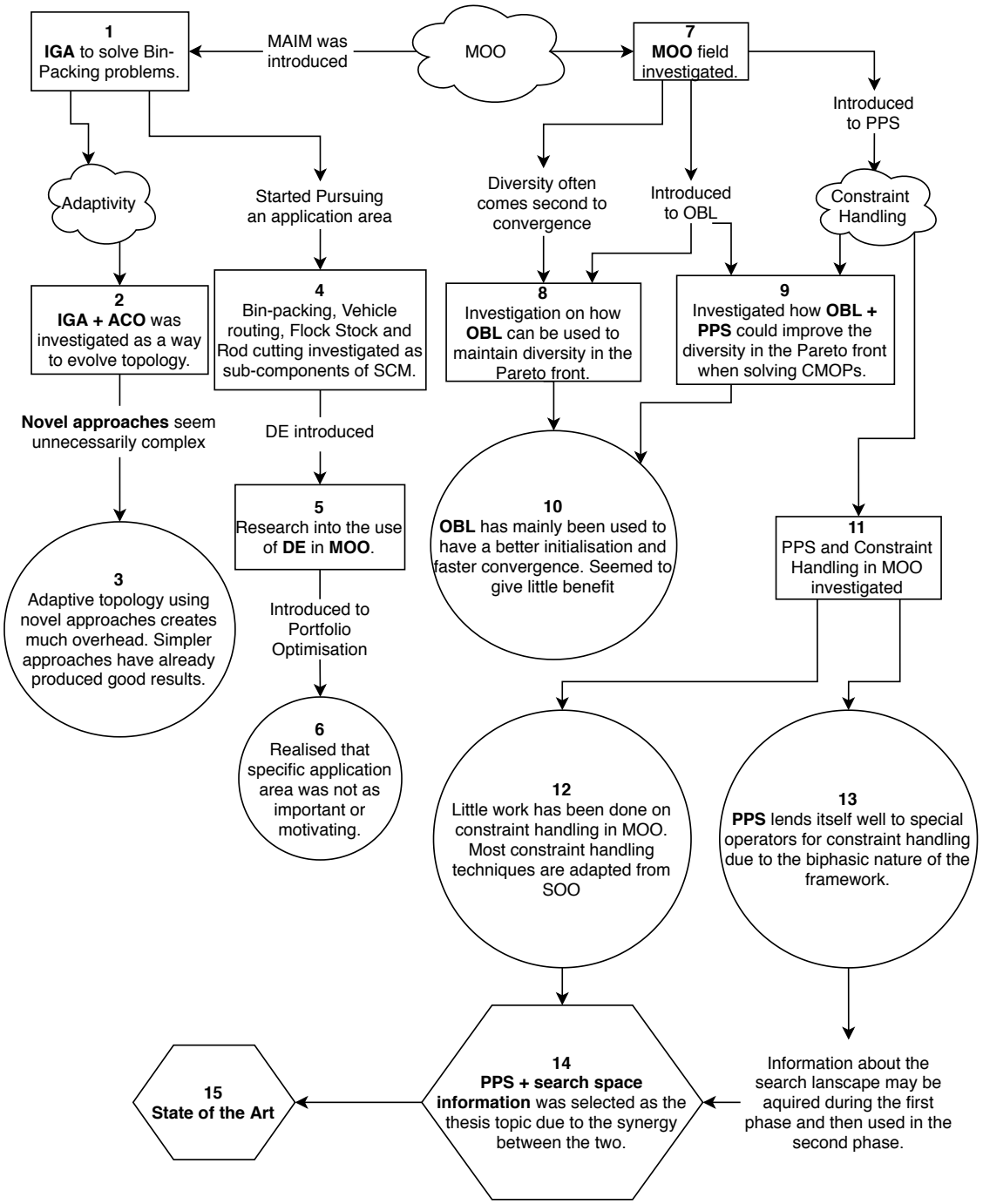


Figure 1.1: Flowchart illustrating the initial literature search.

exploring new ways of utilising the technique could be beneficial. However, OBL was later deemed to not be a good fit for PPS in regards to diversity.

Driven by the interest in PPS, constraint handling in MOO was reviewed. It was found that CMOEAs have received little focus compared to constraint-handling for Single-Objective Optimisation Problems (SOPs) [Del Ser et al., 2019]. Thus, it was decided to define the research objective as *using landscape information to solve CMOPs*.

1.4.2 Structured Literature Review Protocol

The structured literature review protocol was created to find relevant literature for the thesis. Literature Review Questions (LRQs) were defined to control the scope of the review. Due to the open nature of the project the research questions have changed over time. Initially, the focus was on the use of adaptivity in MOO. Over time, this changed to how information gathered during the search could help solving problems and MOO changed to Constrained Multi-Objective Optimisation (CMO).

With the research objective narrowed down, the following LRQs were defined:

- **LRQ1:** What information could be beneficial to gather during the search and how should it be gathered?
- **LRQ2:** What should the gathered information be utilised for and how should it be used?

Search terms were defined to help answer these questions. To find publications, the search engines Google Scholar, IEEEXplore and Web of Knowledge were used. To reduce the amount of literature to be reviewed, some selection criteria were set: *Qualifying-*, *Evaluation-* and *Inclusion Criteria*. These factors have been summarised in table 1.1.

Search Terms	Search space analysis, landscape information, surrogate model, BS, Fitness Landscape Analysis (FLA), MOEA, CMOEA, constraint-handling, two phase, biphasic.
Qualifying Criteria	• Literature should be related to the field of Evolutionary Computation (EC). • Article seems relevant after reading abstract and conclusion. • Should provide results or proof of strengths.
Evaluation Criteria	• Models or algorithms should be reproducible. • The research should be comparable with other approaches or models. • The authors justify their design choices.
Inclusion Criteria	• The work needs to be published by trusted researchers or journals. • The work needs to be dated post the year 2000.

Table 1.1: Search terms and selection criteria for articles used when performing the structured literature review.

1.4.3 Structured Literature Review

The third and final phase of the research process encompassed using the structured literature review protocol to find relevant work. The first step was to map previous efforts to constraint-handling when solving Multi-Objective Optimisation Problems (MOPs). In addition, methods for using information gathering were evaluated. Subsequently, a review of the results was conducted to find approaches that could

lend themselves to PPS. This led to the discovery of *fitness landscape analysis*, *surrogate models* and *boundary search*. The remainder of the literature review was dedicated to further evaluating and researching these methods.

1.5 Thesis Structure

The structure of this thesis is as following: chapter 2 introduces relevant information and concepts required to understand the research. Then, chapter 3 describes the motivating factors for doing the research and the state-of-the-art is highlighted. After that, chapter 4 describes the proposed model, and important design decisions. Finally, chapter 5 contains the experiments and results of the proposed model and chapter 6 discusses the results gathered as well as possible further work and concluding remarks.

Chapter 2

Background Theory

This chapter lays the theoretical foundation needed to understand the contributions of this thesis. Theory on optimisation, constraint handling and EAs is given.

2.1 Optimisation

The purpose of optimisation is to solve a task by finding the optimal solution(s) out of a set of possible solutions. This is achieved by finding the input variables that minimise or maximise, depending on the problem at hand, the output of a function. Without the loss of generality, only minimisation functions are assumed. Normally, the optima of the function, in real-world problems, is not known before-hand and may require extensive search to be found. Formally, SOPs may be defined as in [Coello et al., 2007] as:

Definition 2.1.1. SOP A SOP is defined as minimising the scalar $f(x)$ where x is an n-dimensional decision variable vector $x = (x_1, ..., x_n)$ from some universe Ω. □

$f(x)$ is often called the *objective function* and the output is called the *fitness* of x. The output of the objective function is minimised for all possible solutions x having n *decision variables*. The range of fitness values is known as the *objective space* while the *search space*, Ω, is the range of values for the decision variables.

Figure 2.1 illustrates a simple optimisation problem with only one decision variable x. The global extremum is highlighted by a green circle whilst the local extremum are highlighted by a blue square. This simple example illustrates that there are multiple values for x that give locally optimal solutions, but only one results in the global optimal. Also, notice that going from a local minima to the global is not always easy due to how the landscape between the points may look like.

2.2 Metaheuristics

Stochastic Optimisation is a class of algorithms and techniques that employ some degree of randomness to find solutions to some problem. The most general of these algorithms are called *metaheuristics*. Metaheuristics are applied to problems where the optimal solution is unknown, there is no structured approach to finding it,

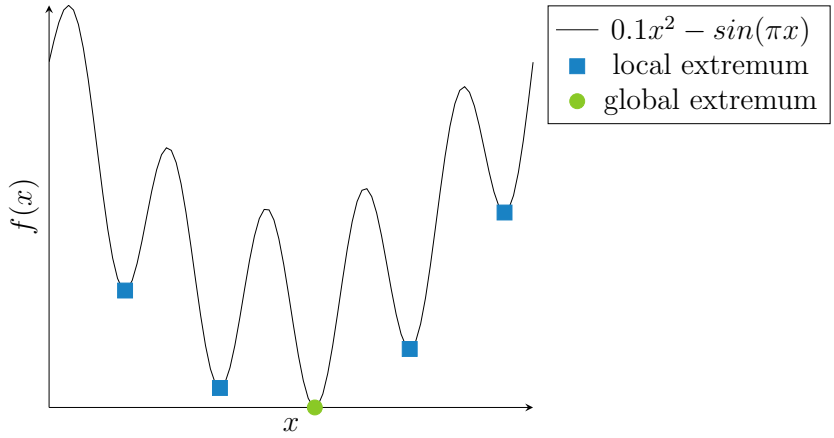


Figure 2.1: Single-objective optimisation example.

there is little heuristic information available and brute force is not a viable option. However, given a candidate solution to the problem, it is possible to test it and assess the quality of the solution.

It is common for metaheuristics to use a combination of *exploration* and *exploitation* to solve a problem. The aim of *exploration* is to find promising regions in the search space, focusing on *global search*. *Exploitation* on the other hand focuses on search near already found promising solutions, focusing on *local search*.

2.3 Evolutionary algorithms

EAs are metaheuristics inspired by biological evolution. These algorithms typically consist of a population of candidate solutions that are mutated and recombined in some way to improve the population, until a sufficient solution is reached.

2.3.1 Differential Evolution

Differential Evolution (DE) is one type of EA. The approach has shown an ability to solve several real world problems [Neri and Tirronen, 2010]. The flow of DE is illustrated in Figure 2.2 and further elaborated in this section.

2.3.1.1 Initialisation

The first step in DE is to **initialise the population**. A common approach to this is simply to initialise each individual at random. If no prior knowledge of the topology of the search space or the viability of each solution, random initialisation encourages *diversity* and the population covers a large portion of the search space. However, using heuristics or knowledge of the problem domain may lead to better starting conditions and therefore also a better convergence. On the other hand, heuristics and domain knowledge may thwart the search by limiting exploration of the whole search space leading to premature convergence.

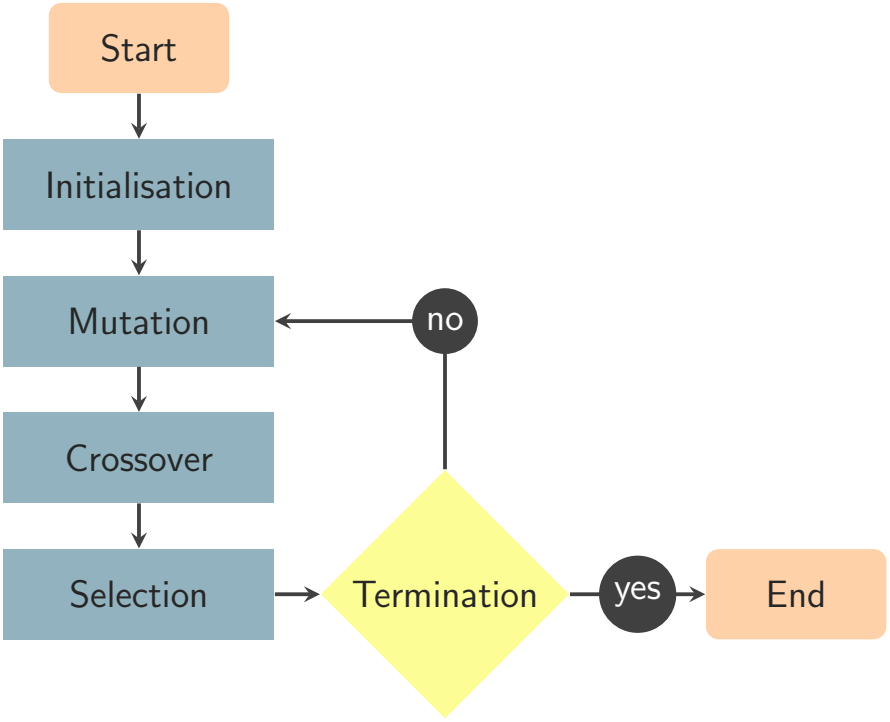


Figure 2.2: Flowchart of DE.

2.3.1.2 Mutation

During each generation, DE generates a mutation vector (*donor vector*) for each individual (*target vector*) in the population. The exact method may vary, but the simplest approach commonly used is described in equation (2.1):

$$d_i = x_{r_1} + \mathcal{F} * (x_{r_2} - x_{r_3}) \tag{2.1}$$

where $i \in 1, ..N$, N is the population size and $i \neq r_1 \neq r_2 \neq r_3$. $\mathcal{F}$ is a scaling factor, usually in the range $(0, 1]$. d_i is the produced donor vector later used during crossover. x_{r_1}, x_{r_2} and x_{r_3} are random distinct individuals from the population.

There are different approaches, either exploiting knowledge regarding the best known individuals or focusing more on random exploration.

2.3.1.3 Crossover

The DE crossover implements a recombination of the created donor vector, d_i and the corresponding target vector p_i to create a *trial vector*. A common approach to crossover is defined below:

$$t_{i,j} = \begin{cases} d_{i,j} & \text{if } rand < CR \text{ or } j == j_{rand} \\ p_{i,j} & \text{otherwise} \end{cases} \tag{2.2}$$
$$t_i = \{t_{i,1}, ..., t_{i,d}\}$$

where t_i is the trial vector created from the target and donor vectors. d is the size of the vectors and $j \in [0, d]$. CR is the crossover rate, which is a predefined

value in the range $[0, 1]$. *rand* is a random number computed for each j in the range $(0, 1)$. Finally, j_{rand} is a random integer computed once for each individual in the range $[0, d]$.

In the case that a value $t_{i,j}$ violates the boundary constraint put upon the value, it is reset using some form of repair function.

2.3.1.4 Selection

Selection is the process of deciding which individual of the target vector in the population and the newly created trial vector should survive to the next generation. Some sort of fitness function or ranking operator is used to describe how good each individual is with respect to the given problem and each other. For SOPs, some numerical value that can be directly comparable to another is sufficient to distinguish the quality of the individuals. For MOPs more advanced evaluation and sorting techniques are needed to compare individuals, like Pareto Dominance (definition 2.4.2).

2.3.1.5 Termination

At the end of each generation the algorithm checks for a certain **stop condition being met**. If this is the case, the termination of the algorithm happens. If not, the algorithm continues the evolution. The condition may be any single or combination of the following:

- A certain number of generations has been iterated.
- A certain number of computations has been performed.
- A certain fitness or evaluation score has been met.

2.4 Multi-Objective Optimisation

In addition to SOPs, there exist problems which have more than one objective to optimise. These problems are subject to MOO. A simplified example of this is minimising cost while maximising quality when buying a product. Most real-world optimisation problems are more comparable to MOPs than SOPs. Formally, MOP may be defined as [Coello et al., 2007]:

Definition 2.4.1. Multi-Objective Optimisation Problem

A MOP is defined as minimising $f(u) = [f_1(u), f_2(u), ..., f_k(u)]$ subject to $u \in \Omega$. $F(u)$ is a vector with k objectives where $u = [u_1, u_2, ..., u_n]$ is an n-dimensional decision variable vector in some universe Ω. □

In MOPs, the concept of "optimum" changes due to having multiple objective functions. This is because the aim is not to find a single global optimum, but rather to find good "trade-offs" between the different objective functions. The most commonly adopted notion of "optimum" is *Pareto optimum*.

Below, a set of Pareto definitions are presented [Coello et al., 2007]. The purpose of these definitions is to simplify explanation in later parts of this thesis:

Definition 2.4.2. Pareto Dominance

A vector $u = (u_1, ..., u_k)$ is said to dominate another vector $v = (v_1, ..., v_k)$ if and only if $\forall i \in \{1, ..., k\},\ u_i \leq v_i \wedge \exists i \in \{1, ..., k\}\ u_i < v_i$. This notion is denoted $u \preceq v$. □

Definition 2.4.3. Pareto Optimality

A solution vector $u \in \Omega$ is Pareto Optimal with respect to Ω if and only if there is no solution vector $v \in \Omega$ for which $v \preceq u$. □

Definition 2.4.4. Pareto-optimal Set (PS)

For a given MOP, the PS is defined as:

$$\mathcal{PS} = \{u \in \Omega \mid \neg \exists v \in \Omega\ v \preceq u\} \tag{2.3}$$

The PS is the set of solutions that are not dominated by any other solution in the search space. □

Definition 2.4.5. PF

For a given MOP and PS, the PF is defined as:

$$\mathcal{PF} = \{f(u) \mid u \in \mathcal{PS}\} \tag{2.4}$$

The PF is the PS plotted in the *objective space*. □

Figure 2.3 illustrates many of the concepts defined above. The problem is a minimisation problem of the two objectives f_1 and f_2. The circles are points belonging to the true PF as per definition 2.4.4. The other sets of points, denoted by different shapes, are sets of *non-dominating* points with different *ranks* or belonging to different non-dominating sets. The rank of a point is dependent of how many non-dominating sets which dominate it. All the squares dominate the triangles and diamonds, while all the triangles dominate the diamonds.

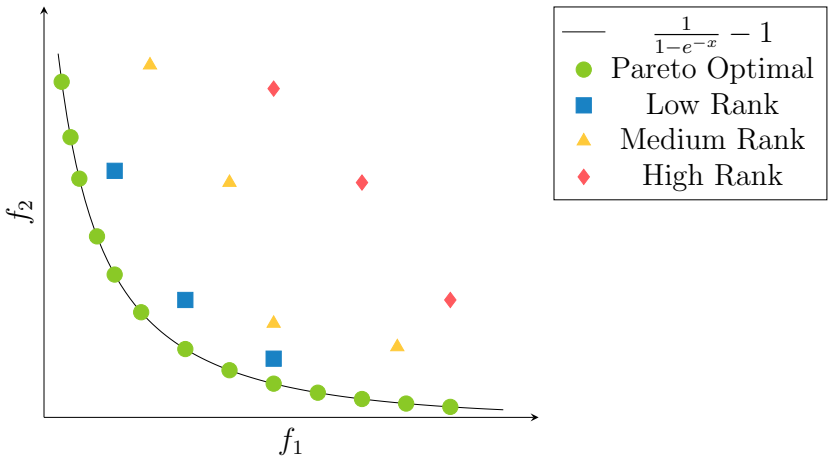


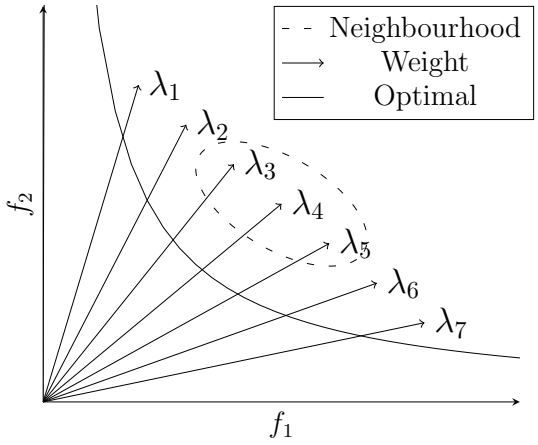
Figure 2.3: Multi-objective optimisation example.

In general, it is not a simple task to find an analytical expression of the line or surface which is the PF. One approach is to compute multiple points in Ω and their

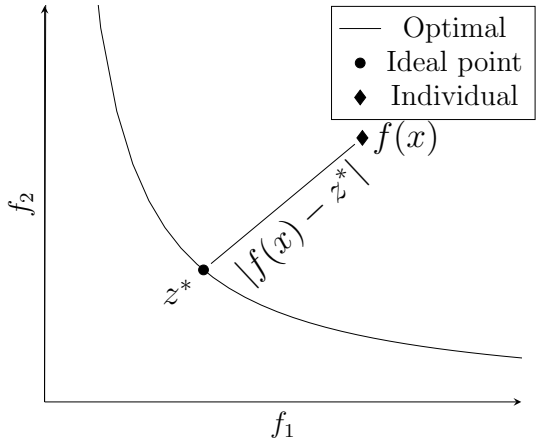
corresponding $f(\Omega)$. When a sufficient number of points have been computed it is possible to determine the PS and then determine the PF. This is done by finding more and more solutions which either dominate or belong to the *current* set of non-dominating solutions of the lowest rank. Note that the best known solutions are not necessarily a part of the actual PF, this is however the end goal.

At a certain point, the search is ended. This may be the result of the solutions being of a certain calibre, a predefined time has passed, or the improvement in points found is halted. Note that at the end of the search, a specific solution has not been selected. The result of the search is a set of options to choose from, which are believed to be the best alternatives. The task at hand is thus to select one of the solutions. This is done through a decision maker, which may either be a person or some automated process.

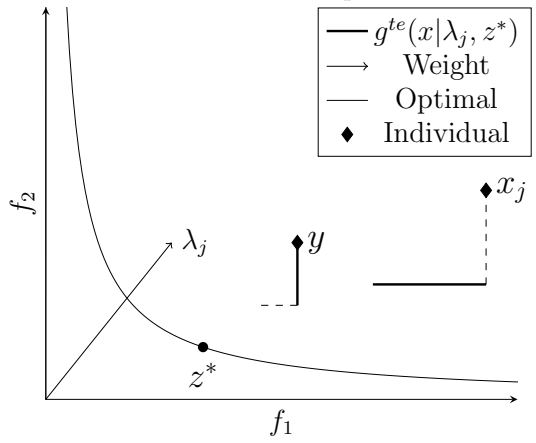
2.4.1 Multi-Objective Evolutionary Algorithm based on Decomposition



(a) Aggregation coefficient vectors.



(b) Distance between the individual x_i and an ideal point z^*.



(c) The calculated Tchebycheff values for x_i and y against the subproblem defined by λ_j and z^*.

Figure 2.4: Different aspects of MOEA/D.

Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) decomposes a MOP into N scalar optimisation subproblems. A *scalar aggregation function* defines these subproblems. One way of defining such a subproblem is to use the *Tchebycheff approach*, as defined in equation (2.5).

$$\begin{aligned} &\text{minimise } g^{te}(x|\lambda, z^*) = \max_{1\leq i\leq m}\{\lambda_i|f_i(x) - z_i^*|\} \\ &\text{subject to } x \in \Omega \end{aligned} \tag{2.5}$$

where $z^* = (z_1^*, ..., z_m^*)^{\mathbb{T}}$ is the reference point, i.e. $z_i^* = \min\{f_i(x)|x \in \Omega\}$ for each $i = 1, \ldots, m$. The reference point, z^*, is updated with every newly generated offspring. This allows the point to move further towards origin as the individuals within the population improve. λ_i is aggregation coefficient vector defining the subproblems weight towards an area of the search space. Each subproblem is solved simultaneously by evolving a population of solutions where primarily information from the neighbourhood of subproblems is used. The neighbourhood relations among the subproblems are defined based on the distances between their aggregation coefficient vectors as visualised in figure 2.4a. A uniformly distributed vector set allows a uniform distribution among the subproblems.

The fundamental mechanic of the Tchebycheff approach is the prioritisation of individuals closest to z^* as visualised in figure 2.4b. The goal of the algorithm is to find one individual that performs superior to all others within a subproblem. An essential key to the algorithm is, therefore, an equal amount of individuals as there are subproblems. The work by Zhang and Li [2007] generates the aggregation coefficient vector from a defined set $\left\{\frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H}\right\}$. H thus needs to be defined and is the controlling parameter for the population count. One generation of the algorithm involves iterating over each subproblem, producing an offspring from randomly selected individuals within the neighbourhood, defined as y. The offspring is then compared against all individuals within the neighbourhood, taking the place of inferior individuals as defined in equation (2.6), where j is the index of each vector in the neighbourhood.

$$x_j = \begin{cases} y & \text{if } g^{te}(y|\lambda_j, z^*) \leq g^{te}(x_j|\lambda_j, z^*) \\ x_j & \text{otherwise} \end{cases} \tag{2.6}$$

Figure 2.4c visualises the comparison between y against x_j within the subproblem defined by λ_j. y has a lower, and therefore better, score visualised by the shorter solid line expanding from the nodes in the figure. In this example, y takes the place of x_j.

2.5 Constrained Optimisation

In most real-world optimisation problems there may exist restrictions either due to characteristics of the environment or available resources. To achieve acceptable solutions, these restrictions need to be satisfied. In general, these restrictions are called *constraints*. Usually, these constraints are expressed through mathematical inequalities or equalities as described in equation (2.7).

$$\begin{aligned} g_i(x) &\leq 0 \text{ for } i = 1, ..., m \\ h_j(x) &= 0 \text{ for } j = 1, ..., p \end{aligned} \tag{2.7}$$

where x are decision variables, m is the number of inequality constraints and p is the number of equality constraints. Greater-than-or-equal-to inequality constraint ($g_j(x) \geq 0$) may be accommodated by multiplying the left side by -1.

Both definitions of SOPs and MOPs can be expanded with these constraints turning them into Constrained Single-Objective Optimisation Problems (CSOPs) and CMOPs respectively.

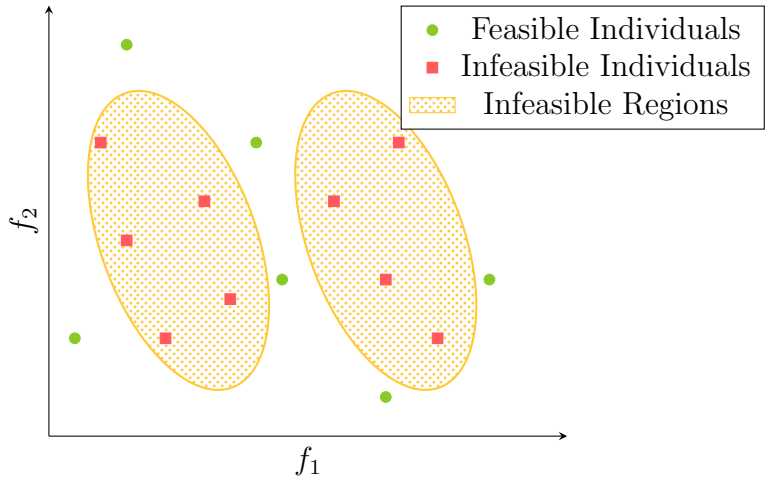


Figure 2.5: Search space with feasible and infeasible individuals.

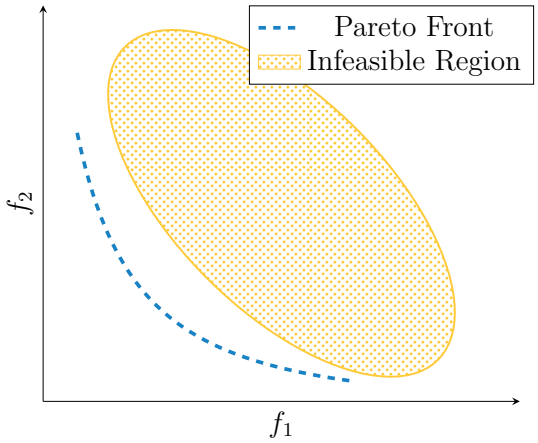
Constraints of a problem define infeasible regions in the decision space. This is illustrated in figure 2.5. Solutions that violate constraints are known as *infeasible individuals* and are located inside these regions - the squares. Solutions that do not violate constraints are called *feasible individuals* - the circles.

Depending on where these regions are located in the objective space, they affect the search for the PF in different ways. These can be grouped into three different categories:

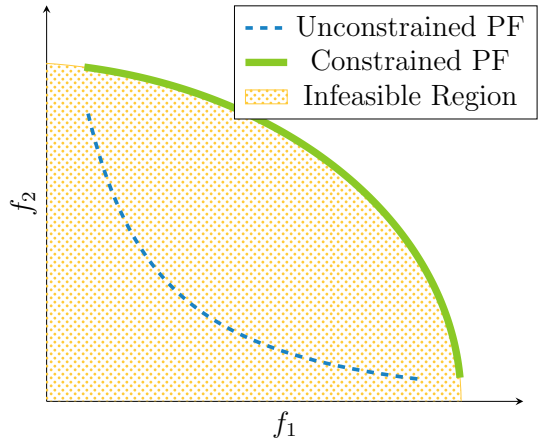
- Infeasible regions block the way to the PF.
- Infeasible regions cover the whole PF.
- Infeasible regions cover parts of the PF.

Figure 2.6a illustrates the first category where infeasible regions cover a large area of the search space. In these cases, the PF are not directly affected or altered by the constraints. With infeasible regions blocking the way to the PF, the challenge is to get passed these regions.

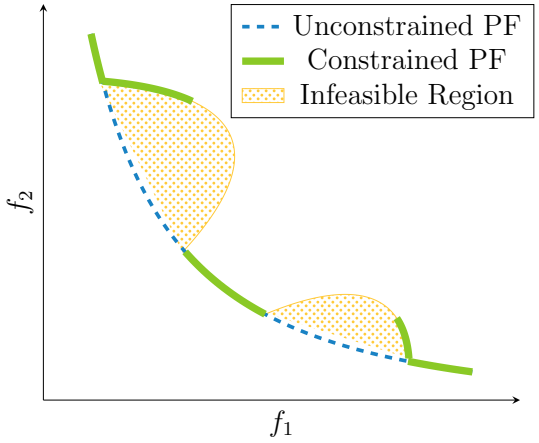
Figure 2.6b illustrates the instance where the whole PF is covered by infeasible regions. In this case the notion of PF is divided into two types: unconstrained PF



(a) Infeasible region blocking the way to the PF.



(b) Infeasible region covering the whole PF.



(c) Infeasible regions covering parts of the PF.

Figure 2.6: Different ways infeasible regions may affect the PF.

illustrated by the blue thin line and constrained PF illustrated by the thick green line. The goal is to approximate the constrained PF.

Figure 2.6c illustrates the third instance where the PF is partially covered by infeasible regions. Here the constrained PF is made up of parts from the unconstrained PF and the boundary between feasible and infeasible space, created by infeasible regions.

In figures 2.6b and 2.6c the constrained PF is located on the border between feasible and infeasible space. The constraints creating the infeasible space which separate the unconstrained from the constrained PF are called *active constraints*. The definitions of what an active constraint is varies. In this work definition 2.5.1 is used:

Definition 2.5.1. Active Constraint

A constraint is said to be active if the boundary of the infeasible region it creates in the objective space intersects the constrained PF. In the case that a problem has any active constraints, an individual that lies on the constrained PF will have a constraint value of 0 for one or multiple active constraints. □

Conversely, a inactive constraint is a constraint which does not intersect with the

constrained PF. Inactive constraints thus have no direct effect on the constrained PF, however, they may still affect the search as they also produce infeasible regions in the objective space.

2.5.1 Constraint-Handling

When solving CMOPs, some form of constraint-handling is required. The purpose of constraint-handling techniques is to guide the search towards the constrained PF opposed to the unconstrained PF.

One such approach is to use *penalty functions*. Using this approach, the constrained optimisation problem is turned into an unconstrained one. This is achieved by expanding the objective function to be optimised as defined in equation (2.8):

$$\begin{aligned} fitness(x) &= f(x) + \phi(x) \\ \text{where } &\begin{cases} \phi(x) = 0, if\ x \in \mathcal{F} \\ \phi(x) > 0, if\ x \notin \mathcal{F} \end{cases} \end{aligned} \tag{2.8}$$

fitness(x) is the expanded objective function. $f(x)$ is the original objective function and $\phi(x)$ represents a penalty for an infeasible individual x, or the cost for making it feasible. The penalty is 0 if x is in feasible space $\mathcal{F}$, and greater than 0 if not. The penalty should be kept as low as possible, but still prohibit infeasible solutions to be seen as optimal. This notion is conceptually simple, however, in practice it is difficult to implement this rule. The difficulty is due to that the exact location of the boundary between the feasible and infeasible regions is unknown in most problems.

Takahama and Sakai [2005] proposed a constraint-handling method called ϵ-Constraint-Handling (ϵCH). The method builds on the idea of penalty and similarly computes a constraint violation as follows:

$$\phi(x) = \sum_i max(0, g_i(x))^p + \sum_j (h_j(x))^p \tag{2.9}$$

where $\phi(x)$ is the sum of all constraint violations defined in equation (2.7) and p is a positive number. The difference here is that the fitness is not expanded by ϕ. Instead, the two following concepts are used:

The two main components of the method are:

- A relaxation of the limit to consider a solution feasible, based on its constraint violation.
- A lexicographical ordering mechanism where the minimisation of the constraint violation precedes the minimisation of the objective function called the ϵ *level comparison*.

Assuming f_i and ϕ_i are the function values and the constraint violation of an individual i respectively. Then, for any $\epsilon \geq 0$, ϵ level comparisons $<_\epsilon$ and $\leq_\epsilon$ between two points are defined as follows:

$$(f_1, \phi_1) <_\epsilon (f_2, \phi_2) \iff \begin{cases} f_1 < f_2, & \text{if } \phi_1, \phi_2 \leq \epsilon \\ f_1 < f_2, & \text{if } \phi_1 = \phi_2 \\ \phi_1 < \phi_2, & \text{otherwise} \end{cases} \tag{2.10a}$$

$$(f_1, \phi_1) \leq_\epsilon (f_2, \phi_2) \iff \begin{cases} f_1 \leq f_2, & \text{if } \phi_1, \phi_2 \leq \epsilon \\ f_1 \leq f_2, & \text{if } \phi_1 = \phi_2 \\ \phi_1 < \phi_2, & \text{otherwise} \end{cases} \tag{2.10b}$$

An optimisation problem solved using the ϵ-constraint handling method is defined as follows:

$$\begin{aligned} (P_{\leq_\epsilon}) \quad & \text{minimise}_{\leq_\epsilon} \quad f(x) \\ & \text{subject to} \quad \phi(x) \leq \epsilon \end{aligned} \tag{2.11}$$

where minimise$\leq_\epsilon$ denotes the minimisation based on the ϵ level comparison $\leq_\epsilon$.

During the evolution, the ϵ level is controlled using equation (2.12).

$$\begin{aligned} \epsilon(0) &= \phi(x_\theta) \\ \epsilon(t) &= \begin{cases} \epsilon(0) \times (1 - \frac{t}{T_c}^{cp}), & \text{if } 0 < t < T_c \\ 0, & t \geq T_c \end{cases} \end{aligned} \tag{2.12}$$

where $\epsilon(0)$ is the initial ϵ level set to the constraint violation of some individual x_θ. The selection of x_θ can either be random or based on some specific criteria, for instance the individual with the largest constraint violation. T_c is some control generation where the updating of the ϵ level is halted and set to 0. Finally, cp is a parameter used to control the speed of reducing the relaxation of constraints. Controlling the ϵ level forces the population further and further out of the infeasible regions of the search space.

2.6 Informed Search

Informed search is the notion of using some form of information to guide the search in a desired direction. This can for instance be to explore specific regions of the search space believed to contain good solutions, or it can be to force the exploration of new regions.

The specific information used, and how it is used may vary. Some approaches are introduced below.

2.6.1 Surrogate Models

For many problems, the objective functions being minimised may prove challenging. This can either be in the form of heavy computations or simply the produced objective landscape being difficult to traverse. A surrogate model is an approximation of the objective function used to construct simpler and lower computational cost models, and is defined in equation (2.13).

$$f'(x) = f(x) + e(x) \tag{2.13}$$

Where $f(x)$ is the original objective function and $e(x)$ is an approximated error. Using individuals and their respective fitness evaluation, these models can generate simpler representations that capture relations between the search space and objective space, and not the underlying process. A benefit with this approach is that there is no need to understand the internal behaviour of $f(x)$, only the input/output behaviour is required.

Shi and Rasheed [2010] grouped the use of surrogate models in EAs into two types: Direct Fitness Replacement (DFR) and Indirect Fitness Replacement (IFR). DFR is an approach that uses the surrogate model to asses the solutions directly in the evolutionary process. This approach assumes that the achieved fitness of the surrogate model are comparable to the fitness achieved by the original objective function. IFR approaches do not use the surrogate model directly in the evolutionary process. Instead, the original objective function is used by the MOEA for a coarse grained search. The surrogate model is used for exploitation in the form of local search.

2.6.2 Fitness Landscape Analysis

A fitness landscape can be described by equation (2.14).

$$FL = (f, d) \tag{2.14}$$

where the fitness landscape, FL, is described by f defining the objective value and d defining the distances between individuals in the search space. This makes the fitness landscape not only dependent upon the problem, but also the selected representation and how operators are used for recombination of individuals.

The purpose of FLA is to gain a better understanding of how a heuristic algorithm performs and progresses through the landscape of the objective space. One approach is to perform a walk through the landscape. This is done by moving, or evolving, individuals while keeping an eye on the development of the fitness. Using this technique, fitness landscapes can be explored with a bias towards the fitter regions by forfeiting details in lower quality and thus less explored regions.

2.6.3 Boundary Search

As discussed in section 2.5, constraints can affect the PF in various ways. Often, this leads to the PF being located on the border between feasible and one or more infeasible regions. BS is based on this assumption and uses special operators to approximate the border between infeasible and feasible search space. One interesting characteristic of BS is the reduction of the search space. The reduction is due to exploration focusing on the part of the search space where feasible and infeasible space is bordering. Several methods for BS have been proposed

Metkar and Kulkarni [2014], viewed every constraint as an extra objective. In this approach, the closer an individual is to a constraint border the less the fitness

of the individual is impacted. One downside with this approach is that inactive constraints may impact the fitness of individuals in a negative manner.

2.7 Benchmarks

To evaluate optimisation algorithms, benchmarks are often used. Benchmarks allow for a controlled setting to find both strengths and weaknesses of an algorithm. With benchmarks, the optimal solution(s) are known beforehand and the functions themselves are often computationally inexpensive. However, benchmarks have difficulties capturing the challenging nature of real-world problems. For MOO, the *true PF* for the test problem is visualised together with the PF found by the algorithm. There are some typical problem characteristics which benchmarks try to emulate. These characteristics include unimodal vs. multimodal, convex vs. concave, and differentiable vs. non-differentiable.

Test suites are a set of test functions which include a mix of different characteristics. By applying an algorithm to the complete test suite one can see which types of problems an algorithm handles well and which types of problems are challenging. Test suites ensure that testing is performed on an assortment of test functions and not only a certain type of problems.

2.8 Multi-Objective Performance Metrics

MOPs produce a set of solutions. Different performance metrics are used to evaluate the quality of these solutions. Usually, these metrics give a numerical value which represents the quality of a given solution or set of solutions. This section describes a few common performance metrics that will be used to evaluate the results of the experiments in this thesis.

2.8.1 Feasibility Rate

The CMOP Feasibility Rate ($\mathrm{FR_c}$) describes the ability of an algorithm to obtain feasible solutions to a CMOP as described by equation (2.15).

$$fr_c = \frac{Q}{n} \tag{2.15}$$

Where Q is the number of runs where the algorithm discovered at least one feasible solution and n is the total number of runs. The value is in the range $[0, 1]$ where 1 is the desired value and represents the reliability of the algorithm to find feasible solutions.

2.8.2 Feasibility Ratio

The Population Feasibility Ratio ($\mathrm{FR_p}$) is the ratio between feasible and infeasible individuals within the population. The Population Feasibility Ratio ($\mathrm{FR_p}$) can is defined by the equation (2.16)

$$fr_p = \frac{F}{n} \tag{2.16}$$

Where F is the number of feasible individuals and n is the population size. If the Population Feasibility Ratio is 1, all of the individuals is feasible.

2.8.3 Inverted Generational Distance

Inverted Generational Distance (IGD) is a metric which measures the distance between the known PF and the true PF by looking at the distance of each solution in PF_{true} to the closest point in PF_{known}. The performance metric not only measures the convergence of the known PF, but also the coverage of the true PF. Due to the point of view being from the true PF, a population which has converged to a small part of the true PF will give a poor score for IGD. Equation (2.17) illustrates how IGD is computed:

$$\begin{aligned} IGD(\mathcal{PF}_{known}, \mathcal{PF}_{true}) &= \frac{calculateDistance(\mathcal{PF}_{known}, \mathcal{PF}_{true})}{|\mathcal{PF}_{true}|} \\ calculateDistance(\mathcal{PF}_{known}, \mathcal{PF}_{true}) &= \sum_{p \in \mathcal{PF}_{true}} d(p, \mathcal{PF}_{known}) \end{aligned} \tag{2.17}$$

Where PF_{known} is the PF found by the algorithm and PF_{true} is the PF of the problem. *calculateDistance* is the method used to calculate the distance. In the case of equation (2.17), a simple summation over the distance is used. d is some predefined function to calculate the distance between a point, p, in PF_{true} and the PF_{known}. The closer IGD is to 0, the better. Obtaining a measure of 0 is optimal, and indicates that the known PF is evenly spread out over the true PF.

2.8.4 Hypervolume

Zitzler and Thiele [1998] was the first to propose Hypervolume (HV) as a performance measure. They identified it as the *size of the space covered* or *size of dominated space*. The HV is obtained using equation (2.18).

$$HV(\mathcal{PF}_{known}) = \mathcal{L}\left(\bigcup_{p \in \mathcal{PF}_{known}} [f_1(p), z_1^r] \times ... \times [f_m(p), z_m^r] \right) \tag{2.18}$$

where $z^r = (z_1^r, ...z_m^r)^T$ is a reference point used to bind the objective space. m is the number of objectives. The measure prefers convex to non-convex regions. In other words, the measure is used for maximisation problems, but works on minimisation problems if the PF is inverted prior to calculating the HV. $\mathcal{L}$ is called the Lebesque measure.

Some calculations of HV in this thesis will be performed on problems with characteristics similar to figures 2.6b and 2.6c where the reference point is defined as the nadir point of the constrained PF. A population thus has the possibility of being located on the unconstrained PF. As the unconstrained PF might be located further

from the reference point than the constrained PF, a population located closer to the unconstrained PF will perform better. In these cases the HV metric will be misleading with regards to the true PF and a infeasible population will seem superior to a feasible population.

2.8.5 Crowding Distance

The Crowding Distance (CD) defines the space in between an individual and it's neighbouring individuals. The concept assumes that the individuals in the neighbourhood are not dominated by the individual. The metric is meant to give an indication of diversity or coverage, where an individual with a larger CD typically can be seen as more *diverse*. The metric is defined in equation (2.19)

$$CD(j) = \sum_{i=1}^{M} \frac{d_i^j}{f_i^{max} - f_i^{min}} \tag{2.19}$$

Where $d_i^j = |f_i^{j+1} - f_i^{j-1}|$ is the distance between the ith objectives of individual the neighbouring individuals $j+1$ and $j-1$. f_i^{max} is the highest objective value achieved within the population, correspondingly f_i^{min} is the lowest objective value. The outermost individuals in the population has their CD set to ∞ as these individuals only has a single neighbour in a given objective dimension.

When CD is referred to in the context of a population, the sum of crowding distances of all individuals in the population is implied. When calculating the sum, the outermost individuals are removed as ∞ cannot be summed.

Chapter 3

Motivation and State of the Art

This chapter presents the state of the art and the motivation for this thesis. First, the recent advances in CMOPs are presented in section 3.1 with emergent challenges. Then the PPS framework used for CMO is elaborated in section 3.2. Finally the fields of informed search and BS in sections 3.3 and 3.4 are presented as possible extensions to the PPS framework.

3.1 Constrained Multi-Objective Problems

Algorithms have been essential to solving emerging problems in the modern world. Where humans are incapable of solving these problems directly, algorithms have been developed as a tool to help solve the issues at hand. As the modern world evolves and becomes more complex, so do the problems that need to be solved [Tanweer et al., 2016; Bonyadi et al., 2013]. From SOPs to MOPs, the complexity increases as conflicting criteria must be taken into consideration. Similarly CMOPs increase the complexity further by adding constraints. CMOPs can be seen in the real world infrastructure and applications as water distribution networks [Monsef et al., 2019], scheduling and vehicle routing problems [Paraskevopoulos et al., 2017].

3.1.1 Artificial Problems

To help develop algorithms suited to solve CMOPs, benchmarks are created. In recent years, real-world problems have been studied and benchmarks currently used have been criticised for not being lifelike enough and not consisting of the same characteristics as real-world problems [Ma and Wang, 2019].

Several benchmarks for testing CMOEAs have been solved without the use of CHM [Tanabe and Oyama, 2017]. The reason is that the constrained PF being unchanged from the unconstrained PF. This is the case for problems where infeasible regions block the way to the PF, described in section 2.5. Thus, approximating the unconstrained PF will produce similar results as approximating the constrained PF [Tanabe and Oyama, 2017].

New constraint composition methods have been created to produce new test suites. The aim of these test suites is to better analyse the performance of CMOEAs using benchmarks that better mimic real-world problems. Li et al. [2016] proposed

a set of new benchmarks where the constrained and unconstrained PF were separated. Similarly, Fan et al. [2019a] and Ma and Wang [2019] created new constraint composition methods to create their own test suites which they called LIR and MW respectively. These test suites are also dominated by problems where the constrained and unconstrained PF are either completely separated or only partially overlapping. Having the two fronts partially or completely separated enhances their capability to accurately evaluate the performance of a CMOEA, as a MOEA without any CHM will have problems producing similar results [Tanabe and Oyama, 2017].

Another aspect of real-world problems which has been neglected by many test suites is that problems often have small feasible regions and a low complexity of the boundary of feasible and infeasible regions [Ma and Wang, 2019]. Both LIR and MW feature problems with small feasible regions reducing the likelihood of MOEAs with no CHM identifying feasible individuals. LIR allows for setting the PF as either convex of concave. Also, it has the ability to make convergence more difficult by applying scaling factors to the objective functions. MW allows for more granulated control and uses constraints to affect that shape and size of the PF to a much larger extent than Fan et al. [2019a].

A downside to the test suite created by Li et al. [2016] is that the problems are nearly identical. The size of the feasible space and the continuity of the PF are the only differences. In contrast, LIR and MW contain different problems. This is in regards of where the feasible and infeasible regions lie in the objective space, the number of objectives and constraints, and the difficulty of the objective functions. Thus, the test suites are more suitable to evaluate the generality of a CMOEA due to their varying characteristics.

Benchmarks problems do not provide sufficiently complex problems. Hence, they are not applicable as replacements for their real-world counterparts and it is therefore difficult to accurately evaluate a CMOEA when all results are good. MW show a much more complex structure as opposed to LIR. Due to them being more complex, MW are better suited to test the CMOEA as a whole. However, LIR is well suited to analyse the performance of CHMs as the main focus of this test suite is the constraints and the difficulty of handling them.

3.2 The Push and Pull Search Framework

After creating the LIR test suite, Fan et al. [2019b] created a new biphasic CMOEA called PPS. The framework performed well on LIR when compared to other state-of-the-art algorithms. However, it has not been tested on other CMOPs.

The framework splits the search into two different stages in an effort to balance objective minimisation and constraint handling. During the first stage, the population is pushed toward the unconstrained PF without considering any constraints. In the pull stage, the population is pulled towards the constrained PF using an Improved ϵ-Constraint-Handling (IϵCH) approach.

Results from the experiments highlight two key features of the framework [Fan et al., 2019b]: It has the ability to get across large infeasible regions and it facilitates information gathering. Due to constraints being ignored during the push phase, the population can move unhindered by infeasible regions. Also, not being restricted

by infeasible regions allows the population to traverse a larger part of the objective space. Thus, information about the landscape may be collected and then used.

As mentioned in section 2.5, infeasible regions may interact with the PF in three distinct ways [Fan et al., 2019b]:

1. Infeasible regions are blocking the way towards the PF, as illustrated in figure 2.6a. In this case, the unconstrained PF and the constrained PF identical. The infeasible regions have no effect during the push phase of PPS due to constraints being ignored. The true PF is approximated during the first stage due to the constrained and unconstrained PFs being the same. Therefore, the pull stage will have little to no effect on the working population.

2. Infeasible regions cover the entire unconstrained PF, as illustrated in figure 2.6b. Due to the unconstrained PF being covered by infeasible regions, the constrained PF lies on some constraint boundary. In this case, PPS will approach the unconstrained PF during the push phase, passing the constrained PF. When the pull stage is commenced, the working population is pulled away from the unconstrained PF and towards the true (constrained) PF.

3. Infeasible regions cover parts of the unconstrained PF, as illustrated in figure 2.6c. In this case, PPS will approximate some of the true PF during the push stage. Later, when the pull stage is commenced, the rest of the true PF will be approximated.

A critical part of the framework is the mechanism to switch between phases. It is beneficial for the framework to stay in the push phase for a sufficient amount of time to either reach or pass the constrained PF. This is to allow the CHM to start the pull phase with an optimal population, effectively guiding the population from optimal *infeasible* solution, to optimal *feasible* solutions. By the original work proposed, a detection mechanism taking all the three previously mentioned situations into account is suggested:

$$r_k = \max\{rz_k, rn_k\} \leq \epsilon \tag{3.1}$$

$$rz_k = \max_{i=1,\dots,m} \left\{ \frac{|z_i^k - z_i^{k-l}|}{\max\{|z_i^{k-l}|, \Delta\}} \right\} \tag{3.2}$$

$$rn_k = \max_{i=1,\dots,m} \left\{ \frac{|n_i^k - n_i^{k-l}|}{\max\{|n_i^{k-l}|, \Delta\}} \right\} \tag{3.3}$$

rz_k represents the largest change of the ideal points for the last l generations, at generation k. rn_k represents the largest change of the nadir points for the last l generations, at generation k. Δ is a small non-negative number to avoid division by zero. r_k represents the largest change in either ideal or nadir points. If r_k is below a certain threshold, ϵ, then the population is assumed to have approximated the unconstrained PF, and thus stagnated. Following this, the frameworks changes to the pull stage to approximate the constrained PF.

Fan et al. [2019b] tested PPS with two different CHMs: ϵCH and IϵCH. IϵCH is an extension of ϵCH. While ϵCH simply reduces the constraint relaxation during the whole run as described by equation (2.12), IϵCH allows for the constraint relaxation to be increased again, as shown in equation (3.4).

$$\epsilon(k) = \begin{cases} (1-\tau) \times \epsilon(k-1), & \text{if } rf_k < \alpha \\ \epsilon(0) \times (1 - \frac{k}{T_c})^{cp}, & \text{if } rf_k \geq \alpha \end{cases} \tag{3.4}$$

where $\epsilon(k)$ is the ϵ level at generation k and $\epsilon(0)$ is set to the max constraint violation of the run at the point of initialisation of the CHM. rf_k is the ratio of feasible to infeasible individuals in the population and α is a value in the range $[0, 1]$. τ is used to control the speed of reducing the constraint relaxation when $rf_k < \alpha$. cp is similarly used when $rf_k \geq \alpha$. Just like in ϵCH, IϵCH updates the ϵ level until the current generation k reaches T_c.

During all phases, an archive NS is kept to store non-dominated and feasible solutions. The archive is updated as described in NSGA-II Deb et al. [2002], calculated from the union of the existing NS and the population. The union set is categorised into fronts by performing non dominant ranking. The ranked fronts are appended to NS iteratively as long as the size of NS is not overflowed by the addition of the front. If appending the front will overflow NS, the individuals from the front are selected based on crowding distance.

Due to the biphasic nature of PPS, the framework lends itself well to information gathering. During the first phase, constraints are ignored and the population traverses the search space unhindered. In the original work, the framework utilises the known largest constraint violation to initialise the CHM. Other than this information, the framework does not utilise the benefit of exploring a larger area before performing a more concentrated search towards the constrained PF.

3.3 Informed Search

General algorithms have the benefit of being applicable to a vast range of problems. Specialised algorithms, using knowledge about the problem being solved, may produce better results but can only be applied to one specific problem [Burke et al., 2007]. Being able to incorporate problem specific knowledge while keeping the approach general may greatly enhance an algorithm.

A common approach is to incorporate preferences into the model, provided by some domain expert or decision maker. These preferences may either restrict the search space [Calborean and Vinţan, 2011] forcing the CMOEA to search through a smaller area or it can guide the search into a smaller area without directly constraining it [Jain and Deb, 2014].

Manually inserting rules or preferences requires some previous knowledge of the problem at hand. This may not always be the case, and this approach is a highly specialised one making the model less applicable to other problems. Jahr et al. [2012] used data mining methods to extract information from previous problems and then used a decision tree model to implement them in solving new ones. The results showed an increase in both convergence and HV values when using the automatically

generated information. Similarly, Lim et al. [2016] used decision trees to implement previously extracted information when solving new problems. Their results also showed an increase in performance compared to other state of the art algorithms.

Depending on the problem, the act of extracting information may be time consuming and computationally or manually heavy [Lim et al., 2016]. Also, there is no guarantee that the information extracted will be useful when solving another problem. Thus, it may not always be worth exerting effort in solving different problems and applying information from them to solve another one. Finally, there might not be any problems similar to the one to be solved. For this reason, it may be difficult extracting relevant information which may help the search. To avoid these problems one can instead extract information from the current problem being solved and use that during the search. In their work, Handoko et al. [2010] showed that utilising information gathered during search resulted in better performance and increased efficiency. This coincides with the review by Díaz-Manríquez et al. [2016] which highlight the potential increase in efficiency from using surrogate models.

Gathering information may add complexity without increasing performance [Bandaru et al., 2017]. This may be the case if the incorrect assumptions are made or the information used is pushing the search away from optimal solutions leading to premature convergence or stagnation. In the work of Pilat and Neruda [2011] it was shown that using surrogate models resulted in problems with the later phases of the evolution. Therefore, effort should be put into making sure that useful information is gathered and that it does not excessively control the search. Jahr et al. [2012] implemented domain knowledge into the mutation operator that would directly affect the transformation performed on individuals during evolution. Jain and Deb [2014] used a different approach where the domain knowledge was implemented as reference points in the landscape where the user would prefer the search to be conducted. Both approaches aim to reduce the negative impact of the knowledge restricting the search by focusing too much on specific areas. The knowledge used by Jahr et al. [2012] would have a higher probability of being used in the early stages of the search. Jain and Deb [2014] differ in that the knowledge is used through the whole search. However, in their approach the preference does not directly restrict the exploration of other areas.

3.3.1 What Knowledge Should be Used?

The more complex the problem is, the more difficult it becomes implementing information into the model that correctly represents the information of the domain expert [Calborean and Vinţan, 2011]. This may result in the domain knowledge negatively interfering with the evolution. By utilising a simpler model, surrogate models aim to increase efficiency by allowing more computations may be performed at a lower cost on a simpler model than the original objective functions. In their review, Díaz-Manríquez et al. [2016] show the potential usefulness of surrogate models where several approaches have outperformed models using the original objective functions directly.

A problem with surrogate models is that they are prone to inaccuracy if the proper model is not selected or it is poorly constructed [Díaz-Manríquez et al.,

2016]. If too much trust is put into the model, the population may converge to a false PF. As shown by Pilat and Neruda [2011], relying too much on the surrogate model stagnates convergence. This may be circumvented by using a mechanism for switching between the use of the surrogate model and the actual objective function. Using such a mechanism is heavily reliant on the switchback parameter, and poor parameter setting may produce bad results [Díaz-Manríquez et al., 2016]. Another approach is to switch between several surrogate models preferring to use the one which performs the best [Rosales-Perez et al., 2013]. This increases the changes of constructing a model which performs well in regards to the original objective functions. However, with increased number of models, there is an increased cost of constructing, using and evaluating them. Thus, the overall benefit of using a simpler model may be lost.

Instead of trying to capture the relationship between the search space and objective space it is possible to analyse how fitness changes in the landscape. This approach, known as FLA, has been used to gain a better understanding of the performance on a set of problem instances. Due to FLA being resource heavy, a question about the usefulness of the method can be raised and whether it is more beneficial to simply solving the problem [Pitzer and Affenzeller, 2012]. The research into FLA has focused mostly on neural network training and the relationship between the objective function and possible designs of the neural network [Bosman et al., 2017; Van Aardt et al., 2017]. FLA is used to get a deeper understanding of a whole problem class, and when wanting to find common aspects and differences between other problem classes [Pitzer and Affenzeller, 2012]. Other than understanding the underlying problem FLA is used for selecting a fitting solver or parts of the solver for a problem. An example of this is the use of FLA for algorithm selection for black-box optimisation problems Wang et al. [2018]. Using a low-cost framework their results show the validity of basing the choice of algorithm on the analysis of the landscape. Little to no work has been found where FLA is used in an online manner to control parameters or searching preference.

When solving CMOPs optimal regions are often located close to the boundary between feasible and infeasible regions [Handoko et al., 2010]. Knowing where these boundaries are located may be beneficial when trying to locate the PF. A potential benefit of knowing where these boundaries are located is that it reduces the search space, to a potentially much smaller area [Leguizamón and Coello, 2009]. However, when optimal solutions are not located on the boundary between feasible and infeasible regions, focusing solely on a too small region around the boundaries of feasible and infeasible space may thus alienate the search from the PF. Therefore, proper mechanisms are needed.

3.4 Boundary Search

BS is a method based on the assumption that optimal solutions often lie on the boundary between feasible and infeasible space. BS can either be in the form of a CHM or a specialised search. The common aspect is that it uses special operators to approximate these boundaries. Over the years, several BS methods have been applied. This section discusses some of them and how they handle different aspects

of constraint handling.

3.4.1 Selecting Constraint Boundaries to Approximate

It is not necessarily the case that all constraint are active at the PF. This makes it preferable to approximate some borders between feasible and infeasible space over others.

When solving CMOPs there can either be one or multiple constraints. When there are multiple constraints, some form of selection can be incorporated to decide which constraints should be approximated and which should not. Three possible ways of handling multiple constraint with BS are [Leguizamón and Coello Coello, 2009]:

1. Focus on one constraint through the whole run.
2. Focus on all the constraints.
3. Focus on all the *active* constraints.

The simplest approach is to focus on a single constraint through the whole search as there is no need for any control logic to switch between which constraint is being focused by the Binary Search (BiS) method. Additionally, effort is not put into approximating multiple constraint boundaries which are not active at the PF.

The major downside with this approach is that it requires the selected constraint to be active, which is not guaranteed unless some information about the problem is known beforehand as discussed in section 2.6. If the selected constraint is not active at the PF, the search may be alienated from promising regions. Research shows that in the case of multiple constraints being active, this approach performs poorly compared to the other two [Leguizamón and Coello Coello, 2009].

Focusing on all the constraints is by far the most common way of handling multiple constraint. This approach has the benefit of not ignoring any constraints. Therefore, no active constraints are missed, and no parts of the PF is actively excluded from the search. A common approach is to incorporate all constraints into the objective value in the form of a penalty function as described in section 2.5.1. This approach to BS has shown to improve the results when solving CMOPs [Metkar and Kulkarni, 2014; Woldesenbet et al., 2009].

If not all constraints are active, then considering all constraints results in using resources on approximating boundaries not close to the PF. Focusing only on the active constraints removes the problem of focusing the search in less promising areas along the boundaries of inactive constraints. However, this approach requires some knowledge about which constraints are active or not. This needs to be known before-hand or extracted through some automated process and implemented into the model. It has been shown that the best and most stable results are achieved when approximating constraint boundaries of only active constraints [Leguizamón and Coello Coello, 2009]. Similar results have been shown by Roald and Molzahn [2019] and Sallam et al. [2017]. Leguizamón and Coello Coello [2009] knew all the active constraints a priori. This differs from the work of Roald and Molzahn [2019] and Sallam et al. [2017] where the active constraints were identified during the

evolution. The methods showed the ability to identify active constraints, but they also heavily rely on correctly identifying them. Even still, with the added cost of identifying the active constraints, better performance was achieved in both cases. However, contradicting results have also been found in the work of Bonyadi and Michalewicz [2014]. Considering only active constraints proved to perform worse than considering both active an inactive constraints.

When approximating multiple constraint-boundaries there is an option to either approximate a single or multiple constraints at a time.

In the case of considering one constraint at a time, there is a need for some logic to control which constraint should be approximated. A simple approach to this is to allocate a certain number of generation to each constraint going in a round-robin fashion [Leguizamón and Coello, 2009]. Other than this work, no other contributions only focusing at a single constraint at a time has been found.

The most common approach is to implement the constraint handling as a part of the evolutionary process. This is done either through the use of penalty functions [Woldesenbet et al., 2009] or some sort of ranking based on the objective function and the constrain values [Metkar and Kulkarni, 2014]. Common with both approaches is that there is no need for any extra control logic, and different variations have shown that they have consistently good performance. Different variations also exist where either some form of weighing or extraction of a subset to focus on is used [Bonyadi and Michalewicz, 2014].

3.4.2 Approximating Boundaries

A characteristic of BS is the reduction of the search space. This is due to the focus on the intersection between feasible and infeasible regions. It is therefore paramount to select a suitable method of approximating these areas.

By turning the constraints into objective values for optimisation the constraint handling is incorporated into the evolutionary process. By basing the selection of individuals on a ranking system using the objective values, individuals closer to borders are preferred over those further away [Metkar and Kulkarni, 2014]. This approach is simple, and the population is naturally drawn towards the boundary regions. Both Sallam et al. [2017] and Bonyadi and Michalewicz [2014] reduced the search space by creating a boundary around the constraints borders. The difference between the two is that Sallam et al. [2017] would only focus constraints identified as active, while Bonyadi and Michalewicz [2014] showed that also considering inactive constraints yielded good results. Similarly to the approach of Metkar and Kulkarni [2014], the use of boundaries would naturally push the population towards the desired regions during the evolutionary process.

The boundaries can either be static [Bonyadi and Michalewicz, 2014] or they can gradually shrink [Sallam et al., 2017]. By reducing the boundary size, the population will edge closer and closer towards the border between feasible and infeasible regions. For this method to be efficient, it is necessary to have proper parameter setting for the reduction of the boundary. If the boundary shrinks too slowly, the population may have problems approximating the PF properly. Conversely, if it shrinks too fast, the population may pass by, or be hindered in reaching optimal locations.

Similarly, with static boundary sizes proper parameter setting is key. With too large boundaries, it is easier to find feasible individuals, however the selection pressure towards the constraint boundary is reduced. On the other hand, with a too small boundary the population will have greater difficulty with locating feasible solutions.

Leguizamón and Coello [2009] differ in their approach from the ones discussed above in that a BiS operator was used to close in on the border between feasible and infeasible space. This method does not shrink the search space like the ones in [Sallam et al., 2017] and [Bonyadi and Michalewicz, 2014], but pairs up feasible and infeasible individuals and moves them closer and closer to each other. For this to be possible, population must consists of both feasible and infeasible individuals. This method also requires more logic, in addition to the normal evolutionary process. Finally, some method of handling overlapping infeasible areas is required for the BiS method to work properly.

3.5 Binary Search

Leguizamón and Coello Coello [2009] proposed an operator using binary search between pairs of individuals. Given a pair x and y, where x is in the feasible space and y is in the infeasible space, the middle point m is calculated. If m is located in feasible space, then x is moved to m and conversely if m is in infeasible space. This continues until some stop condition is met.

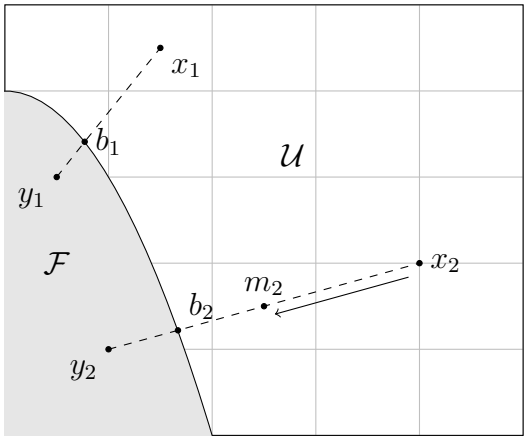


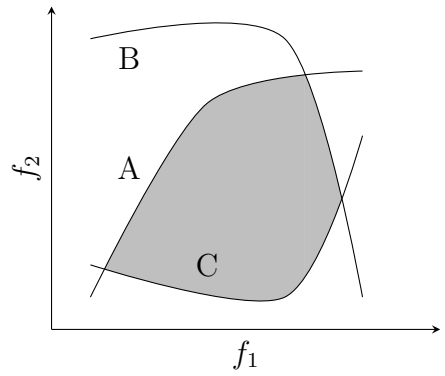
Figure 3.1: Boundary search using binary search between pairs of points.

This binary search approach is illustrated in figure 3.1. $\mathcal{U}$ denotes the feasible space and $\mathcal{F}$ denotes the infeasible space. x_i and y_i denote pairs of feasible and infeasible points with b_i being the point on the border between the two points. The goal is to approximate b_i for each pair. For the pair x_2 and y_2 we see that the middle point m_2 has been calculated. Due to it being feasible, x_2 is moved to m_2 as illustrated with the arrow. This continues for each pair until the stopping condition is met. The condition used is defined in equation (3.5).

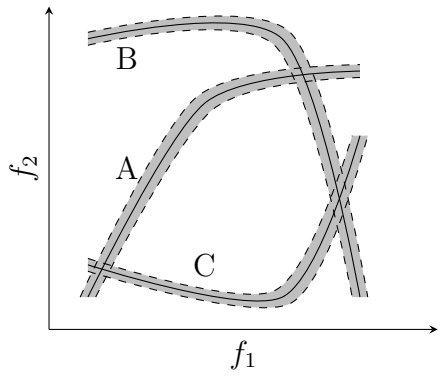
$$DtB(m) \leq \xi \text{ AND } feasible(m) \tag{3.5}$$

where $DtB(m)$ is the distance from the middle point to the boundary and ξ is a user defined parameter. $feasible(m)$ denotes that the point is feasible or not.

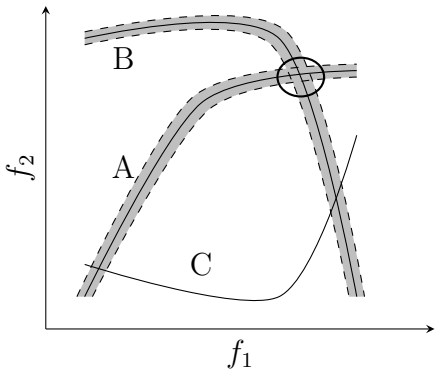
3.6 Reduced Search Space



(a) Feasible space surrounded by three constraints.



(b) Reduced search space round constraint borders



(c) New feasible space defined by reduced search space around active constraints.

Figure 3.2: Reduced search space.

Sallam et al. [2017] reduced the search space to only focus on active constraints. Figure 3.2 illustrates the method of reducing the search space. A, B, and C are constraint boundaries surrounding the feasible space, as illustrated in figure 3.2a. To create the reduced search space, a border around the constraint boundaries is created as illustrated in figure 3.2b. Assuming that only A and B are active constraints, the reduced search space around C is ignored. Now, the new feasible space is in the cross section between A and B as illustrated in figure 3.2c.

A more in-depth explanation of the steps is given below.

3.6.1 Active Constraints Detection

Active Constraints Detection (ACD) is the act of identifying constraints that *may be* active. It is important to note that even though a constraint is identified as active by Reduced Search Space (RSS) it is not guaranteed to actually be that.

A random individual from the lowest rank in the population is selected. The distance of this individual to the boundary of each infeasible space is then used to determine if the constraint is active or not, as shown in equation (3.6) [Sallam et al., 2017] :

$$a_i = \begin{cases} true & \text{if } 0 \leq |g_i(x)| \leq Val \\ false & \text{else} \end{cases} \tag{3.6}$$

where a_i denotes if constraint g_i is active. $g_i(x)$ is the constraint value of individual x for the constraint g_i. Val is a predefined value, usually low. ACD is concerned with the distance of an individual to the constraint boundary and not whether or not the constraint is upheld. For this reason the absolute constraint value is used. The constraints that are not viewed as active are ignored during the constraint handling. ACD is performed at a certain interval defined by a user-specified parameter. Each time ACD is performed, a new individual is selected.

3.6.2 Creating Boundary Areas

After ACD is performed, a boundary area is created for each active constraints. This way, the search space is reduced, focusing on these boundary areas. Each area is defined by equation (3.7) below:

$$-\delta_{in} \leq g_i \leq \delta_{out} \tag{3.7}$$

$$\tag{3.8}$$

where δ_{out} is used to include parts of the infeasible regions around the active constraint g_i. δ_{in} is used to include regions of the feasible space around the active constraint.

The initial value of δ_{out} depends on the feasibility ratio of the population. If less than 20% of the population is feasible, then the initial δ_{out} value is set to the maximum constraint violation of the top 20% of the population. If more than 20% of the population is feasible, the initial δ_{out} value is set to 1.

δ_{in} is simply initialised to be a large value to avoid alienating the feasible regions from the search.

3.6.3 Shrinking the Boundary Areas

Both δ_{out} and δ_{in} are updated during evolution. This is to focus the search closer and closer to the boundary between feasible and infeasible space. For each generation during the pull stage, the deltas are updated as described in equation (3.9) and equation (3.10).

$$\delta_{out}(t) = \begin{cases} \delta_{out}(0) \times (1 - \frac{cfe}{FES_c})^z, & cfe \leq FES_c \\ 0, & cfe > FES_c \end{cases} \tag{3.9}$$

where $\delta_{out}(t)$ is the size of the boundary areas on the infeasible side, at generation t. $\delta_{out}(0)$ is the initial delta value. cfe is the current number of function evaluations performed during a run and $FESc$ is a user defined value representing the maximum number of function evaluations before shrinking the boundary area on the infeasible side to 0. Dividing cfe by $FESc$ creates a non-negative number below 1 which then

allows the boundary to be shrunken for each generation. z is a parameter used to control the reduction of δ_{out}.

$$\delta_{in}(t) = \begin{cases} \delta'_{in} = \delta_{in}(0) - cfe \times \frac{\delta_{in}(0) - \delta_{in}^{min}}{MAX_{FES}}, & \delta'_{in} < \delta_{in}^{min} \\ \delta_{in}^{min}, & \delta'_{in} \geq \delta_{in}^{min} \end{cases} \tag{3.10}$$

where $\delta_{in}(t)$ is the size of the boundary area from the feasible side at generation t. $\delta_{in}(0)$ is the initial delta value set when changing to the pull stage. cfe is the current number of function evaluations performed and δ_{in}^{min} is the minimum allowed value for δ_{in} set to $0.002 \times \delta_{in}(0)$. MAX_{FES} is the maximum number of function evaluations before evolution is stopped. δ_{in}^{min} is used to prevent δ_{in} from ever reaching 0, while still keeping the boundary close to this value. If the boundary is set to be exactly on the border between feasible and infeasible space, then the search space will be reduced to an extremely small area making evolution difficult. δ_{out} on the other hand is eventually set to 0 as the final solutions need to be feasible. This is to increase the selection pressure towards feasible individuals at the end of the run.

Chapter 4

Model

This chapter describes the proposed algorithmic model based on the research into the state og the art described in chapter 3. Section 4.1 introduces the extensions made to PPS and explains design decisions made regarding the use of Boundary Search (BS). Then, section 4.2 illustrates the flow of the general model, using several flowcharts. The first flowchart shows a high-level overview of the model and each phase is described in more detail. A simulator developed as part of this thesis is outlined in section 4.3. Finally, an overview of the relevant parameters for PPS, BiS and RSS is given in section 4.4.

4.1 Extending PPS with BS

The original PPS framework has the ability to gather information during the initial phase, without the information affecting the population. The information could be utilised for better initialisation and parameterisation of the later phases, increasing the total performance of the algorithm. This work extends PPS with two different Boundary Search (BS) methods.

The first approach, described in section 4.2.8, adds a third phase in between the push and pull phase. The binary phase aims at moving the population closer to the boundary between feasible and infeasible space before the pull phase is initialised. The second approach replaces the CHM in the pull phase with a method approximating the boundary between feasible and infeasible space. This approach is further elaborated in section 4.2.10.

In addition to the extension of the functionality of framework by BS, visualisation and logging tools have been implemented to aid in analysing the framework and the suggested extensions. To differentiate between the different approaches, the following names and corresponding acronyms have been given to them:

- Push Pull Epsilon (PPS-E): The original PPS framework, using ϵCH.
- Push Pull Improved Epsilon (PPS-IE): The original PPS framework, using IϵCH.
- PPS: Refers to both PPS-E and PPS-IE.

- Push Binary Pull Epsilon (BiS-E): The original PPS framework, using ϵCH, extended with BiS.
- Push Binary Pull Improved Epsilon (BiS-IE): The original PPS framework, using IϵCH, extended with BiS.
- Push Binary Pull Search (PBPS): Refers to both BiS-E and BiS-IE
- Push Pull Reduced Search Space (PPS-RSS): The original PPS framework, extended with RSS.

These abbreviations are used in the subsequent chapters when discussing the different model specifications tested.

4.2 Proposed Framework Flowchart

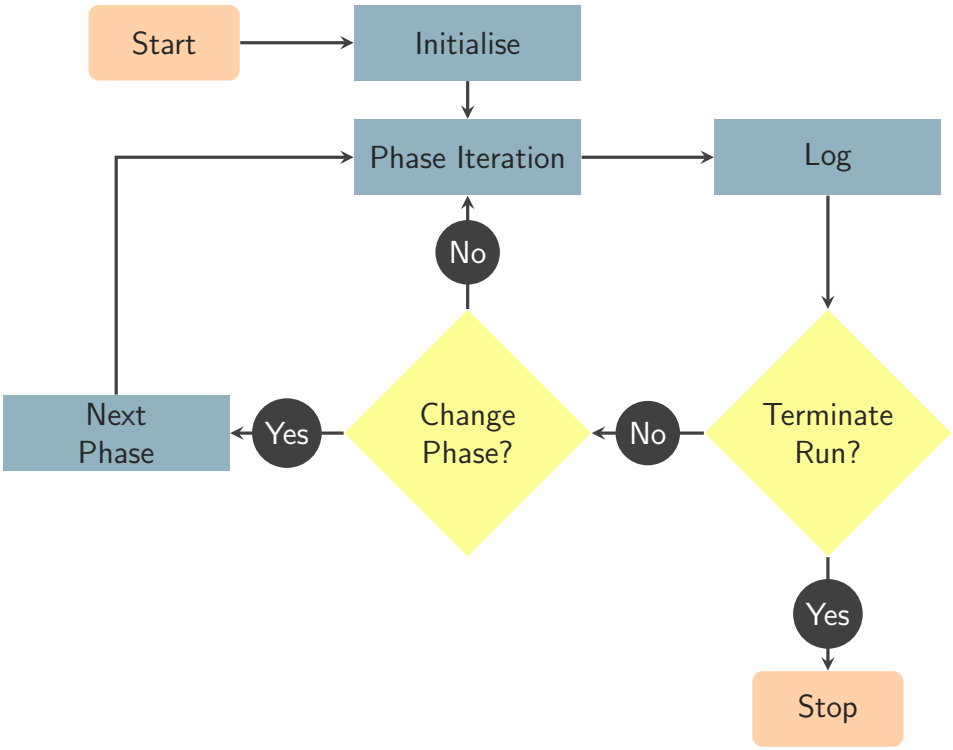


Figure 4.1: Flowchart illustrating the proposed model.

Figure 4.1 presents an overview of the PPS framework. Each node in the flowchart represent a step in the algorithm and is explained further in this section.

4.2.1 Initialise

The first step is to initialise the framework. The initialisation entails setup of which phases will be performed, the underlying MOEA and the selected CHM. Parameters for the framework are set (see section 4.4), and an initial random population is generated.

The setup of phases is simply an ordered list of all the phases to be entered during the run. MOEA/D (see section section 2.4.1) is implemented as the MOEA. As previously mentioned in section 2.4.1, the population count in MOEA/D is equal to the number of weights. The implementation of MOEA/D in this thesis initialises the weights according to the approach by [Zhang and Li, 2007], where weights are generated from the values in the set $\left\{\frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H}\right\}$. H needs to be defined to generate the set, and is the controlling parameter for the population size.

In the original implementation of PPS [Fan et al., 2019b], the decomposition strategy is implemented as $g^{te}(x|\lambda^i, z^*) = max_{j=1,\ldots,m}\frac{|f_j(x)-z_j^*|}{\lambda_j^i}$ unlike the equation defined by Zhang and Li [2007] (see equation (2.5)). If using the original weight generation approach, at least one weight would yield the number 0 for the denominator in the decomposition equation. Modifications to the original approach are made by removing the values that would cause vectors containing the number 0. The implementation generates weights from the set equation (4.1).

$$\left\{\frac{1}{H}, \ldots, \frac{H-1}{H}\right\} \tag{4.1}$$

The population count can be calculated by a defined H, and the number of objectives m, by equation (4.2).

$$N = \binom{H-3+m}{m-1} \tag{4.2}$$

Further description of the framework will assume that the weights are initialised.

4.2.2 Phase Iteration

After initialisation, the evolutionary process begins. How a generation is evolved depends on the current phase and requires different explanations. The evolution during the push phase is explained in section 4.2.7. The binary phase is elaborated in section 4.2.8. Finally, the pull phase is elaborated in section 4.2.9. Note that the model allows any and all of the phases to be used, creating the opportunity for different combinations.

4.2.3 Log

When logging is utilised, the framework logs relevant data for each generation. This information may include the current $\mathrm{FR_p}$, IGD and HV of the current population and the archive. In addition, the framework saves a plot of the current generation used to create a video of the whole evolutionary run. The main purpose of the logging functionality has been to aid in the analysis of the new framework. However, it may also be used as a live feed of how the run is performing.

4.2.4 Terminate Run?

After the logging step, the framework checks if the halting condition has been met. This is the case if T_{max} (see table 4.1) has been reached. If it is time to terminate the

run, the framework exports a video from all the plots created during the evolutionary run, and returns the resulting archive of the current generation.

4.2.5 Change Phase?

If the run is not terminated, the framework checks if it should change phases. The run terminator is a parameter (see section 4.4) and is global. Phase termination is local for each phase, and the condition for changing phases depends on the current phase. The termination of each phase is elaborated in more detail during the description of each phase. If the condition for changing phases is not met, the framework continues with the next generation.

4.2.6 Next Phase

In the case that the phase should change, the next step is to change phases. All phases are stored in a common list and the Next Phase step simply changes the current phase to the next. Then, the evolution carries on to the next generation, evolving the population as dictated by the new phase.

4.2.7 Push Phase

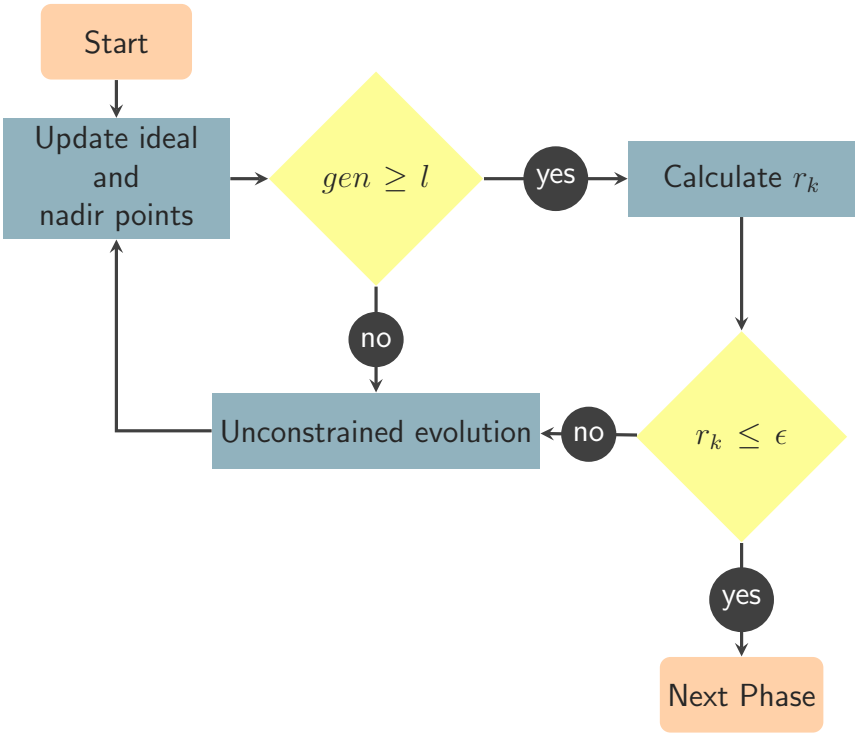


Figure 4.2: Flowchart illustrating the push phase.

Figure 4.2 illustrates the flow of the push phase, which remains unchanged from the original work of Fan et al. [2019b]. The goal of this phase is to reach the unconstrained PF before proceeding to the next phase.

The first step is to **update the ideal and nadir points** which are required for the break condition (see equation (3.1)). At the beginning of the run, the ideal and nadir points are calculated using the initial population. Every subsequent generation uses the newly evolved population to update the ideal and nadir points.

$gen \geq l$ signifies that l generations have passed since the initiation of the run. If this is this is the case, then r_k is calculated according to equation (3.1). In the case that r_k is less than or equal to a predefined ϵ, the framework identifies the push phase as complete and initiates the next phase.

If either $gen \leq l$ or $r_k > \epsilon$, then the framework remains in the push phase and evolves the population. As long as the break condition for the phase is not met, the phase proceeds to **Unconstrained evolution** which evolves the population without taking constraints into consideration. Exactly how the population is evolved to the next generation depends on the underlying MOEA. In the implementation of this thesis, MOEA/D (see section 2.4.1) is implemented as the CMOEA.

4.2.8 Binary Search Phase

The push phase is terminated when the change in nadir- and ideal points is less than some threshold, as described in equation (3.1). Figure 2.6b visualises problems where the unconstrained and constrained PFs have a significant gap when switching from unconstrained to constrained optimisation. For problems sharing this characteristic, individuals have to gradually evolve back through explored space during the pull phase, before reaching the desired boundary between the constrained and unconstrained space. To reduce the search in infeasible space, a guided search towards the boundary is proposed, by exploiting an archive of feasible solutions. This approach is implemented to be used as a phase between the original push and pull phases. This phase is called BiS.

To allow the population to move towards the constrained PF, two sets are used.

1. A feasible set, containing feasible individuals.

2. An infeasible set, containing infeasible individuals.

The feasible set is a copy of the archive used in PPS (see section 3.2). The archive is updated during the push phase and contains the discovered feasible individuals closest to the constrained PF when BiS is initialised. The infeasible set will contain the infeasible individuals in the population.

Figure 4.3 illustrates the phase iteration during the binary phase. The first step of BiS is to **Create Pairs** of feasible and infeasible individuals. This is only performed once at the start of the phase. This step is elaborated in section 4.2.8.1. The next step is to **Move Individuals** closer to the boundary between feasible and infeasible space by performing BiS. Section 4.2.8.2 elaborates this step. Then, a check is performed to verify if the stop condition is met. If this is not the case, then the algorithm **Removes Pairs** within a certain distance measure from the BiS, as described in section 4.2.8.2. Furthermore, the process of moving the remaining pairs of individuals is repeated. Finally, section 4.2.8.3 elaborates how the model **Selects Individuals** to become the new population for future generations in the pull phase when the stop condition is met.

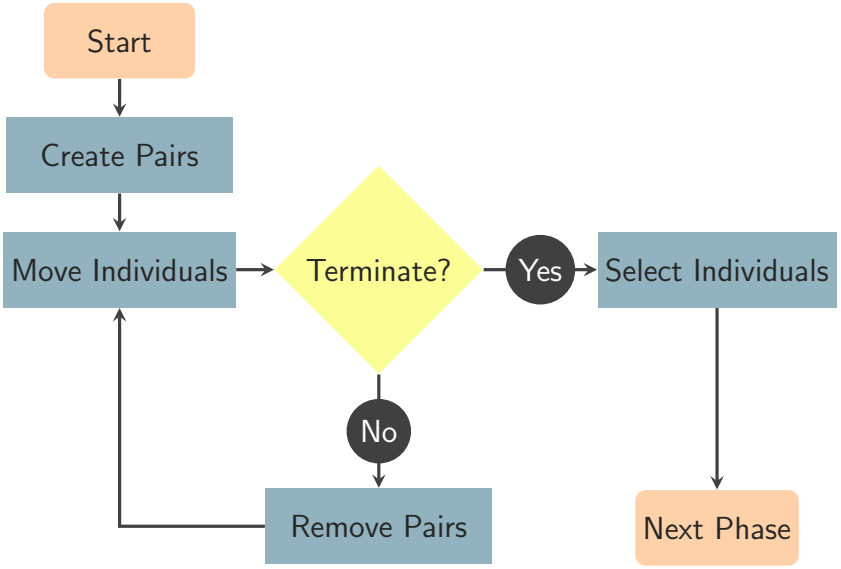


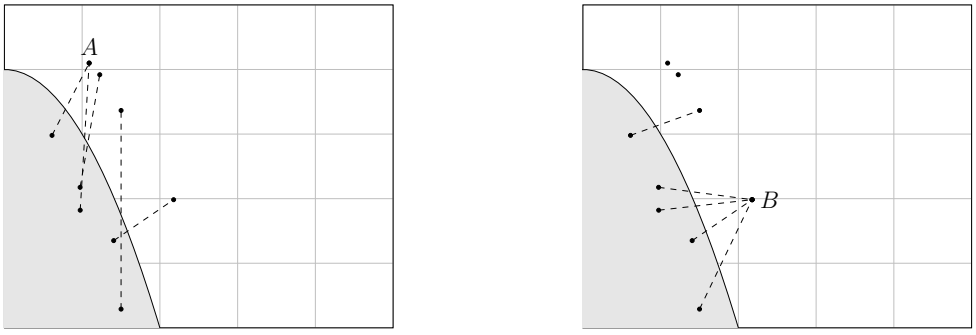
Figure 4.3: Flowchart illustrating the proposed binary phase.

4.2.8.1 Pairing Strategy

When pairing feasible and infeasible individuals two strategies are discussed with the goal of improving convergence towards the PF while taking diversity and spread of the population into account.

1. Random allocation
2. Closest neighbour allocation

To make the discussion comprehensible, a few definitions are required. The size of the the *feasible* set is defined as N_f and the size of the *infeasible* set as N_i. Pairs where the feasible individual is only paired with a single infeasible individual, are referenced to as exclusive pairs. Pairs where the feasible individual is paired with multiple infeasible individuals are referenced to as open pairs.



(a) Random allocation. **(b)** Closest neighbour allocation.

Figure 4.4: Visualisation of pairing strategies.

The random allocation strategy is performed by iterating over the infeasible set and pairing each index in the set with a randomly selected individual in the feasible

set. Figure 4.4a visualises a feasible set with 4 individuals and an infeasible set with 5 individuals. For the exclusive pairs, the intersection between the dashed line highlighting the pair, and the boundary, shows where the individuals will reach the boundary. As seen in the figure, two infeasible individuals share the feasible individual A, which is described as an open pair. This pair will not reach the boundary on the intersection, as the individual A will be affected by both of the individuals in the open pair. Based on this observation, many exclusive pairs would yield a better coverage when all pairs have reached the boundary. Having few open pairs would mean clustering towards a single point. It is thus believed that minimising the number of open pairs should be a priority.

The closest neighbour allocation strategy ensures the closest individuals from each set to be paired. All distances between individuals are evaluated before pairing the infeasible individual with the closest feasible individual. As mentioned by Fan et al. [2019b], the PPS framework is prone to problems where the unconstrained PF has one optima and individuals are converging towards a single point. In this scenario, all infeasible individuals would have the same distance to a given feasible individual, as their position in the infeasible space would be equal. The result of this would be a single open pair where all infeasible individuals share the same feasible individual. A similar scenario may occur, as visualised in figure 4.4b. The figure shows multiple infeasible individuals sharing an open pairing with the feasible individual B. This scenario is also possible with the random allocation strategy. However, using random allocation makes this less likely.

Based on this short discussion, it is clear that open pairs is possible for both strategies. However, section 5.1.1.3 further shows that the earlier hypothesised downside of open pairs could be neglectable. The random allocation is a more efficient approach, as calculating the distances prior to pair allocation is unnecessary. Additionally, the random allocation is better suited to pair infeasible individuals with multiple feasible for scenarios mentioned earlier with a single optima. This work implements the random allocation strategy, but proposes the possibility of utilising closest neighbour.

4.2.8.2 Distance Measure

Different approaches for evaluating similarity between individuals could be utilised as a measure for the difference between the individuals within a BiS pair. A simple approach is calculating the distance between the pair,and stop updating the pair when some distance is reached. The model assumes that the feasible set is close to the boundary and that a distance between the pair being within a threshold, γ, in euclidean distance is a sufficient measure. Distances in two different spaces are discussed with the goal of finding the most promising measure to stop the search as close to the boundary as possible.

1. Distance in search space
2. Distance in objective space

In the case of measuring the distance in search space, there exists cases where two genotypes are close, but the objective values have a significantly wider gap. Ad-

ditionally, the search space usually consists of more dimensions than the objective space, meaning a higher cost of calculation. For problems with decision constraints [Liu and Wang, 2019], this space might be required to search through as the boundary is defined in the search space. This work focuses on solving CMOPs with objective constraints and the goal is to reach some distance in the area of feasible space which is defined by objective constraints. The distance measure implemented in this work is therefore measured in objective space.

The distance between the feasible individual x_a and the infeasible individual x_b in the objective space $\mathbb{R}^m$ is measured in euclidean distance as described in equation (4.3).

$$dist(x_a, x_b) = \sqrt{\sum_{i=1}^{m}(f_i(x_a) - f_i(x_b))^2} \tag{4.3}$$

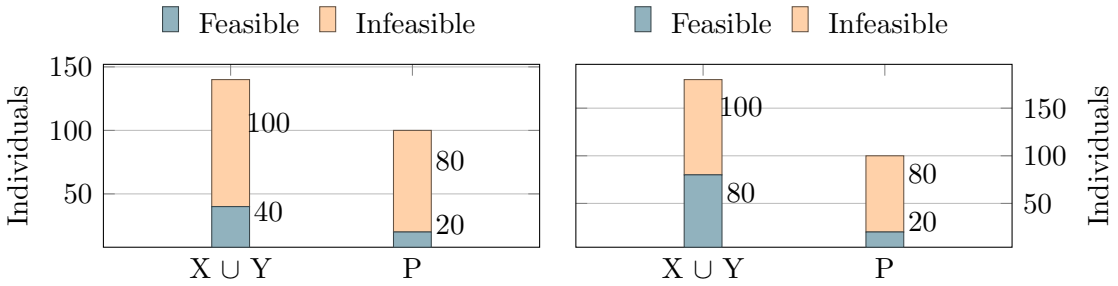
The distance is used to evaluate if the pairs are close enough to stop the phase and proceed to the pull phase. One iteration of the BiS phase iterates over every pair. The distanced between each pair is measured and if the distance is greater than γ, an offspring is created by calculating the midpoint between the genotype of the two individuals. If the distance is less than or equal to γ, the pair between the individuals is removed and no new offspring is produced from the individuals. The feasible or infeasible individual is overwritten if the offspring is feasible or infeasible respectively. The phase stops if no pair has been updated to a new position.

4.2.8.3 Selection Strategy

When the BiS of the phase is stopped, the union size of the two sets will be greater than the allowed population size. To aggregate the sets to an appropriate size, some individuals must be selected to the pull phase population and some must be discarded. At the end of the BiS the two sets consist of feasible individuals with a size of N_f and infeasible individuals with size N_i. Different weighting of selecting feasible and infeasible individuals will result in different diversity in the population at the start of the pull phase. The selection strategy proposed for selecting individuals for the population P with size N_p, from the feasible set X and the infeasible set Y, where $X = (x_1, \ldots, x_{N_f})$, $Y = (y_1, \ldots, y_{N_i})$ is described in equation (4.4)

$$p_i = \begin{cases} x_i, & \text{if } r \leq \kappa\frac{N_p}{N_f} \text{ and } i \leq N_f \\ y_i, & \text{otherwise} \end{cases} \tag{4.4}$$

Selection in EAs is usually performed by randomly selecting elements with a given probability (see section 2.3.1.3). The goal of the selection at the end of BiS is to provide the population with some $\mathrm{FR_p}$. $\kappa\frac{N_p}{N_f}$ defines the probability of selecting feasible individuals out of N_p pulls such that the population will have approximately the $\mathrm{FR_p}$ κ. Figures 4.5a and 4.5b visualise the effect of the selection parameter for two different feasible sets.



(a) Feasible set with size 40 and the resulting population.

(b) Feasible set with size 80 and the resulting population.

Figure 4.5: Effect of $\kappa\frac{N_p}{N_f}$, visualised with $\kappa = 0.2$, N_p=100.

After selection is finished, the max constraint violation is updated to the max violation of population after selection. IεCH and εCH with a high max violation would pull the population far away from the PF at the start of the pull phase. The update allows exploration in the area around the boundary.

4.2.9 Pull Phase

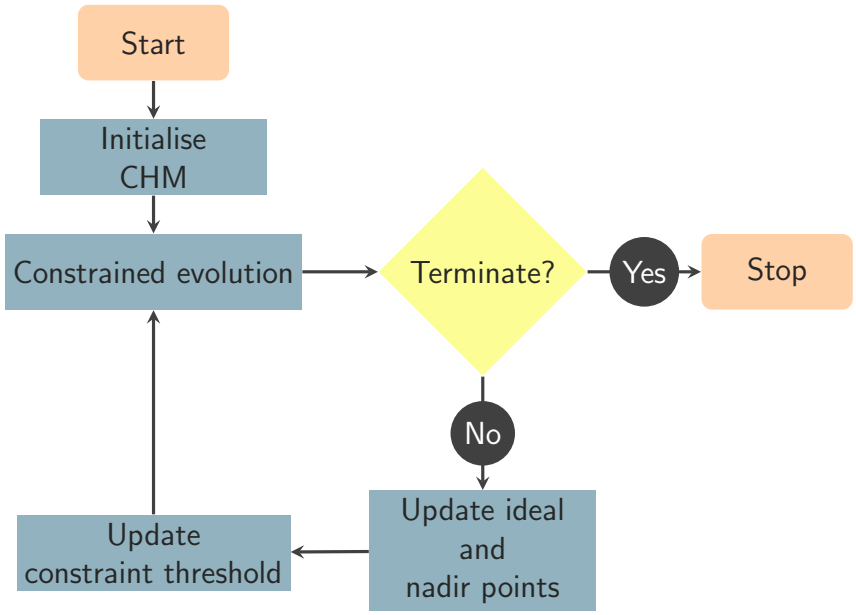


Figure 4.6: Flowchart illustrating the pull phase.

The third and final phase is the pull phase. The phase is illustrated in figure 4.6.

The first step of the phase is to **initialise the CHM**. In the implemented framework, the three possibilities are εCH, IεCH and RSS. For all methods, initialisation entails setting the initial constraint relaxation. For RSS, an additional step is performed: ACD, which is described in section 4.2.10.1.

The next step is **Constrained evolution** which evolves the population with constraints taken into consideration. Similarly as in the push phase, the underlying MOEA dictates exactly how the population is evolved. MOEA/D (section 2.4.1) is

implemented as the MOEA in this work. As constraints are considered in this phase, MOEA/D requires to be extended. This work extends MOEA/D with IϵCH and ϵCH according to the implementation in Fan et al. [2019b]. Additionally MOEA/D with RSS is proposed, further explained in section 4.2.10.

The phase continues until T_{max} is reached. If the halting condition has not been met, the **ideal and nadir points are updated** using the newly evolved population. Then, the next step is to update the CHM. For ϵCH this simply entails shrinking the allowed constraint violation as described in section 2.5.1. IϵCH will either shrink the allowed constraint violation, or it will relax it further as described in section 3.2. Similarly, RSS will shrink the boundaries around the active constraints to continuously force the population closer to the border between feasible and infeasible regions. When T_c or $FESc$ is reached (see section 4.4, the relaxation of constraint is set to 0.

Below is an in-depth explanation of the RSS method implemented in this work and how it affects the search in the pull phase. The implementation is based on the work of Sallam et al. [2017]. Design decisions and changes are highlighted herein.

4.2.10 Boundary Search with Reduced Search Space

RSS replaces the CHM during the pull phase. RSS uses both information about which constraints are active at the constrained PF and the amount of constraint violation to reduce the search space. RSS creates a new border around the edge between feasible and infeasible space for the active constraints (see figure 3.2b). After creating the borders, individuals outside them are considered infeasible and individuals inside are considered feasible. This results in a priority of individuals that are close to the boundary between feasible and infeasible regions. The new feasible space is gradually shrunken to focus the search.

4.2.10.1 Active Constraints Detection

ACD is performed as described in section 3.6.1. However, a few changes were made to make the work of Sallam et al. [2017] fit better with the PPS framework:

- ACD is only performed once.
- Multiple individuals are used for ACD.

PPS ignores constraints during the first phase of the framework. As the population moves unhindered by infeasible regions, it is likely that multiple constraint boundaries are approached and passed during this phase. Due to constraints being ignored, information regarding which constraint boundaries the population is close to is not needed. Also, performing ACD during this part of the search will waste computations as the knowledge of which constraints the population is close to is not used until constraints are being considered. For these reasons, it was decided to only perform ACD once, when switching from the push phase to the pull phase. It would have been possible to perform ACD multiple times during the pull phase. However, when PPS switches from the push to the pull phase, the population is assumed to have approximated the unconstrained PF. If the constrained PF is not the same

as the unconstrained one, then the infeasible region(s) separating the two PFs will likely have a boundary close to the unconstrained PF and the need for re-evaluating which constraints are active and not is reduced.

When performing ACD only once, there is a strong bias created due to the dependency of the single individual selected for ACD. With a single random individual selected, the approximation of constraint boundaries and the performance of the search is highly dependent on this single individual. To reduce this dependency, multiple individuals are used. First non-dominating sort is performed on the population, then from the ranked population a set of $NumACD$ individuals are selected beginning with the lowest rank. Then ACD is performed as described by equation (4.5):

$$a_i = true \text{ if } \exists x \in S, 0 \leq |g_i(x)| \leq Val \tag{4.5}$$

where a_i denotes if constraint g_i is active. x is some individual in the set of individuals, S, selected for ACD. This way, only constraints which none of the individuals are close enough to will be classified as inactive. If the original ACD method was used, as described in equation (3.6), then the last individual used for ACD would potentially override the information of the previously used individuals.

Applying the changes explained above, there is a larger possibility for multiple active constraints to be identified, as the individuals selected will cover a larger part of the search space. Furthermore, the probability of incorrectly classifying active constraints as inactive is reduced. However, there is also a larger probability of classifying inactive constraints as active. The reason is that individuals will be located in different areas of the landscape and thus there is a larger probability of them being closer to inactive constraints. This is viewed as less of a problem, as false positives are preferred over false negatives. They are preferred due to them making the search space larger than it could have been, while false negatives removes possibly optimal locations from the search space.

4.2.10.2 Creating Boundary Areas

After ACD, a boundary area is created around each identified active constraint as described in section 3.6.2.

If ACD did not identify any constraint as active, then the model does not reduce the search space. The total constraint violation is used in combination with feasibility rules to select individuals for the next generation. The lack of constraint relaxation is due to the assumption that if no active constraints are identified then the unconstrained and constrained PF is the same. There is also a possibility of Val being set to a too low value, and thus the active constraints are not identified correctly using equation (4.5). However, no method for guarantying no false negatives exists.

4.2.10.3 Shrinking the Boundary Areas

Initially, when changing from the push phase to the pull phase, both delta values, $\delta_{out}(0)$ and $\delta_{in}(0)$, are set to the be maximum constraint violation of an individual

in the working population. This differs from the work of Sallam et al. [2017] as described in section 3.6.2. It is assumed that the population has approximated the unconstrained PF when the push phase is over. Thus, a reasonable size for the boundary is the current largest distance of an individual in the population to a constraint boundary. This way, the population will not keep moving further into constrained space, but rather be forced to move towards feasible space. Also, the applicability of setting of the boundary to 1 by Sallam et al. [2017] is highly dependant on the problem at hand, where a boundary size of 1 may either be either too large or small.

Both δ_{out} and δ_{in} are updated during evolution. This is to focus the search closer and closer to the boundary between feasible and infeasible space. The updating scheme follows the work of Sallam et al. [2017] as described in section 3.6.3.

The two borders are updated differently to put more selection pressure on feasible individuals. This is achieved by shrinking δ_{in} much slower than δ_{out}.

4.2.10.4 Evaluating Feasibility

If the problem has no active constraints, the feasibility of an individual is determined by the total constraint violation as described in equation (4.6).

$$\Psi(x) = \sum_{i=1}^{m} g_i(x) \tag{4.6}$$

where $\Psi(x)$ denotes the total constraint violation of the individual x. $g_i(x)$ is the violation of constraint g_i by individual x and m is the total number of constraints.

In the case that the problem has active constraints the constraint violation is calculated as described in equation (4.7):

$$\Psi_{R2S}(x) = \sum_{i=1}^{m} \begin{cases} \min(|L|, |R|), & \text{if } C \\ 0, & \text{else} \end{cases} \tag{4.7}$$

$$L = \delta_{in} - |\min(0, g_i(x))| \tag{4.8}$$

$$R = \delta_{out} - \max(0, g_i(x)) \tag{4.9}$$

$$C = 0 \geq L > \delta_{in} \text{ and } 0 \geq R > \delta_{out} \tag{4.10}$$

where $\Psi_{R2S}(x)$ denotes the total constraint violation of the individual x. L denotes the distance individual x is from the border on the feasible side and R denotes the distance individual x is from the border on the infeasible side. C is a condition used to determine whether x is withing the boundaries set by δ_{in} and δ_{out}. If the condition is upheld, then x is outside the boundaries and the constraint violation of the individual is increased. If the condition is not upheld, the individual is inside the boundaries, and the constraint violation should not be increased.

4.3 Simulator

A simulator has been created for the proposed framework to be tested. The simulator consists of seven different modules, as illustrated in figure 4.7. The simulator has

been designed to be a modular system using interfaces for the various modules to accommodate multiple different implementations. The interfaces define the common methods required for the model to work, but does not specify a specific implementation itself. This results in the inner workings of each interface being independent of each other, making the system flexible and open for later additions or changes. Each module is described below.

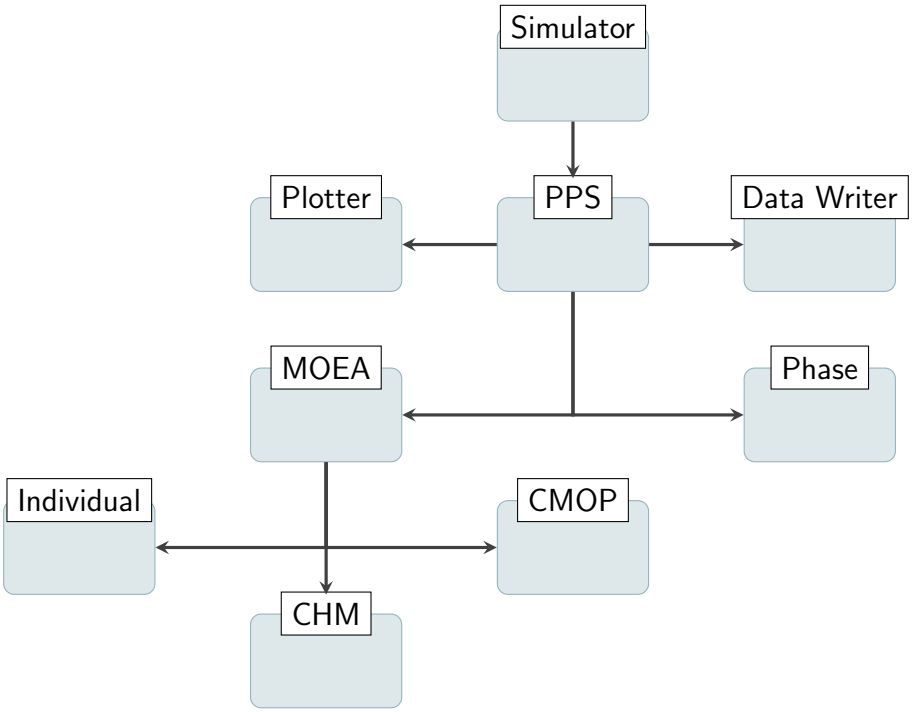


Figure 4.7: Overview of simulator modules.

The **Simulator** module is at the highest level, and is the module that the user interacts with. This module works as a wrapper around the whole system to set up, run and tear down simulations. Each simulation starts with the initialisation of the Simulator module using the user-defined parameters described in table 4.1.

The simulator is built to be initialised with a test suite of problems, where each problem is run a set amount of times and the result archive for each run is extracted. The simulator initialises every PPS module concurrently and each PPS module is able to run in parallel for enhanced performance utilising multiple CPU cores.

The **PPS** module performs a single run of a problem and communicates with the **Data Writer** and the **Plotter** to gather information about the population during the run and store this information. This module also communicates with the **MOEA** module which handles evolution of the population and the **Phase** determining which phase the algorithm is in at different stages of the run.

The **Plotter** allows the simulator to plot the population in the objective space as it evolves. This is achieved through the two exposed methods $Plot()$ and $ExportVideo()$. In this work, the plotter plots the current population at each generation and exports it as an image. At the end of each run of PPS a video showing how the evolution transpired is exported using all images generated from the run.

The **Data Writer** module handles data gathered from the population at each generation and stores them in a text file. This data can then be used for analysis of the module with the given parameters. In this work, the Data Writer stores the generation count, phase, FR_p, IGD and HV. The last three values can be stored both for the population and for the archive, during the run. This way, both the evolution itself can be monitored, and the achieved best feasible solutions.

The **MOEA** module defines the required methods to evolve the population. The MOEA module does not handle constraints directly. CHM is handled in a dedicated module which interacts with the MOEA module. In this work, only MOEA/D has been implemented. It is however possible to incorporate other MOEAs as long as they implement the interface defined by the simulator.

The **Phase** module keeps track of which phases the run consists of. The simulator takes in a set of phases that changes the behaviour of the algorithm when different criteria are fulfilled. This work has implemented the three phases: Push, Binary and Pull as described above.

The **Individual** module represent the population individuals. Having the functionality of the individuals separated into a module allows for different representations of individuals or solutions to the specified problem.

The **CMOP** module allows for different problems to be designed and implemented. It has been used during preliminary and experimental testing to test the model on multiple different benchmarks. All relevant information about the problem is fetched from this module. In this work the following test suites have been implemented: LIR1-12 and MW5, MW6, MW9, MW10, MW11, MW13. These are are elaborated in section 2.7.

The **CHM** module allows for different CHM to be implemented. In this work, three different methods have been implemented and are readily available to be used. These three are the ϵCH- IϵCH- and RSS methods. The CHM module allows the model to handle constraints during the Pull phase.

4.4 Parameters

The parameters of the model are presented and explained in table 4.1. The table contains the relevant parameters for the experiments performed and detailed in chapter 5.

PPS Parameters	
T_{max}	The maximum number of function evaluations. Acts as the halting condition for the search.
T_c	The control generation where the relaxation of constraints is set to 0.
α	A parameter used to control the searching preference between feasible and infeasible regions during the pull phase.
τ	A parameter used to control the speed reducing the constraint relaxation in the case that the ratio of feasible individuals is less than α.

cp	A parameter used to control the speed of reducing the constraint relaxation in the case that the ratio of feasible solutions equals or is greater than α.
l	The number of generations to look at the rate of change between ideal and nadir points when checking if the framework should terminate the push phase.
Δ	A small number used to avoid division by 0 when calculating the change in ideal and nadir points.
ϵ	A parameter used to change from the push phase to the next. If the change in ideal and nadir points is less than ϵ, the push phase is ended.
H	The parameter defining the distribution of weights. The weights are generated according to the original work in [Zhang and Li, 2007] with a slight adjustment described in section 4.2.1.
T	The neighbourhood size.
CR	The crossover rate which control the number of parameter values copied from the mutant vector during crossover.
F	A parameter used to scale the differential variation during mutation.
δ	a number defining the the probability of selecting the closest t subproblems, or all subproblems, to evaluate the inclusion of offspring individuals.
nr	The maximum number of individuals being replaced by a child.
di	The distribution index for the polynomial mutation operator defined in Deb [2000].
pm	The mutation probability.
BiS	
κ	The selection probability of feasible individuals of the resulting population after BiS. With $\kappa = 0.3$, the population will aim to have a feasibility ratio of 30% after BiS.
γ	The minimum distance to dissolve pairing between feasible and infeasible individuals. When the distance between a pair of individuals reach the value of γ, they are excluded from further BiS.
RSS	
$FESc$	The number of function evaluations performed before δ_{out} is set to 0 and δ_{in} is ignored. Similar to T_c in PPS used to set the constraint relaxation to 0.
$numACD$	The number of individuals used for ACD. For each individuals, the distance to each constraint border is checked.
Val	The value used to identify a constraint as active or not. Usually set to a small number. If the distance to the constraint border is larger than Val then the constraint is deemed inactive.
Z	The parameter used to control the speed of shrinking the boundary from the infeasible region side. A larger Z leads to faster shrinking.

Table 4.1: Model Parameters.

Chapter 5

Experiments and Results

This chapter presents the experiments conducted to evaluate the proposed model. Testing has been divided into two parts: preliminary and experimental. First, preliminary testing lay the basis for the experimental phase. The experimental testing aims to answer the research questions defined in section 1.2. An experimental plan presenting the tests conducted and which questions they aim to answer have been created. Finally, the results are presented.

5.1 Preliminary Testing

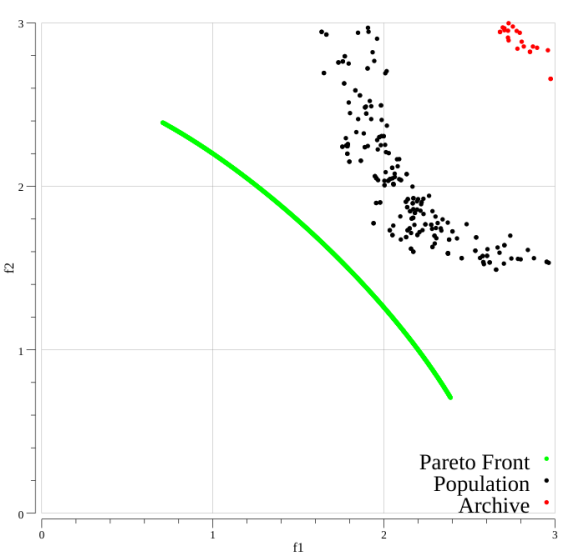
The preliminary testing was conducted both during and after the development of the proposed model. During the development of the model, tests were conducted to aid with design decisions. At the end of the preliminary testing, a parameter sweep was conducted to identify suitable parameters for the experimental phase.

5.1.1 Initial Testing

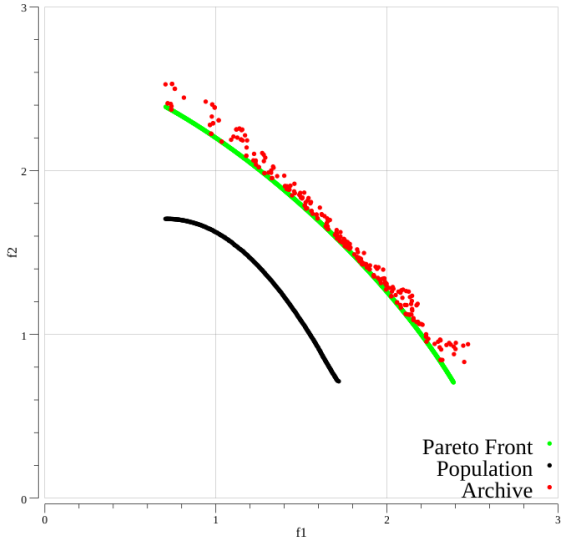
The first step of the preliminary testing was to replicate the work of Fan et al. [2019b]. Implementing the original framework and test suite, LIR (section 2.7), would ensure that PPS was correctly implemented and ready to be further developed.

Videos showing the search were generated to aid in evaluating the behaviour of the PPS framework and possible improvements. This contibuted in analysing evolution and the movement of the population in the original work by Fan et al. [2019b].

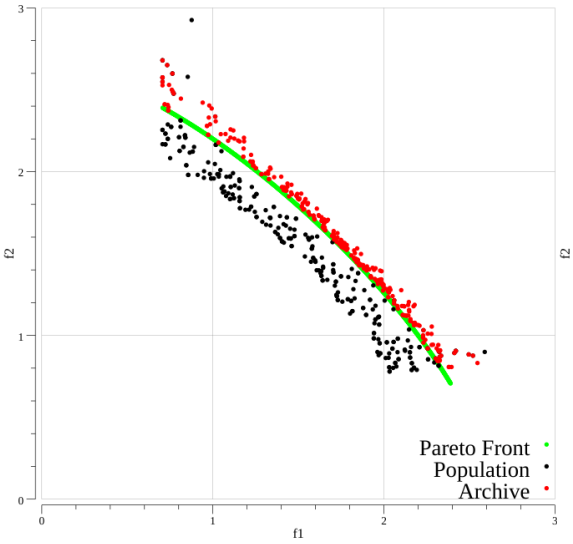
Figure 5.1 depicts four frames from a video of PPS solving LIR8. The plots consist of points visualising the population, archive and the PF. The population contains all the individuals in the current generation. The archive is a set of the most optimal feasible individuals identified during the run of the algorithm. This archive is returned as the set of solutions found, at the end of the run. The constrained PF is visualised as a green line in the objective space. In figure 5.1a the framework is still in the push phase, and constraints are ignored. For the visualised problem, the constrained PF is defined by the boundary between feasible and infeasible space. The unconstrained PF exists within the constrained space and closer to the origin than the constrained PF. In figure 5.1b, the population has passed



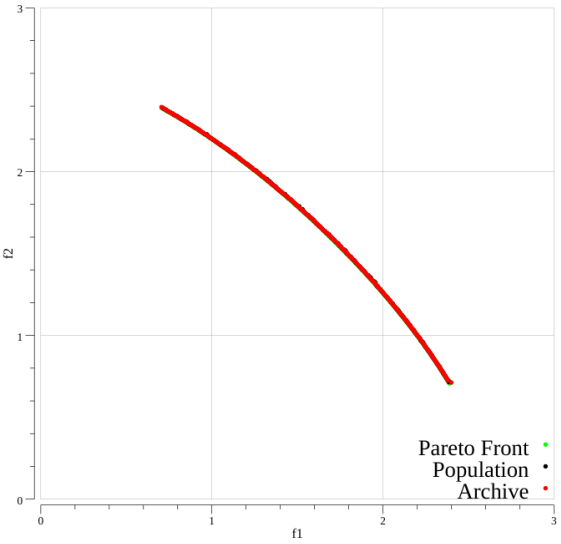
(a) Push phase: generation 40.



(b) Last generation of push phase: generation 192.



(c) Pull phase: generation 197.



(d) Final generation: generation 999.

Figure 5.1: PPS finding feasible individuals during the push phase on LIR8.

the constrained PF, and approximated the unconstrained PF by the MOEA/D optimiser exclusively evaluating objectives and ignoring constraints. In addition, the framework has found many feasible solutions during the push phase close to the constrained PF, which are saved in the archive. These are highlighted as red circles in the figure. Figure 5.1c shows the framework in the pull phase edging closer to the constrained PF. Simultaneously, the archive is updated as more feasible individuals dominating previous members of the archive are discovered. At the same time, the archive is updated as more feasible individuals that dominate previous members of the archive are discovered. Finally, figure 5.1d shows the final generation where the constrained PF has been approximated.

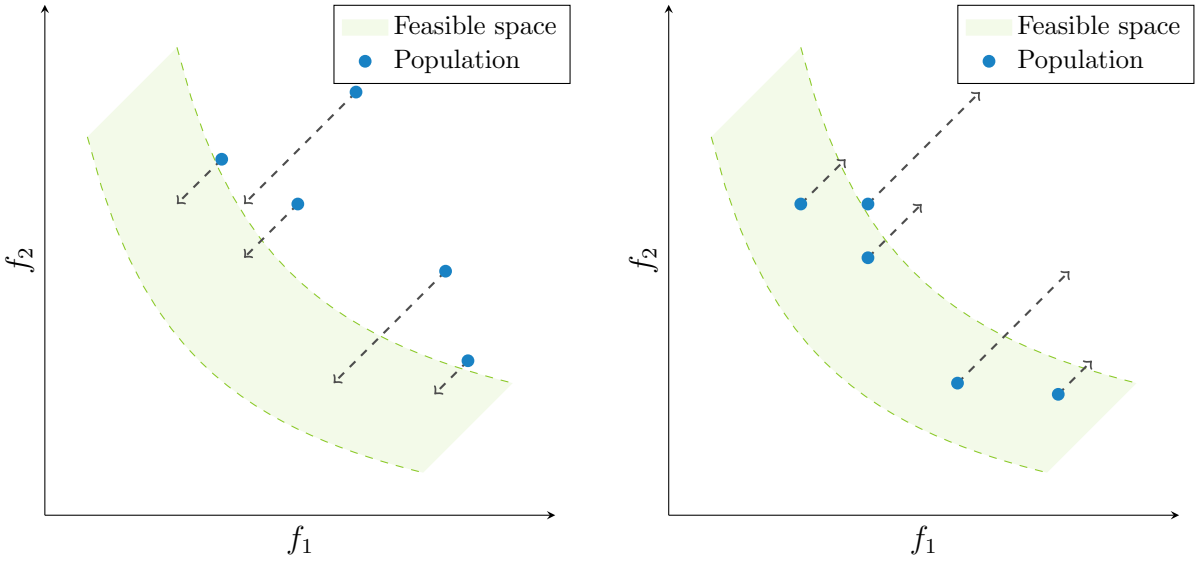
Figure 5.1 shows that the framework is able to locate feasible individuals close the the unconstrained PF during the push stage. Running similar tests on the other

LIR benchmarks showed that this was the case for problems with moderate to large feasible regions. For problems with small feasible regions, it was more uncommon due to the low probability of crossing these regions when ignoring constraints. However, the use of this knowledge is not utilised by the original framework to guide the search, as it relies solely on the CHM to guide the population in the right direction during the pull stage.

5.1.1.1 Alternating Between Binary Search and Improved ϵ-Constraint-Handling

During the pull phase, PPS uses IϵCH to handle constraints [Fan et al., 2019b]. IϵCH raises the allowed constraint violation of the population whenever the ratio of feasible points in the population, $\mathrm{FR_p}$, is above some threshold, see section 2.8.2. During the initial testing, it was hypothesised that the performance of PPS could be enhanced by combining BiS and IϵCH by alternating between the two methods. The idea was to perform BiS whenever the $\mathrm{FR_p}$ was below a certain threshold, and IϵCH whenever the $\mathrm{FR_p}$ was above a certain threshold.

Experiments showed that at the beginning of the pull phase, BiS would pull the population quickly towards the constrained PF as visualised in figure 5.2a. As the $\mathrm{FR_p}$ quickly rose, IϵCH would immediately pull the population away from the constrained PF again as visualised in figure 5.2b. This resulted again in BiS pulling the population back towards the constrained PF, creating a *tug of war* without the population being able to explore the boundary region between feasible and infeasible space. To avoid this tug of war, BiS was implemented as a new phase itself. This phase would be entered after the push phase and before the pull phase to produce an improved initial state for the pull phase.



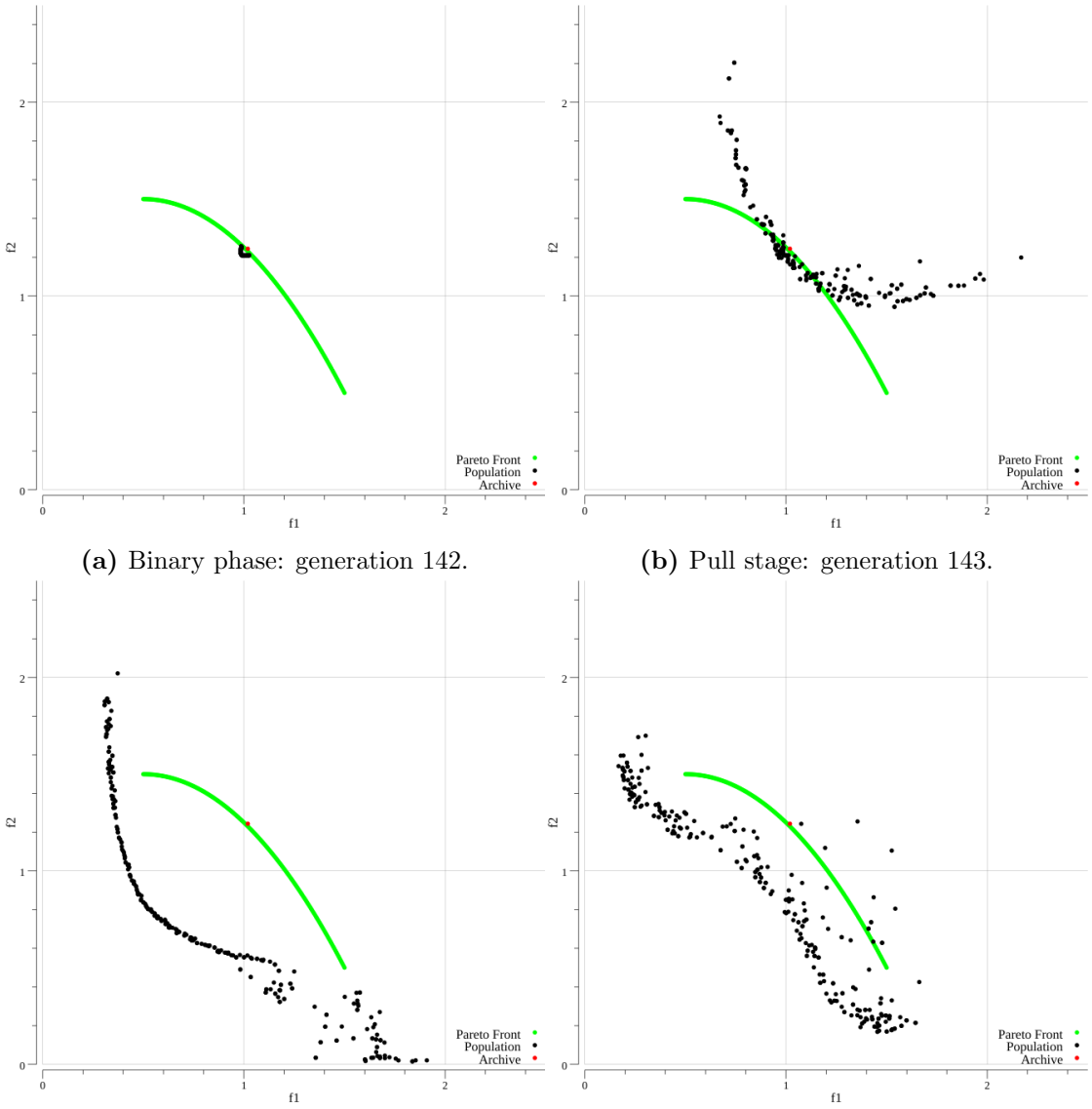
(a) Low $\mathrm{FR_p}$: BiS closes the gap to feasible space.

(b) High $\mathrm{FR_p}$: IϵCH pushes individuals out of feasible space.

Figure 5.2: Illustration of interaction between BiS and IϵCH.

5.1.1.2 Initial Constraint Threshold

In the original framework, the initial threshold for IεCH was set to be the maximum violation of the working population during the change from push to pull phase [Fan et al., 2019b]. However, this would yield complications when using the new BiS phase.



(a) Binary phase: generation 142.

(b) Pull stage: generation 143.

(c) Pull stage: generation 192.

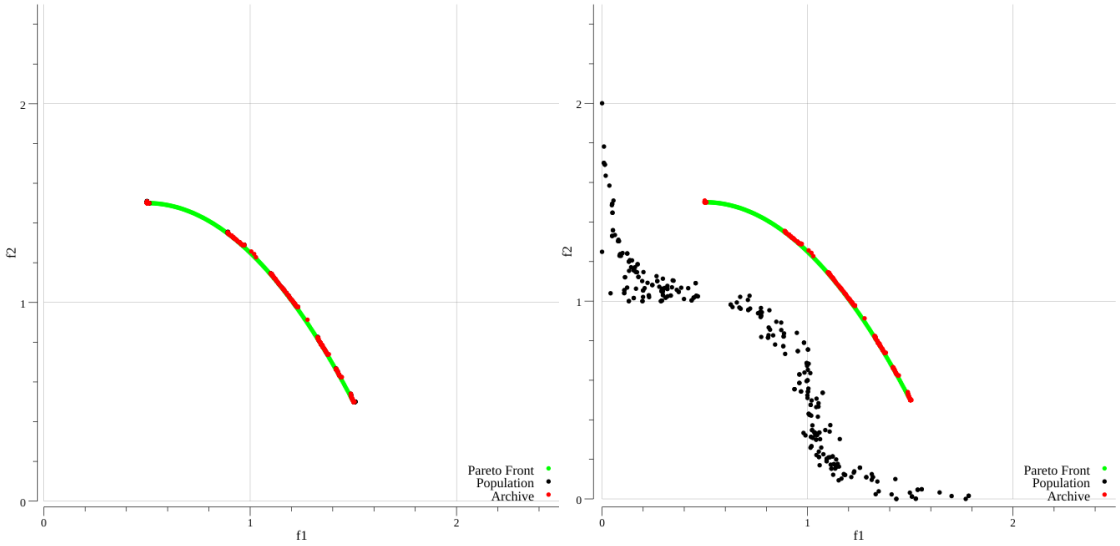
(d) Pull stage: generation 205.

Figure 5.3: Too large initial constraint violation on LIR1.

First, the population would immediately be pushed away from the constrained PF after entering the pull phase. This was due to the constraint relaxation being much larger than the constraint violation of the individuals after the BiS. Thus, the quick jump towards the constrained PF would be lost. This effect can be viewed in figure 5.3. In figure 5.3a the search is in the final generation of the BiS phase and has closed in the gap to the constrained PF. Up until generation 192, the population moves away from the constrained PF, as evident in figures 5.3b and 5.3c. At this

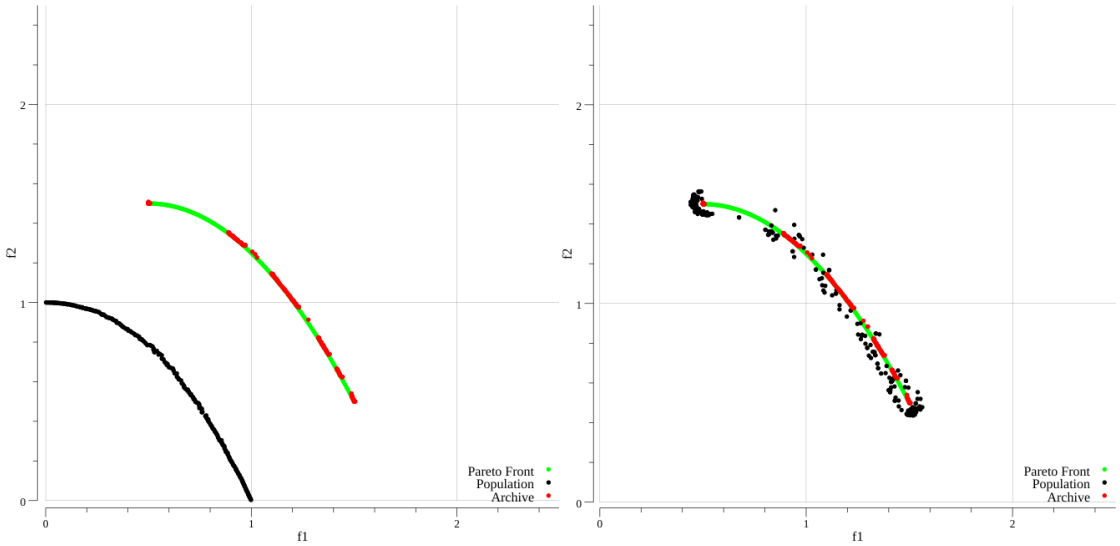
point, the constraint threshold used by IεCH is reached and the population starts moving towards the constrained PF again as shown in figure 5.3d.

Another disadvantage of using the original initialisation of the max constraint violation is when IεCH is relaxing the allowed threshold. This may be seen in figure 5.4. In figures 5.4a to 5.4c the population traverses back to the unconstrained PF due to the relaxation of the constraint threshold being too high. It is not until several generations later that the population has closed the gap to the constrained PF again, as seen in figure 5.4d. This phenomenon repeated itself several times during the evolutionary process. As a result, several generations are wasted traversing back and forth between the unconstrained and constrained PF.



(a) Pull phase: generation 433.

(b) Pull phase: generation 438.



(c) Pull phase: generation 468.

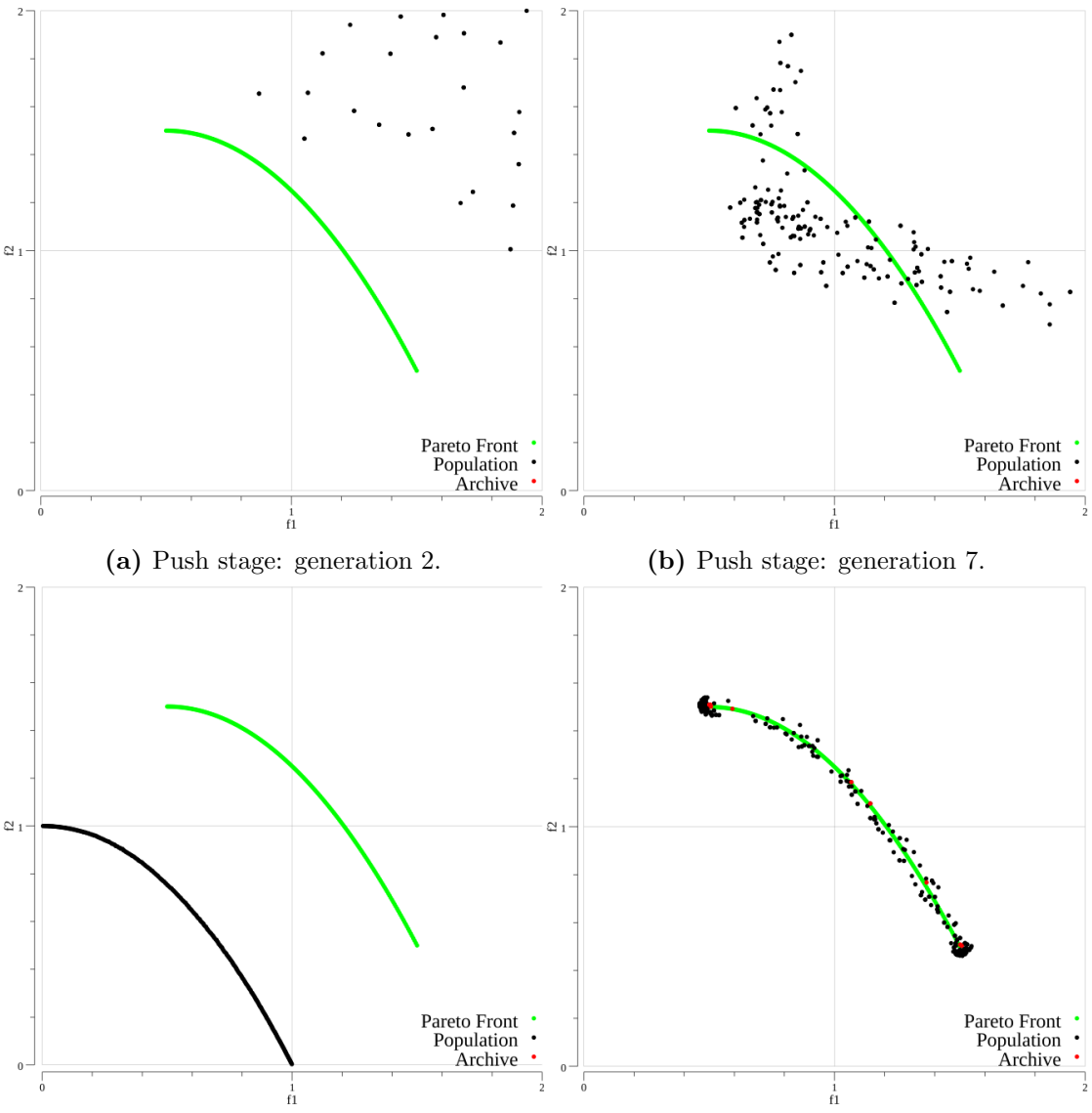
(d) Pull phase: generation 513.

Figure 5.4: Too large relaxation of constraint threshold on LIR1.

To circumvent this, instead of using the max constraint violation identified, the max constraint violation of the generation at the change of phases was selected. The experiments resulted in resetting the max violation after the BS. Thus, the initial

allowed constraint violation fit better to the current population after the switch from binary phase to pull phase.

5.1.1.3 Archive Size



(a) Push stage: generation 2.

(b) Push stage: generation 7.

(c) Push stage: generation 110.

(d) Pull stage: generation 213.

Figure 5.5: PPS not finding feasible individuals during the push phase on LIR1.

For problems with small feasible regions, PPS is either unable to or finds few feasible individuals during the push stage. This can be seen in figure 5.5. Figures 5.5a to 5.5c show the population during the push phase. As it moves past the constrained PF in figure 5.5b no feasible individuals are found. This is evident as no red circles are plotted. Due to no feasible individuals being found, BiS is never performed and the framework enters the pull phase immediately after the push phase. It is not until several generations into the pull stage that the population starts to become feasible, as seen in figure 5.5d.

These results show that the proposed BiS method may not be applicable for all types of problems. This is due to the lack of feasible individuals found during the push phase. Thus, BiS has no set of feasible individuals to form pairs with and the binary phase is skipped.

The fact that the framework is not guaranteed to find feasible individuals during the push phase raised a question regarding the size of the archive. If the framework finds few feasible individuals during the push phase, the archive will be small. Thus, the population may cluster to a small region of the search space when using BiS. This could hinder the population in covering the whole constrained PF.

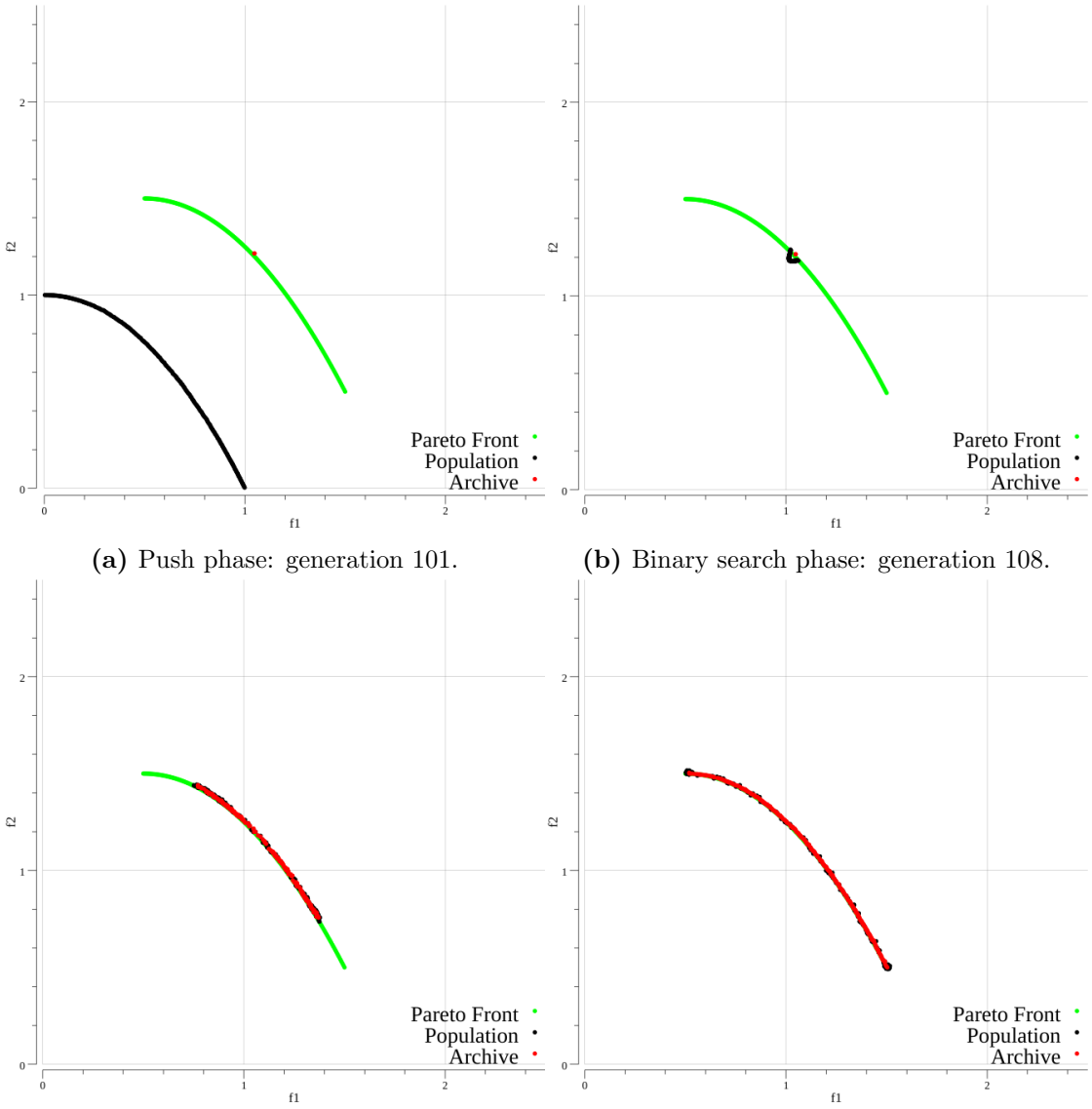
	PPS-IE		BiS-IE	
	Mean	**SD**	**Mean**	**SD**
LIR1	7.76e-03	1.69e-03	5.20e-03	2.95e-03
LIR2	4.85e-03	1.11e-03	2.36e-03	6.55e-05

Table 5.1: Comparison of IGD between PPS-IE and BiS-IE on problems with small feasible regions.

Table 5.1 shows the result of four runs of PPS-IE and four runs of BiS-IE on problems LIR1 and LIR2. For each run of BiS-IE, BiS was performed due to finding one feasible individual during the push phase. The binary phase has thus never been skipped due to not finding any feasible individuals during the push phase. Note that the number of runs is small. However, it still shows that the applicability of BiS may not be reliant on the size of the archive. This contradicts the initial hypothesis of having few feasible individuals to pair with during BiS would reduce the performance of the algorithm.

Figure 5.6 shows one of the runs BiS-IE on LIR1 used to generate the results in table 5.1. Figure 5.6a shows the population the last generation before entering the binary phase. The population has spread itself out over the unconstrained PF, and a single feasible individual has been found during the push stage. This is the only individual in the archive, denoted by the red circle. As shown in figure 5.6b, after only seven generations in the binary phase, the population has crossed the infeasible region blocking the way to the constrained PF. The population is now clustered around the single feasible point in the archive. Nonetheless, the BiS-IE is still able to produce competitive results even though the population had a low diversity when entering the pull phase. This can be seen in figures 5.6c and 5.6d.

In some cases PBPS with a small archive would outperform PPS as is evident in table 5.1. This is due to the framework using fewer generations moving through infeasible regions to reach the boundary. This can be seen when comparing figure 5.5 and figure 5.6. As shown in figures 5.5c and 5.5d, the population takes around one hundred generations to reach the constrained PF. figure 5.6a show that the binary phase only requires seven generations to traverse the same distance. Taking a leap towards feasible regions as shown in figures 5.6a and 5.6b allows for a longer search along the boundary between feasible and infeasible regions. Thus, more generations are spent searching promising regions rather than traversing towards them. Based on these results, it was decided to not restrict the use of BiS based on the size of the archive. Even if only a single feasible individual is found during the push phase,



(a) Push phase: generation 101.

(b) Binary search phase: generation 108.

(c) Pull phase: generation 142.

(d) Pull phase: generation 359.

Figure 5.6: Reduced diversity after binary phase on LIR1.

the binary phase will be entered and BiS be performed.

5.1.2 Parameter Sweeping

Sections 5.1.3 to 5.1.5 present tests to identify parameters for the two BS methods BiS and RSS. All problems containing two objectives in the LIR test suite have been selected for the sweep. Additionally, the following MW problems have also been used: MW5, MW6, MW10, MW13, MW9 and MW11 from Ma and Wang [2019].

In section 5.1.3, only the MW problems are used. The LIR problems are selected for comparison to the original work of Fan et al. [2019b]. The MW problems are selected to evaluate the original and the proposed models on untested problems. All problems were run 8 times with a selected parameter combination, and the FR_c,

IGD and HV were calculated. The metrics calculated are described in sections 2.8.1, 2.8.3 and 2.8.4 respectively.

5.1.3 Parameter Sweeping for MW Problems

The parameter sweep is split into two phases:

- **Optimiser values**
  The CR and F values for MOEA/D, the crossover rate and differential weight respectively, see sections 2.3.1.2 and 2.3.1.3.

- **Constraint handling values**
  The τ, α and cp values for IϵCH, mentioned in equation (3.4).

5.1.3.1 Optimiser Values

During initial testing, the parameters from the original work proved to be incapable of solving the objective functions for the introduced MW problems. The MOEA/D optimiser implemented herein utilise DE crossover (see section 2.3.1) to produce new offspring. When an offspring is created, a mutation operator is used on the newly generated individual to slightly modify the genotype of the individual. The mutation operator aims to mutate offspring within the vicinity of the generated individual [Deb, 2000]. Based on this information it was concluded that changing the parameters of CR and F would have the greatest impact. The values for CR and F was tested in the range $CR, F \in [0, 1]$.

Table 5.2 shows the results of the parameter sweep. The table contains the best results for each problem in the MW test suite. The parameter combinations have been selected by first looking at the highest values of $\mathrm{FR_c}$. Then, IGD has been selected to break ties between $\mathrm{FR_c}$ values. Finally, HV has been used to break potential ties in IGD.

The IGD and HV values are the mean calculated from the feasible runs. The reference point used to calculate the HV metric is set to the nadir point of the PF multiplied by 1.1. The value of 1.1 is gathered from the CEC2020 competitions evaluation criteria [Wang et al., 2020].

In the case where $\mathrm{FR_c}$ was 0% the IGD and HV were not calculated. Also, for runs with an archive where no points dominated the reference point, HV was not calculated.

No set of CR and F gave the best result for multiple problems. This is evident in table 5.2 as the CR and F values are different for each problem. The only exceptions are MW13 and MW11 which both have $CR = 0.3$, and MW6 and MW11 where both have $F = 0.3$. However, some sets of parameters gave a $\mathrm{FR_c}$ of 100% for all problems showing that they were consistently able to solve the CMOPs.

PPS was unable to consistently solve MW10 with any combination of parameters. This is evident in table 5.2, where the $\mathrm{FR_c}$ is 75%.

Table 5.3 shows the set of parameters where a $\mathrm{FR_c}$ of 100% was achieved for all CMOPs, except MW10. Any of these parameter sets should be able to solve the problems consistently. The parameter set $CR = 0.9$ and $F = 1.0$ performed the

Problem	CR	F	FR	Mean IGD	Mean HV
MW5	0.7	0.8	100%	8.23e-03	0.38
MW6	0.2	0.3	100%	3.34e-01	0.23
MW9	0.4	0.9	100%	7.34e-03	0.48
MW10	0.1	0.6	75%	3.59e-01	0.32
MW11	0.3	0.3	100%	7.38e-03	2.28
MW13	0.3	1.0	100%	1.63e-01	2.78

Table 5.2: Best parameters for each problem.

best in both mean IGD and mean HV across all problems. Therefore, $CR = 0.9$ and $F = 1.0$ will be used for comparisons and testing for the BS tests on all MW problems.

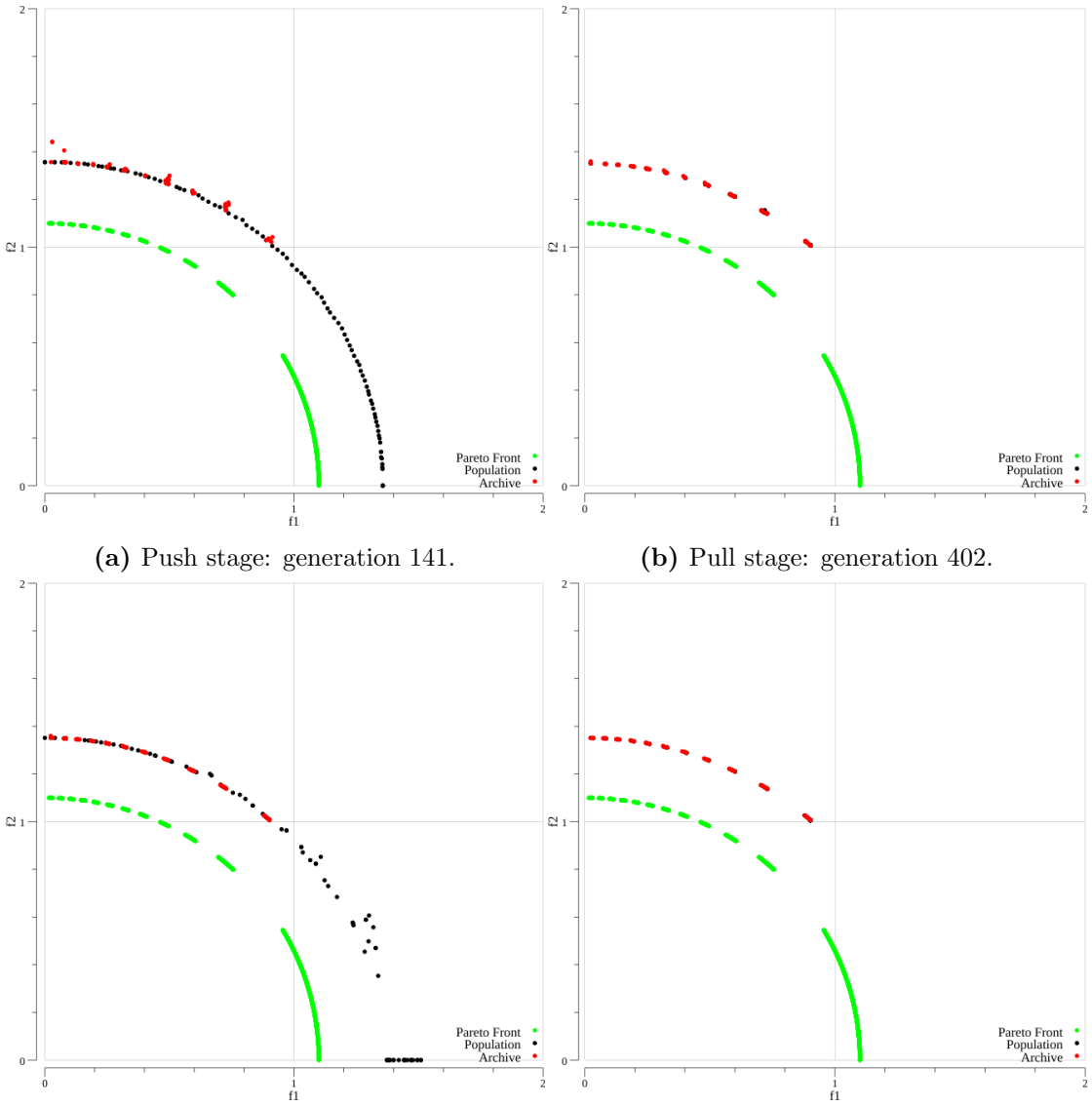
Additional fine tuning of parameters, such as the neighbourhood size of MOEA/D, T, (see section 2.4.1) could be tested further. However, prior work by Fan et al. [2019b] reported no significant difference when tuning this parameter. Other parameters for the mutation operator could also be tested for increased performance. However, as the main contribution of this thesis is the use of BS, the work by Fan et al. [2019b] was used to select values for the remaining parameters. The goal of the preliminary test is not to find the optimal values for all parameters, but a set of parameters that is able to solve the problem and thus allows further testing with BS and RSS. Therefore, no further sweeping of parameters unrelated to constraint handling or BS is performed.

CR	F	Mean IGD	Mean HV
0.1	0.7	3.08e-01	1.08
0.1	1.0	2.18e-01	0.70
0.2	0.7	2.57e-01	1.32
0.2	0.8	1.98e-01	1.19
0.2	0.9	2.13e-01	1.13
0.2	1.0	1.82e-01	1.03
0.3	0.8	1.99e-01	1.18
0.4	0.8	1.85e-01	1.01
0.4	0.9	1.52e-01	1.45
0.4	1.0	1.65e-01	1.46
0.5	0.9	2.16e-01	1.15
0.6	0.8	1.82e-01	1.41
0.6	0.9	1.96e-01	1.21
0.7	0.9	2.05e-01	1.20
0.7	1.0	1.73e-01	1.45
0.9	**1.0**	**1.49e-01**	**1.46**

Table 5.3: Parameters with 100% FR_c over all problems.

5.1.3.2 Constraint Values

Section 5.1.3.1 and table 5.2 show poor IGD results for MW6, MW10 and MW13, caused by the solution archive being far from the PF.



(a) Push stage: generation 141.

(b) Pull stage: generation 402.

(c) Pull stage: generation 765.

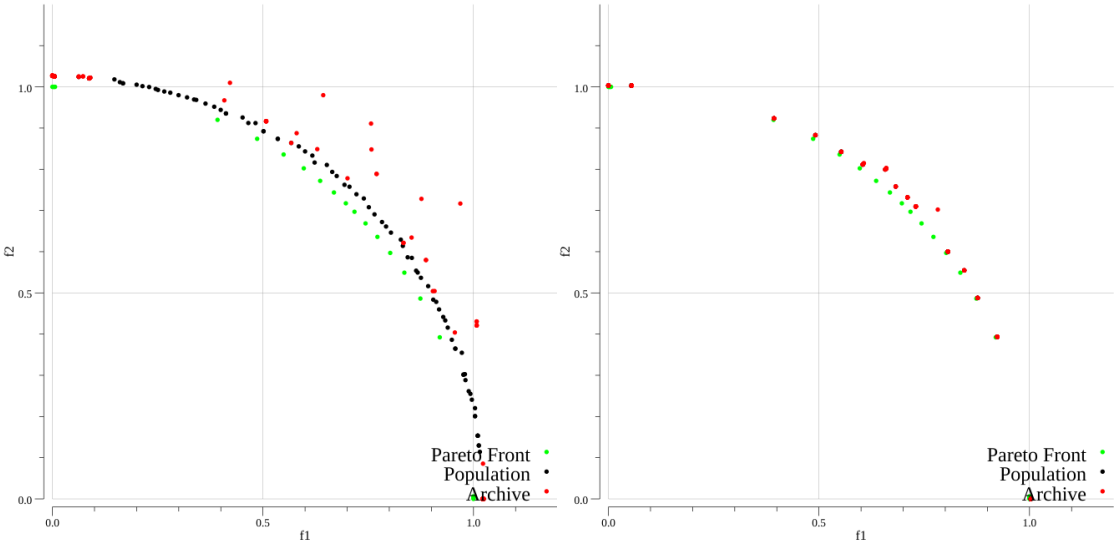
(d) Final generation: generation 1000.

Figure 5.7: PPS unable to properly solve MW6.

Figure 5.7 shows a run of PPS trying to solve MW6. In figure 5.7a the final generation of the push phase is shown. At this generation, the population has stagnated before the pull phase is initialised. The MOEA/D optimiser was unable to locate any part of the constrained PF during the push phase. In stead the population has converged to a local optima surrounding the true PF. As seen in figure 5.7b, the pull phase is initialised believing the unconstrained PF has been found during the push phase. The population is evolved with constraints considered and approximates similar disjoint segments that constitute the constrained PF. However, the population is unable to get any closer to the constrained PF. Figures 5.7c and 5.7d shows relaxation and then the reduction of the constraint threshold performed by IϵCH. As seen, the population is not evolved further towards the PF, but along the shape of the local optima.

PPS was able to find good solutions for the MW5, MW9 and MW11 CMOPs

using the original constraint handling parameters of $\alpha = 0.95, \tau = 0.1, cp = 2.0$ from Fan et al. [2019b]. As a comparison to the earlier mentioned difficulties of finding the unconstrained PF of MW6 - figure 5.8a shows PPS finding the unconstrained PF of MW5 during the push phase. Figure 5.8b shows the final generation where all the feasible disjoint sections on the PF have been discovered.



(a) Push stage: generation 500. **(b)** Pull stage: generation 1000.

Figure 5.8: PPS finding the unconstrained PF of MW5.

The original work by Fan et al. [2019b] assumes that when switching phases from push to pull the population has reached the unconstrained PF and the change in nadir and ideal point have been brought to a halt (see section 3.2). If this is not the case, then the introduction of a CHM in the evolutionary process will have little effect on the ability of the framework to locate the constrained PF.

From this testing it was concluded that the problem in approximating the unconstrained PF of MW6, MW10, MW13 was caused by the selected optimiser not being able to solve the objective functions. Inability to approximate the unconstrained PF in turn results in inability to approximate the constrained PF.

Analysing which MOEA is best suited to solve the underlying MOP for these problems is outside the scope of this thesis. Tuning the parameters of the CHM would not improve the ability of the framework to approximate the unconstrained or constrained PF. Therefore, additional experiments to find parameters for IϵCH were not conducted and the values from [Fan et al., 2019b] were used.

5.1.4 Parameter Sweep for Binary Search Phase

Parameters for BiS are described in section 4.4. κ and γ were tested in the ranges $[0.01, 0.2]$ and $[0.1, 0.5]$ respectively.

The LIR problems ran using the original values $CR = 1.0$ and $F = 0.5$ [Fan et al., 2019b]. The MW problems ran with the values $CR = 0.9$ and $F = 1.0$ discovered in section 5.1.3.1. The best runs for each problem are presented in table 5.4.

Problem	κ	γ	FR	IGD	HV
LIR1	0.5	0.02	100%	5.80e-03	0.65
LIR2	0.5	0.03	100%	3.91e-03	0.98
LIR3	0.4	0.10	100%	5.26e-03	0.54
LIR4	0.1	0.05	100%	3.27e-03	0.78
LIR5	0.5	0.04	100%	1.54e-03	1.03
LIR6	0.5	0.05	100%	2.13e-03	0.70
LIR7	0.1	0.01	100%	2.81e-03	2.04
LIR8	0.5	0.01	100%	2.68e-03	2.04
LIR9	0.1	0.20	100%	1.90e-03	2.78
LIR10	0.5	0.03	100%	1.96e-03	2.56
LIR11	0.5	0.20	100%	3.06e-03	3.44
LIR12	0.3	0.04	100%	3.35e-02	4.26
MW5	0.5	0.02	100%	4.31e-03	0.39
MW6	0.1	0.03	100%	5.96e-01	-
MW10	0.2	0.04	37.5%	5.93e-01	0.15
MW13	0.1	0.01	100%	2.05e-01	2.59
MW9	0.2	0.20	100%	1.21e-02	0.47
MW11	0.4	0.20	100%	7.84e-03	2.27

Table 5.4: Best binary sweep runs for all problems.

No set of parameters was able to achieve 100% FR_c on MW10. Parameters that were able to achieve 100% FR_c on all problems are gathered and presented in table 5.5. These parameters proved to perform sufficiently for all tested problems. The mean IGD and HV are calculated from the results of all problems, excluding MW10. No parameter set was superior in both IGD and HV. However $\kappa = 0.1, \gamma = 0.03$ performed best in IGD and second best in HV. $\kappa = 0.1, \gamma = 0.01$ performed best in HV and second best in IGD. As a goal of BiS in this thesis is to approximate the boundary and bring the population within the proximity of the PF, the parameters $\kappa = 0.1$ and $\gamma = 0.03$ are selected for further experiments.

κ	γ	Mean IGD	Mean HV
0.1	0.01	6.94e-02	**2.03**
0.2	0.01	9.95e-02	1.81
0.4	0.01	7.04e-02	1.92
0.1	0.03	**6.40e-02**	1.93
0.1	0.05	9.45e-02	1.83
0.2	0.05	6.99e-02	1.92
0.1	0.10	1.01e-01	1.82
0.4	0.10	7.66e-02	1.92
0.5	0.10	7.31e-02	1.92

Table 5.5: Mean IGD and Mean HV for all successful parameter sets.

5.1.5 Parameter Sweep for Reduced Search Space Operator

Parameters for the RSS operator are described in section 4.4. The parameters tested are highlighted in table 5.6.

Parameter	Min Value	Max Value	Step Size
$numACD$	10	50	10
Val	0.05	0.15	0.01
Z	0.5	3.0	0.5

Table 5.6: RSS parameter sweep overview.

In addition to the metric mentioned in section 5.1.2, for each combination of parameter values, the success rate of ACD was logged. This was to get insight into the degree of how RSS affected the performance using the different combination of parameter values. In the case of unsuccessful ACD, RSS is not used during the search as no constraints are viewed as active.

Table 5.7 highlights the highest achieved ACD rate for each value of Val over all problems. Intuitively, the larger values achieve better results, as larger values of Val allows individuals to be further away from constraint boundaries and still view them as active. Also, a larger number of individuals being used for ACD results increases the probability of one of them being close to a boundary. Note that no combination of values achieve 100% success rate. This is due to LIR1 and LIR2. For these two problems, the constraint boundary is simply too far away. Out of all runs, only one single run managed to identify an active constraint when solving LIR1. The distance from the population to the constraint boundary is simply larger than the value range tested for Val.

The aim is to analyse how RSS affects the performance of PPS. As the effect of RSS is dependent on it actually identifying some constraint as active it was decided that further experiments would use values 0.15 and 50 for Val and $NumACD$ respectively. For testing LIR1 and 2, $Val = 0.25$ was selected based on a manual testing showing that this was approximately the distance to the constraint boundaries at the end of the push phase.

Val	NumACD	ACD successful
0.05	50	71.83%
0.06	50	73.24%
0.07	50	75.23%
0.08	50	75.23%
0.09	40	86.74%
0.10	50	87.68%
0.11	50	88.03%
0.12	50	88.26%
0.13	50	88.50%
0.14	50	88.85%
0.15	50	88.95%

Table 5.7: Highest rate of successful ACD for different Vals.

A parameter sweep was conducted for all combinations of values for the three parameters $numACD$, Val and Z. However, after selecting values for $numACD$ and Val, the amount of data needed to be analysed to select a value for Z was greatly reduced. Table 5.8 summarises the results from this analysis. For each Z value the table highlights the mean IGD HV, in addition to the standard deviation of both metrics. The results are based on all eight runs for all problems. No value performed strictly better in all categories. Due to the overall well performance, $Z = 2$ was selected as the parameter value for later experiments.

Z	Mean IGD	STDEV IGD	Mean HV	STDEV HV
0.5	1.23e-01	2.20e-01	1.65	1.25
1.0	8.71e-02	1.84e-01	**1.82**	**1.21**
1.5	1.05e-01	**1.63e-01**	1.56	1.26
2.0	**7.98e-02**	1.66e-01	**1.82**	**1.21**
2.5	1.36e-01	2.11e-01	1.56	1.26
3.0	1.06e-01	1.88e-01	**1.82**	1.23

Table 5.8: The effect of Z-values on the performance of the framework.

5.2 Experimental Plan

In order to answer the RQs presented in section 1.2 an experimental plan was developed and executed. The plan is split into four phases, each aiming to answer one of the RQs. Each phase is further elaborated below.

5.2.1 Plan Phase T1

RQ 1: What landscape information extracted during the evolutionary search can benefit PPS to increase convergence to and coverage of the constrained PF?

Hypothesis: It is expected that PBPS will result in better or on par convergence and coverage with PPS. Also, it is expected that PPS-RSS will perform on par with PPS-E.

Test Overview

T1.1 Testing convergence and coverage for PPS.

T1.2 Testing convergence and coverage for PBPS.

T1.3 Testing convergence and coverage for PPS-RSS.

5.2.2 Plan Phase T2

RQ 2: How do different problem characteristics affect the performance of BS?

Hypothesis: It is expected that PPS-RSS will perform poorly on problems with disconnected PFs. It is also expected that PBPS will perform better on problems with large infeasible regions than both PPS and PPS-RSS.

Test Overview

T2.1 Testing PPS on convex problems.

T2.2 Testing PPS on concave problems.

T2.3 Testing PPS on problems with connected PFs.

T2.4 Testing PPS on problems with disconnected PFs.

T2.5 Testing PPS on problems with small feasible space.

T2.6 Testing PBPS on convex problems.

T2.7 Testing PBPS on concave problems.

T2.8 Testing PBPS on problems with connected PFs.

T2.9 Testing PBPS on problems with disconnected PFs.

T2.10 Testing PBPS on problems with small feasible space.

T2.11 Testing PPS-RSS on convex problems.

T2.12 Testing PPS-RSS on concave problems.

T2.13 Testing PPS-RSS on problems with connected PFs.

T2.14 Testing PPS-RSS on problems with disconnected PFs.

T2.15 Testing PPS-RSS on problems with small feasible space.

5.2.3 Plan Phase T3

Phase T3 focuses on *how* the population moves after the push phase, while the constrained PF is approximated. Herein the *traversal* of the population will be evaluated. The term is used herein to describe the movement of the population in the objective space and will be used during the discussion in section 5.4.2.

RQ 3: How does the use of BS affect the traversal to the unconstrained PF through infeasible space?

Hypothesis: It is expected that PPS and PPS-RSS will achieve similar results. In addition, it is expected that PBPS will achieve better results than both PPS and PPS-RSS.

Test Overview

T3.1 Testing traversal of PPS.

T3.2 Testing traversal of PBPS.

T3.3 Testing traversal of PPS-RSS.

5.2.4 Plan Phase T4

Phase T4 focuses on *where* the population moves after the push phase, while the constrained PF is approximated. Herein the term *bias* will be used as a description of focusing the search in a specific area, or a preference towards or inability to escape local minima.

RQ 4: How can BS introduce a bias towards certain areas of the objective space?

Hypothesis: It is expected that PBPS will produce a less diverse population. This is due to the focus on small areas of the search space. Also, it is expected that PPS-RSS will produce a similarly diverse population as PPS. It is expected that both PBPS and PPS-RSS will show similar ability to avoid local minima as PPS.

When looking for a potential bias towards certain areas of the objective space CD and IGD of the population is used. This is due to the assumption that a low CD implies that the population is clustered around the same area, while a high CD means that the individuals are spread more out. Whether or not a high or low CD is beneficial will depend on the problem, and will be discussed if necessary. Also, the IGD may indicate if the population is prone to finding local minima instead of approaching the constrained PF.

Test Overview

T4.1 Test diversity in population when using PPS.

T4.2 Test diversity in population when using PBPS.

T4.3 Test diversity of population when using PPS-RSS.

5.3 Experimental Setup

The following section describes the experimental setup for the different phases of the experimental testing. First, a general setup for the whole experimental testing phase is given. Then, experimental setup specific for each phase is elaborated.

Table 5.9 highlights all benchmarks used during the experimental testing. Figure 5.9 shows the LIR problems [Fan et al., 2019a]. For LIR1-4, the feasible space is highlighted by a green overlay. For the rest of the problems, feasible space is white. Note however that the legend for LIR5 is wrong: the grey areas are infeasible space, not feasible. The unconstrained PF is highlighted by a red line in all problems, and the constrained PF is identified by blue circles. Figure 5.10 shows the MW problems [Ma and Wang, 2019]. The feasible space is greyed out while infeasible space is white. Note, this is the opposite as in figure 5.9. The unconstrained PF is highlighted in blue, while the constrained PF is highlighted in red.

A specialised test suite has been created for each phase of the testing, testing different aspects of the models created. Each test suite is described in tables tables 5.13 to 5.15. For each benchmark, the original name is given. Further, relevant

Benchmark	Objective Functions	Constraints	Decision Variables	Search Space Range
LIR1	2	2	30	$0 \leq x \leq 1$
LIR2	2	2	30	$0 \leq x \leq 1$
LIR3	2	3	30	$0 \leq x \leq 1$
LIR4	2	3	30	$0 \leq x \leq 1$
LIR5	2	2	30	$0 \leq x \leq 1$
LIR6	2	2	30	$0 \leq x \leq 1$
LIR7	2	3	30	$0 \leq x \leq 1$
LIR8	2	3	30	$0 \leq x \leq 1$
LIR9	2	2	30	$0 \leq x \leq 1$
LIR10	2	2	30	$0 \leq x \leq 1$
LIR11	2	2	30	$0 \leq x \leq 1$
LIR12	2	2	30	$0 \leq x \leq 1$
MW5	2	3	25	$0 \leq x \leq 1$
MW6	2	3	25	$0 \leq x \leq 1.1$
MW10	2	1	25	$0 \leq x \leq 1$
MW13	2	2	25	$0 \leq x \leq 1.5$
MW9	2	1	25	$0 \leq x \leq 1$
MW11	2	4	25	$0 \leq x \leq \sqrt{2}$

Table 5.9: Benchmarks used for experimental testing.

characteristics are highlighted: **Shape** describes whether the unconstrained and constrained PF are convex (CV) or concave (CC). The values can be any combination of "CV" and "CC", separated by a "/". The first abbreviation denotes the shape of the unconstrained PF and the abbreviation after "/" denotes the shape of the constrained PF. **Region Size** shows the size of the feasible and infeasible regions respectively. The possible values are S, M, and L which stand for small, medium, and large. **Overlapping PFs** highlights to what degree the unconstrained and the constrained PF are overlapping. "N" signifies no overlap, "F" signifies a full overlap, and "P" signifies partial overlap. Finally, **Disconnected PF** highlights the benchmarks where the constrained PF consists of disconnected segments.

Table 5.10 shows the general parameter setting for each phase. A full list of parameters with descriptions can be found in section 4.4.

All phases and their corresponding test runs have been run on a computer with the specifications described in table 5.12. All the cores were used to run the experiments. Each benchmark has been run 30 times for each model.

When discussing the results, several tools will be used. T-tests will be performed to analyse if there is a significant difference in performance ($P < 0.05$). The results of the T-tests will be presented in tables containing only the problems where a significant difference is present. Boxplots will be used to visualise the mean performance either in regards to IGD or HV over the 30 runs for a problem. The boxplots will also highlight the Standard Deviation (SD) for the 30 runs. In addition, line chart will be used. The line charts may highlight IGD, HV, FR_p or CD. These values will be shown over a span of generations from either a single run or the median over

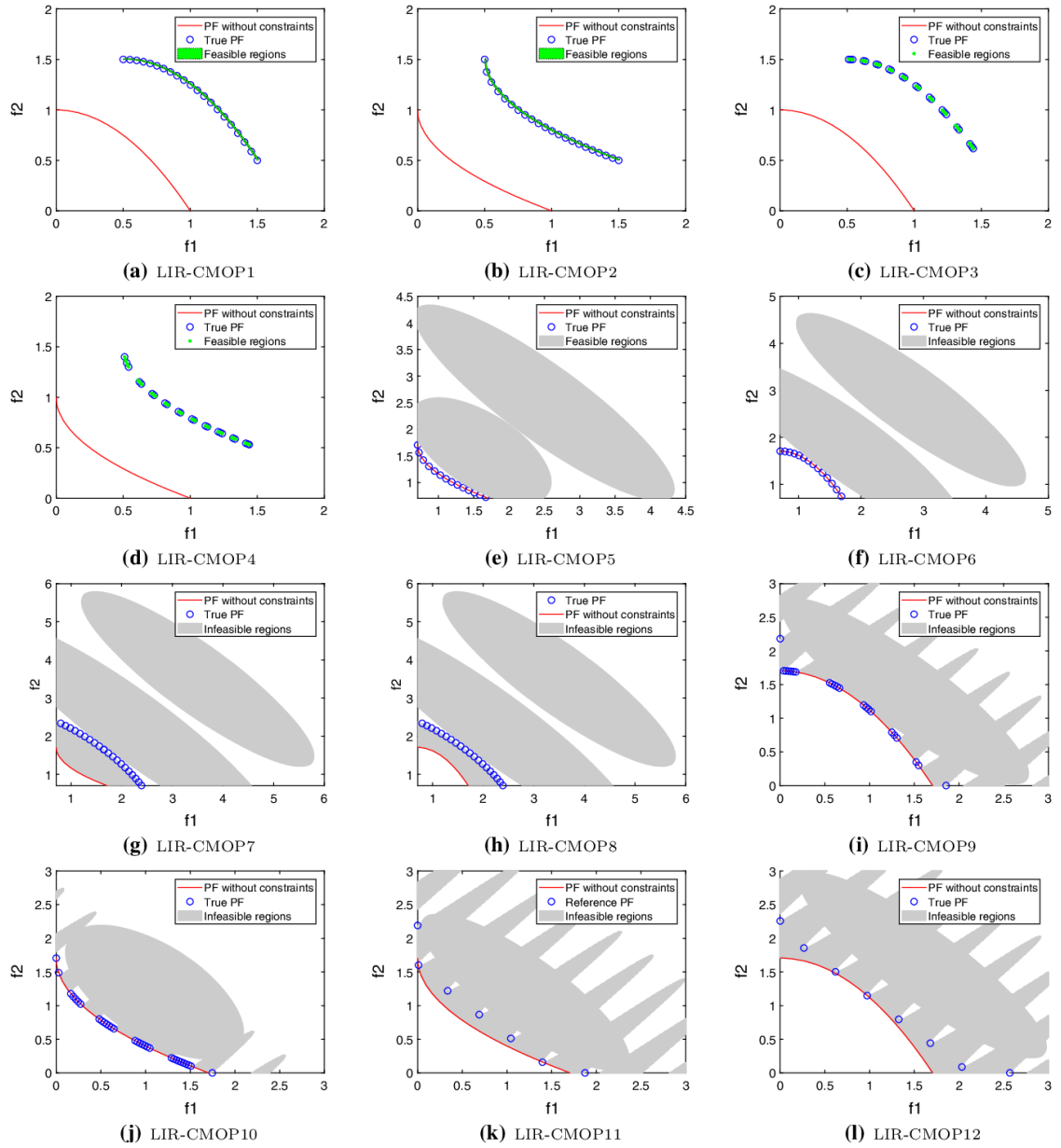
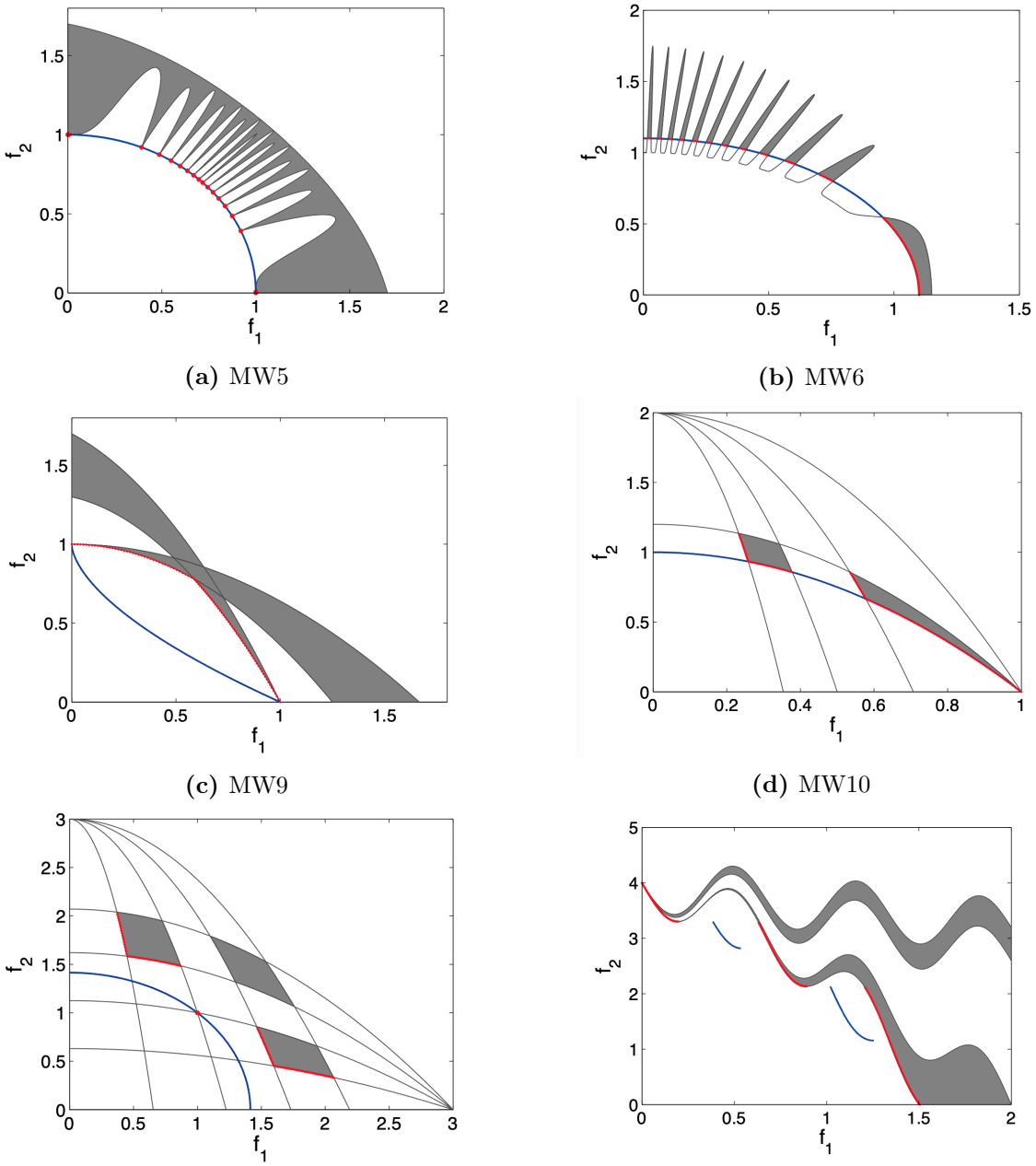


Figure 5.9: LIR problems used during experimental testing [Fan et al., 2019a].

several runs. Finally, line charts will also be used to look at the end performance for several runs as illustrated in figure 5.11. The red line represents the number of generations where BiS was performed for each of the 30 runs. This is used to indicate how many and which runs BiS was performed for the 30 runs. A spike in the red line indicates that BiS was performed for a run, while a flat line at 0 indicates that BiS was not performed. The blue line indicates the HV and the green line the IGD achieved for each run. Note that the y-axis does not represent any exact value as the point of these plots is to highlight the correlation between BiS and performance.



(a) MW5 **(b)** MW6

(c) MW9 **(d)** MW10

(e) MW11 **(f)** MW13

Figure 5.10: MW problems used during experimental testing [Ma and Wang, 2019].

5.3.1 Setup Phase T1

T1 aims to answer RQ1. In order to accurately answer this question, the results of the PPS and PPS with BS approaches will be compared. PPS has previously shown good results on the selected benchmarks. The reason for choosing the same problems is to see if the previous success can be further enhanced. Also, it should be easier to see if Boundary Search is not beneficial to PPS. Each model will be run 30 times on the problems described in table 5.13.

To test the convergence of the different models, the IGD metric is calculated, (see

PPS		
	LIR	**MW**
T_{max}	3e+05	1e+05
T_c	800	800
α	0.95	0.95
τ	0.1	0.1
cp	2.0	2.0
l	20	20
Δ	1e-05	1e-05
ϵ	1e-03	1e-03
H	301	101
T	30	30
CR	1.0	0.9
F	0.5	1.0
δ	0.9	0.9
nr	2	2
di	20	20
pm	$\frac{1}{30}$	$\frac{1}{25}$
BiS		
	LIR	**MW**
κ	0.1	0.1
γ	0.03	0.04
RSS		
	LIR	**MW**
$FESc$	$T_{max} \times 0.9$	$T_{max} \times 0.9$
$numACD$	50	50
Val	0.15	0.15
Z	2.0	2.0

Table 5.10: Common parameter setup.

RSS	
Val	0.26

Table 5.11: Parameters for LIR-CMOP1 and LIR-CMOP2.

Attribute	**Description/Value**
Model Name	Intel(R) Core(TM) i7-4770 CPU 3.40GHz
Architecture	x86_64
CPU Cores	8
Threads per Core	2
RAM	16GB

Table 5.12: Computer Specifications.

section 2.8.3). The IGD metric measures the distance between the true PF and the solution set produced by the model. A lower score implies better convergence than a higher score. The IGD will be calculated from the archive of feasible individuals

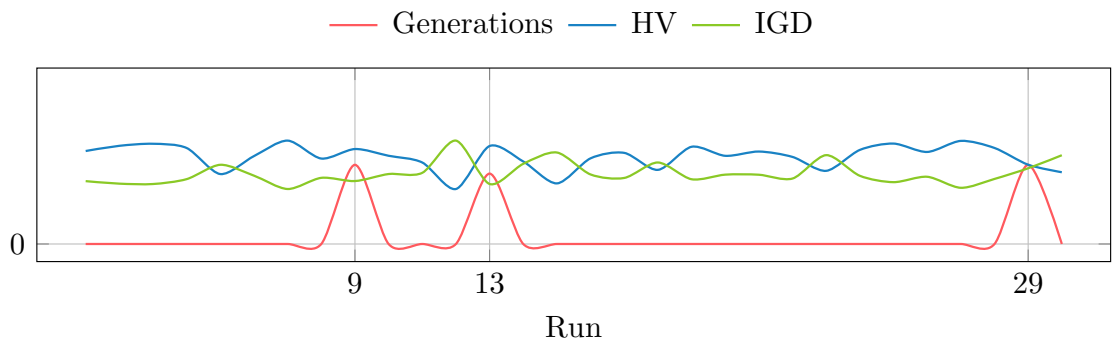


Figure 5.11: Example linechart comparing performance over several runs.

Benchmark	Shape	Region Size (Feasible/Infeasible)	Overlap	Disconnected
LIR1	CV/CV	S/L	N	-
LIR2	CC/CC	S/L	N	-
LIR3	CV/CV	S/L	N	X
LIR4	CC/CC	S/L	N	X
LIR5	CC/CC	L/M	F	-
LIR6	CV/CV	L/M	F	-
LIR7	CC/CV	M/M	N	-
LIR8	CV/CV	M/M	N	-
LIR9	CV/CV	M/L	P	X
LIR10	CC/CC	M/L	P	X
LIR11	CC/CC	M/L	P	X
LIR12	CV/CC	M/L	P	X

Table 5.13: T1 Test Suite.

produced from the model, (see section 3.2). Calculating the IGD from an empty archive is impossible. In the cases where no feasible individuals are discovered, the total number of runs will be reduced when calculating the mean value, as the infeasible runs would be undefined. Mean values will be compared and equal sample sizes is beneficial when performing the comparison. The consistency proved by the accomplishment of finding feasible individuals in every run is more desirable than high performance but low consistency. Thus the CMOP Feasibility Rate (FR_c) (see section 2.8.1) was given a high priority and a FR_c of 100% a criteria for inclusion in the results.

To test the coverage of the different models, the HV metric is calculated, (see section 2.8.4). The HV metric measures the volume created between the population and a reference point. The reference point for all calculations is the nadir point of the true PF multiplied by 1.1. Any multiplier value can be used when calculating HV, however this work follows the evaluation criteria defined in Wang et al. [2020]. A higher HV score implies better coverage than a low score. In addition to removing results not achieving 100% FR_c as mentioned above, results containing a run where no solution dominates the reference point is discarded. The calculation of HV requires individuals to dominate the reference point, and no domination results in an

undefined HV.

Both metrics require information about the true PF to be calculated. Data about the true PF of the LIR problems are gathered from Fan [2017]. Data for the MW problems are gathered from Wang et al. [2020].

5.3.2 Setup Phase T2

T2 aims to answer RQ2 These tests focus on how the two BS perform on different problems. The aim here is to identify which problem characteristics the methods may be suited for and which pose a challenge for them and why. The two methods are tested on the benchmark problems listed in tables 5.13 and 5.14. The problems have been chosen due to their different characteristics such as problems with both convex and concave PFs. Also, problems from the MW test suite are added to increase the pool of problems with both similar and different characteristics.

Benchmark	**Shape**	**Region Size (Feasible/Infeasible)**	**Overlap**	**Disconnected**
MW5	CV/CV	M/M	P	X
MW6	CV/CV	S/L	P	X
MW9	CC/CV	M/L	N	-
MW10	CV/CC	S/L	P	X
MW11	CV/CC	M/L	P	X
MW13	CC/CC	M/L	P	X

Table 5.14: T2 Test Suite.

5.3.3 Setup Phase T3

T3 aims to answer RQ3. To accurately answer this question the evolutionary process during the binary- and the pull phase will be analysed. Table 5.15 highlights the problems used during this phase. The problems in the test suite were chosen due to them having an infeasible region between the unconstrained and the constrained PF.

To evaluate the traversal between the constrained and unconstrained PFs, the median of the populations IGD value will be presented when the populations are at the following states:

1. Last generation of push phase
2. First generation of pull phase
3. Last generation of pull phase

The last generation of the push phase defines the starting point of either BS method. The first generation of the pull phase will be used to showcase the effect of the BiS phase. In models not containing a BiS phase, the first pull generation is simply the next generation after the last push generation. The last generation of the pull phase allows comparison of how the CHM performed during the pull phase.

The HV and initial allowed constraint violation from these generations are also presented for selected problems to support the discussion. The development of FR_p is also plotted for selected problems to support the discussion.

Benchmark	**Shape**	**Region Size (Feasible/Infeasible)**	**Overlap**	**Disconnected**
LIR1	CV/CV	S/L	N	-
LIR2	CC/CC	S/L	N	-
LIR3	CV/CV	S/L	N	X
LIR4	CC/CC	S/L	N	X
LIR7	CC/CV	M/M	N	-
LIR8	CV/CV	M/M	N	-
LIR11	CC/CC	M/L	P	X
LIR12	CV/CC	M/L	P	X
MW9	CC/CV	M/L	N	X
MW11	CV/CC	M/L	P	X
MW13	CC/CC	M/L	P	X

Table 5.15: T3 Test Suite.

5.3.4 Setup Phase T4

T4 aims to answer RQ4. To accurately answer this question, the evolutionary process during the binary and the pull phase will be analysed. The IGD and CD will be logged and analysed to see how the diversity of the population changes. Visualisations of the population in the objective space is presented to show the populations behaviour. Benchmarks from the previous experimental phases are selected for discussion based on the results achieved, where the aim is to analyse why good and bad results occur.

5.4 Experimental Results

The following section discusses the results from the experimental testing described in sections 5.2 and 5.3. After performing Plan Phase T1 and and Plan Phase T2, it was realised that the results from these two phases would be more easily discussed together rather than separately. For this reason, section 5.4.1 discusses the results from phase T1 and phase T2. Then, section 5.4.2 discusses the results from phase T3. Finally, the results from phase T4 are elaborated in section 5.4.3.

5.4.1 Results Phase T1 and Phase T2

Tables 5.16 and 5.17 show a sample of the results from the experiments described in sections 5.2.1 and 5.2.2. Given the importance of comparison between the models, rather than the exact values of IGD and HV, the full tables of results have been placed in appendix .1. Table 1 shows the exact IGD and table 2 shows the exact

Problem		BiS-IE	PPS-IE	BiS-E	PPS-E	PPS-RSS
LIR1	MEAN	1.1045e-02	**8.8040e-03**	8.1249e-02	1.0060e-01	1.1054e-01
	STD	7.9447e-03	4.1645e-03	5.9163e-02	4.0359e-02	4.4406e-02
LIR3	MEAN	2.5340e-02	**7.5015e-03**	1.1112e-01	1.2076e-01	1.3406e-01
	STD	6.8518e-02	4.0834e-03	5.1968e-02	5.1519e-02	6.9498e-02
LIR7	MEAN	3.1774e-03	**2.9001e-03**	3.2108e-03	3.0568e-03	3.1797e-03
	STD	1.1356e-03	7.8821e-05	7.5886e-04	1.4729e-04	1.0240e-04
MW9	MEAN	2.0151e-01	1.9068e-01	**1.8612e-01**	2.6728e-01	2.8065e-01
	STD	5.1044e-03	3.1373e-03	4.2815e-03	6.0034e-03	9.7723e-03

Table 5.16: Sample of IGD results of BiS-IE, PPS-IE, BiS-E, PPS-E and PPS-RSS. Best performance is highlighted for each problem.

Problem		BiS-IE	PPS-IE	BiS-E	PPS-E	PPS-RSS
LIR1	MEAN	6.4321e-01	**6.4678e-01**	5.6248e-01	5.4716e-01	5.3873e-01
	STD	1.6697e-02	2.7374e-03	5.8156e-02	3.8310e-02	4.3014e-02
LIR3	MEAN	5.2058e-01	**5.3681e-01**	4.4402e-01	4.3296e-01	4.2586e-01
	STD	6.4708e-02	2.9635e-03	4.9291e-02	4.5763e-02	5.9883e-02
LIR7	MEAN	2.0368	**2.0402**	2.0370	2.0396e	2.0392e
	STD	8.3316e-03	1.5486e-03	6.7472e-03	1.7812e-03	1.2675e-03
MW9	MEAN	4.5257e-01	**4.6663e-01**	4.6161e-01	4.5543e-01	4.3263e-01
	STD	7.4792e-03	6.4746e-03	6.7706e-03	7.4763e-03	9.8116e-03

Table 5.17: Sample of HV results of BiS-IE, PPS-IE, BiS-E, PPS-E and PPS-RSS. Best performance is highlighted for each problem.

HV achieved by the models on the LIR problems. Similarly, tables 3 and 4 show the exact IGD and HV achieved for the MW problems.

Tables 5.16 and 5.17 show that there is a tendency for the PPS models to outperform their BiS counterparts and PPS-RSS. To get a deeper insight into the results and the trend they are showing, T-tests have been performed between the different models. These are shown in tables 5.20 and 5.23 to 5.25. The tables show the problems where there is a significant difference in performance. The Model column signifies which model performed significantly better ($P < 0.05$).

BiS-IE will primarily be compared against PPS-IE and BiS-E against PPS-E to analyse how BiS may affect performance. The reason for this is that BiS-IE and PPS-IE are similar except the addition of BiS in BiS-IE. Thus, it is likely that a difference in performance is caused by BiS, as there is no other dissimilarity between the models. This is the same for BiS-E and PPS-E. Finally, PPS-RSS is evaluated both against PPS-IE and PPS-E.

5.4.1.1 Binary Search

Table 5.20 shows the LIR problems where there was a significant difference in performance between BiS-IE and PPS-IE (table 5.18) and between BiS-E and PPS-E (table 5.19). LIR7 and LIR8 both have a medium to large infeasible area between the unconstrained and constrained PF. LIR9 and LIR12 both have overlapping unconstrained and constrained PFs.

Problem	IGD		HV	
	P Value	Model	P Value	Model
LIR7	0.30	-	**0.04**	**PPS-IE**
LIR8	**0.00**	**BiS-IE**	0.30	-
LIR9	**0.01**	**PPS-IE**	0.11	-
LIR12	0.23	-	**0.01**	**PPS-IE**

Table 5.18: T-test comparing BiS-IE and PPS-IE.

Problem	IGD		HV	
	P Value	Model	P Value	Model
LIR8	**0.03**	**BiS-E**	**0.01**	**BiS-E**
LIR9	0.12	-	**0.02**	**PPS-E**
LIR12	**0.01**	**PPS-E**	**0.02**	**BiS-E**

Table 5.19: T-test comparing BiS-E and PPS-E.

Table 5.20: T-tests comparing BiS-IE with PPS-IE and BiS-E with PPS-E on LIR problems.

Problem	IGD		HV	
	P Value	Model	P Value	Model
MW9	**0.00**	**PPS-IE**	**0.00**	**PPS-IE**
MW11	**0.00**	**BiS-IE**	**0.00**	**BiS-IE**

Table 5.21: T-test comparing BiS-IE with PPS-IE.

Problem	IGD		HV	
	P Value	Model	P Value	Model
MW9	**0.00**	**BiS-E**	**0.00**	**BiS-E**
MW11	**0.00**	**BiS-E**	**0.00**	**BiS-E**

Table 5.22: T-test comparing BiS-E with PPS-E.

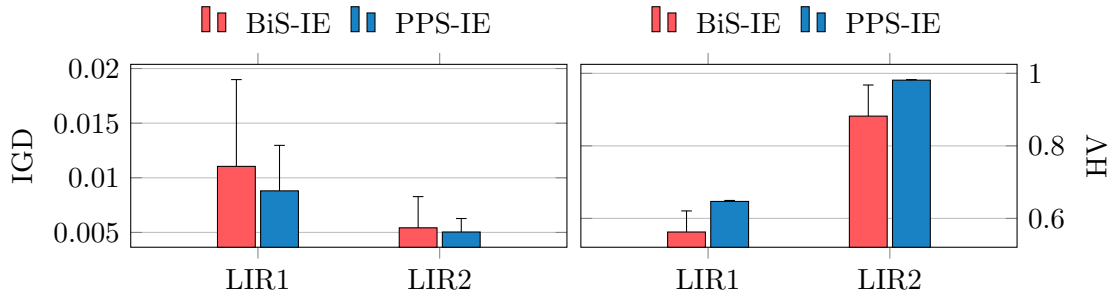
Table 5.23: T-tests comparing BiS-IE with PPS-IE and BiS-E with PPS-E on MW problems.

Table 5.23 illustrate the same as table 5.20, however for the MW problems.

Tables 5.20 and 5.23 show that there is a significant difference in performance only on a few of the problems from the test suites used (see tables 5.13 and 5.14). Thus, even though tables 5.16 and 5.17 indicate that PPS outperforms the BiS models the difference is not significant. To get a better understanding of these results, further analysis was performed. The following discussion has been divided into three sections focusing on different problem characteristics: Small Feasible Regions, Disjoint PFs, and Convex and Concave PFs.

Small Feasible Regions

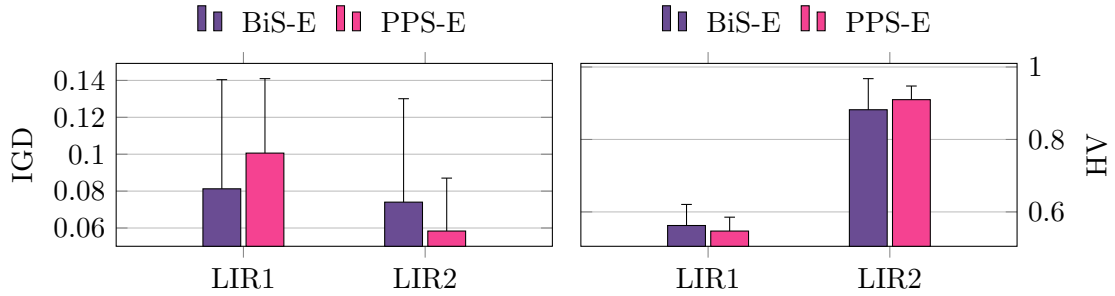
LIR1 and LIR2 are two problems with small feasible regions. Figure 5.12 shows that PPS-IE outperforms BiS-IE in regards to both IGD and HV on LIR1 and LIR2.



(a) IGD comparison. **(b)** HV comparison.

Figure 5.12: Comparison of mean IGD and HV between BiS-IE and PPS-IE on LIR1 and LIR2.

Figure 5.12a shows that PPS-IE achieves a better IGD and a lower SD. Similarly, figure 5.12b shows that PPS-IE achieves better HV and a lower SD than BiS-IE. This was unexpected as the introduction of BiS was thought to improve performance on problems where the unconstrained and constrained PF were separated by an infeasible region. Based on the results from the preliminary testing (see table 5.1) it was expected that small feasible regions would not have a negative effect on the performance of BiS. It was thus expected that BiS-IE would perform better or on par with PPS-IE on these problems. The T-test ($P < 0.05$) showed that no *significant* difference between the two models is detected on these problems.



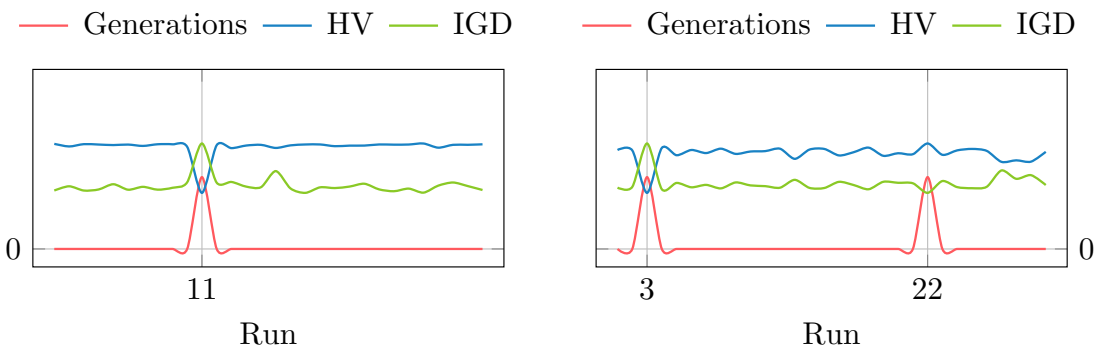
(a) IGD comparison. **(b)** HV comparison.

Figure 5.13: Comparison of mean IGD and HV between BiS-E and PPS-E on LIR1 and LIR2.

Figure 5.13 shows the performance of BiS-E compared to PPS-E on LIR1 and LIR2. Similarly to BiS-IE, BiS-E is outperformed on LIR2 in regards to IGD and HV as shown in figures 5.13a and 5.13b respectively. However, BiS-E performs better than PPS-E on LIR1 achieving a lower IGD and a higher HV. Note however that the SD for BiS-E is larger than for PPS-E on LIR1.

To understand why the addition of BiS did not improve the performance on problems with small feasible regions, each of the 30 runs were further analysed. In order to accomplish this all runs of the models were plotted in figures 5.14 and 5.15.

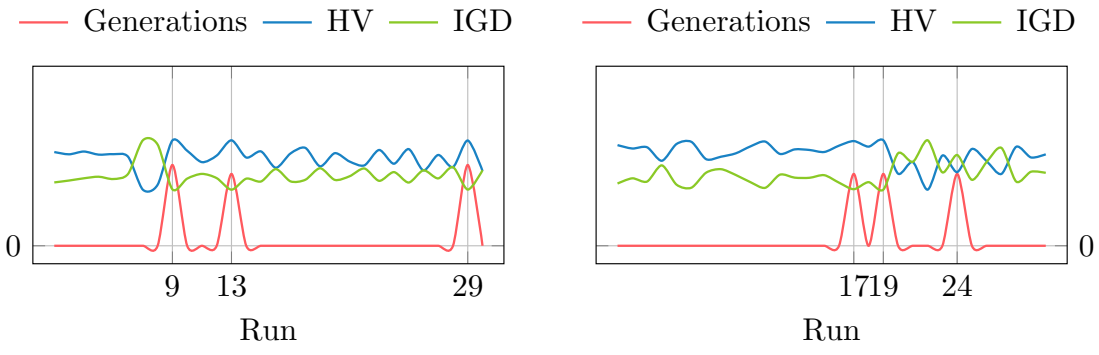
The preliminary tests in section 5.1.1.3 discussed that having a feasible set containing only a single individual, BiS would be able to develop a high diversity and a good performance, indicated by a low IGD and a high HV. However, further exam-



(a) LIR1 **(b)** LIR2

Figure 5.14: Effect of BiS on IGD and HV for BiS-IE on LIR1 and LIR2.

ination shows that this may not always be the case. Figure 5.14a shows that only one of the 30 runs of BiS-IE performed BiS on LIR1, as indicated by the spike in the generations line for run 11. The blue line shows a noticeable dip in HV for this run. Similarly, the green line shows an increase in IGD. For LIR2, BiS was performed for run 3 and run 22 shown in figure 5.14b. In run 3, the introduction of BiS resulted in an inferior performance just as in run 11 for LIR1. On the other hand, run 22 for LIR2 showed a better performance both in HV and IGD. Run 22 managed to be the best performing of all 30 runs on LIR2. Figure 5.15a show the 30 runs of BiS-E



(a) LIR1 **(b)** LIR2

Figure 5.15: Effect of BiS on IGD and HV for BiS-E on LIR1 and LIR2.

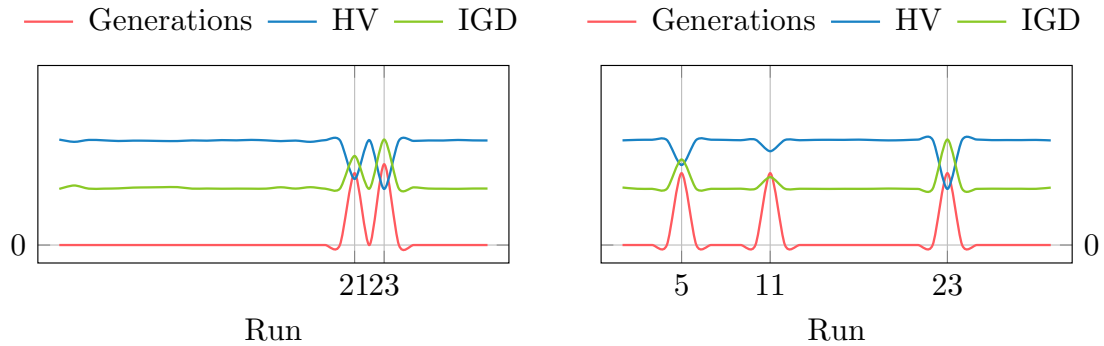
on LIR1. In run 9, 13 and 29 BiS was performed. In all cases, the introduction of BiS produced better results than when BiS was not performed. For LIR2, BiS was performed for run 17, 19 and 24 as shown in figure 5.15b. For run 17 and 19 the performance was increased. For run 24 the performance was not increased, however the introduction of BiS did not reduce performance to the same extent as the cases in figure 5.14. The results in figure 5.15 show that BiS improves the performance more often than reducing it, contradicting what is seen in figure 5.14 where the opposite is the case. The preliminary testing indicated that BiS would still perform well even when problems had small feasible regions and the number of feasible individuals found during the push phase was low (see section 5.1.1.3). The results from experimental testing however show that BiS may have a negative impact on

performance after all.

It is clear from figures 5.14 and 5.15 that BiS was performed a total of 3 times for BiS-IE on these two problems, and a total of 6 times for BiS-E. The lack of runs where BiS was performed is due to the small feasible regions making it unlikely to find feasible individuals during the push phase. With no feasible individuals found, the BiS is skipped, and the model goes straight into the pull phase. The low number of runs where BiS was performed for these two problems makes it impossible to draw any definitive conclusions. To be able to better analyse the performance of BiS on problems with small feasible regions, more runs where BiS was performed is needed. A possibility would be to perform runs until 30 runs with BiS was obtained for both models. This would allow for a more clear analysis of the performance of BiS on problems with small feasible regions. However, the lack of occurrences of BiS implies that the consistency of the model is low and the likelihood of it initiating is unpredictable. Thus, BiS is not suited for problems with small feasible regions.

In figure 5.14 the fluctuation in performance over the 30 runs is less, with BiS-runs introducing most change. In figure 5.15, the performance for each run varies more, even when BiS is not performed. This is likely due to the IϵCH method which periodically increases the constraint relaxation in BiS-IE (see equation (3.4)). Thus, the population is able to move away from the constrained PF creating diversity in the population and then finding new parts of the PF. ϵCH on the other hand has only a single opportunity to approach the PF. As there is no increase in constraint relaxation, the population may lack the diversity needed to properly move around in the search space. Thus, depending on the initial approach to the constrained PF, the performance may vary. The difference in stability can also be seen in tables 5.16 and 5.17 where the exact SD for LIR1 is listed.

Disjoint PFs



(a) LIR3 **(b)** LIR4

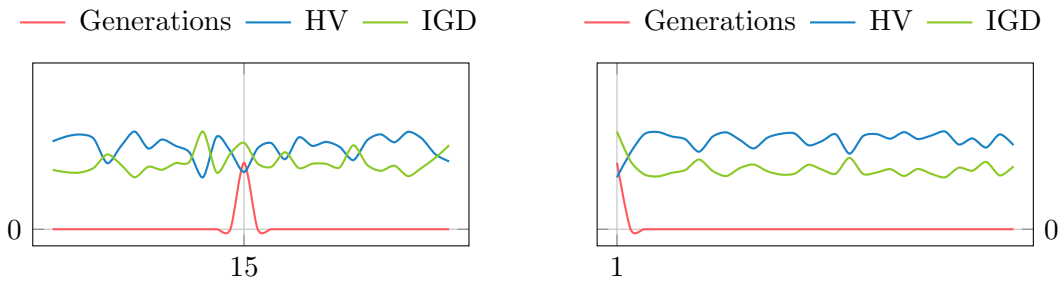
Figure 5.16: Effect of BiS on IGD and HV for BiS-IE on LIR3 and LIR4.

LIR3 and LIR4 are two problems with disjoint PFs in addition to small feasible regions. Figures 5.16 and 5.17 show the same information as figures 5.14 and 5.15 respectively, but for LIR3 and LIR4.

The figures show that BiS was performed even fewer times than on LIR1 and LIR2. This was expected due to the disjoint nature of the constrained PF and

making feasible area of these problems even smaller than those of LIR1 and LIR2 (see figure 5.9). Therefore, the models would have greater problems identifying any feasible individuals during the push phase.

For all the occurrences of BiS in both models, the performance was reduced. This can clearly be seen in figure 5.16a where the IGD is increased and the HV is reduced for run 21 and 23, and in figure 5.16b for run 5, 11 and 23. The same is shown in figure 5.17a where run 15 had one of the worst performances of the 30 runs and in figure 5.17b where run 1 shows a clear dip in HV and increase in IGD resulting in the worst run. LIR1 and LIR3, and LIR2 and LIR4 would be



(a) LIR3 **(b)** LIR4

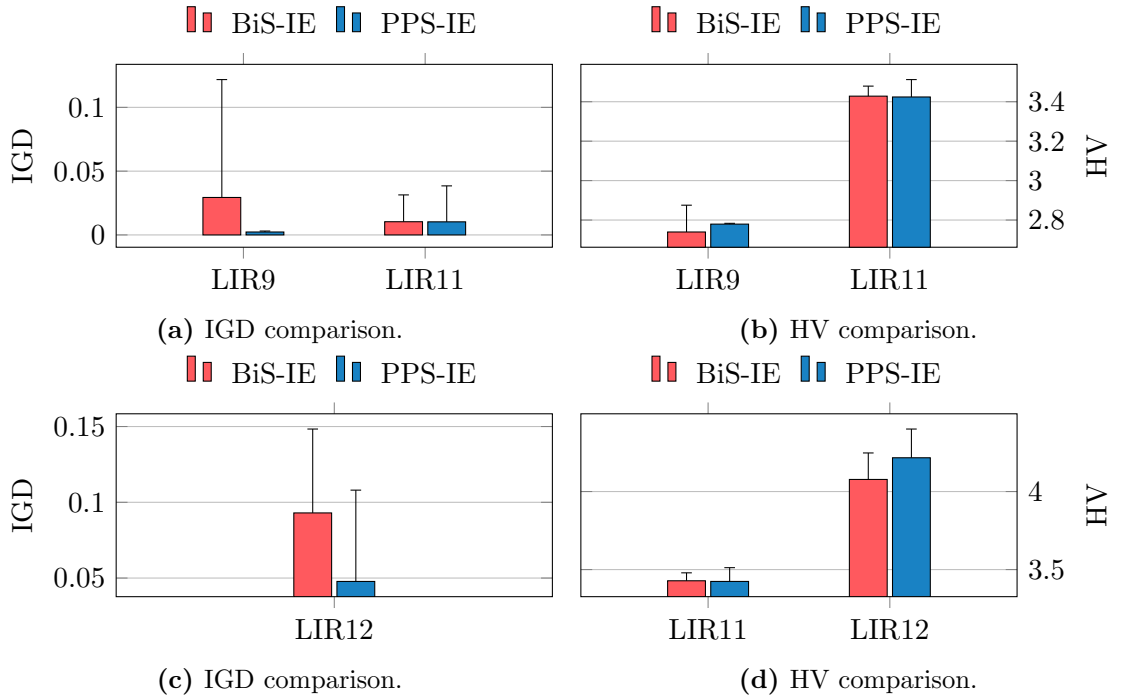
Figure 5.17: Effect of BiS on IGD and HV for BiS-E on LIR3 and LIR4.

identical if LIR3 and LIR4 did not have disjoint PFs. Thus, the reason for the poor performance of BiS on these problems is attributed to this difference and how the BiS models work. The disjoint nature of the PF makes it not only more difficult to identify feasible individuals during the push phase, but it also makes it more difficult for the population to identify all different sections of the PF. For BiS to be able to approximate all the disjoint sections of the constrained PF a feasible individual must have been found in all of them. If only a single individual in a specific section of the PF is identified during the push phase, BiS will move the whole population to this section. In addition, the reduced initial constraint relaxation (see section 5.1.1.2) makes it difficult for the population to locate the other sections of the constrained PF by prohibiting the population to move too far away from the feasible area. Again, it is evident that the results of BiS-IE are fluctuating less than BiS-E as seen i figures 5.16 and 5.17.

LIR9-12 also have disjoint sections. Figure 5.18 shows that BiS-IE is outperformed on LIR9 and LIR12. For LIR10, the performance of BiS-IE is similar to that of PPS-IE as shown in figures 5.18a and 5.18b.

Figure 5.19 shows the performance of BiS-E and PPS-E on LIR9, LIR11 and LIR12. Similarly to BiS-IE in figures 5.18a and 5.18b, BiS-E is outperformed on LIR9 as shown in figures 5.19a and 5.19b. Figure 5.19c shows that opposite to BiS-IE and PPS-IE, the performance of BiS-E on LIR11 is poorer than BiS-E and the performance on LIR12 is more similar.

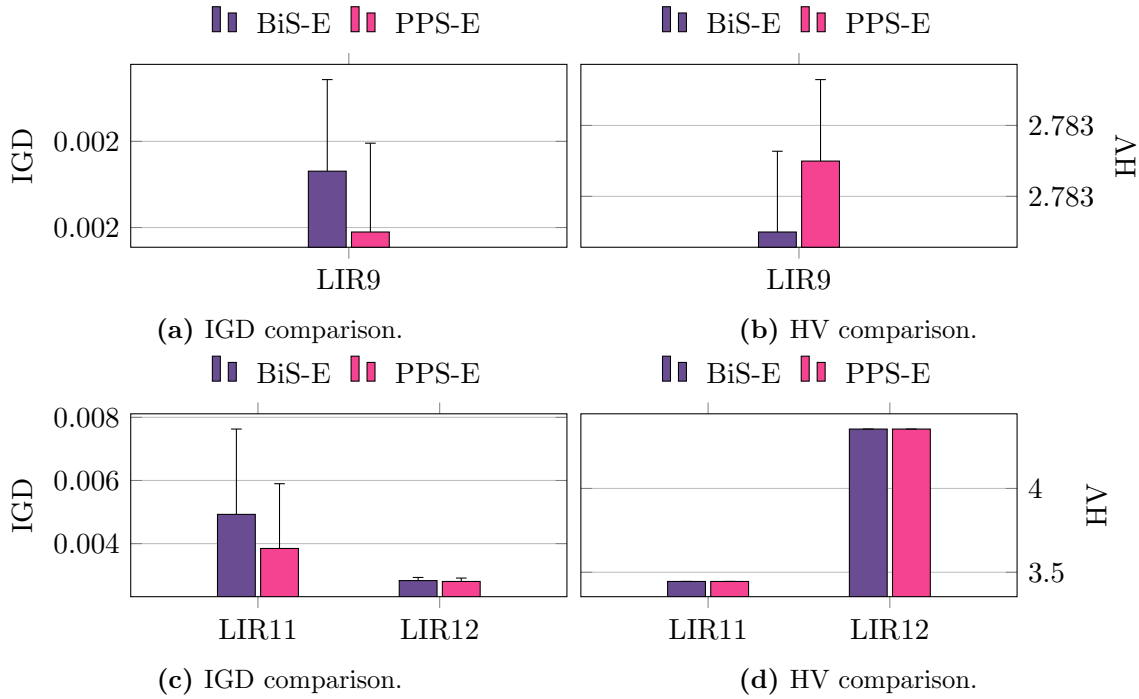
BiS-IE and BiS-E show a noticeable SD for these problems. Figures 5.18a and 5.18b show that it is especially the case on LIR9 for both models. To better understand why the performance was so unstable for the problems in figures 5.18



(a) IGD comparison. **(b)** HV comparison.

(c) IGD comparison. **(d)** HV comparison.

Figure 5.18: Comparison of mean IGD and HV between BiS-IE and PPS-IE on LIR9, LIR11 and LIR12.



(a) IGD comparison. **(b)** HV comparison.

(c) IGD comparison. **(d)** HV comparison.

Figure 5.19: Comparison of mean IGD and HV between BiS-E and PPS-E on LIR9, LIR11 and LIR12.

and 5.19, the 30 runs for each of them were plotted similarly as with LIR3 and LIR4. These plots showed that there were no clear correlation between the number of generations the search was in the binary phase and the performance of the model.

Figure 5.20 has been included to convey this. The red line shows that nearly all of the 30 runs performed BiS as run 7 was the only one where BiS was performed for 0 generations. Also, as the graph shows, there is no clear correlation between the BiS search and the performance of the model. Similar graphs were generated for all the problems, however they are left out as they show similar behaviour. The lack of correlation between BiS is dissimilar to LIR1-4 where a clearer dip or spike in performance can be seen in figures 5.14 to 5.17.

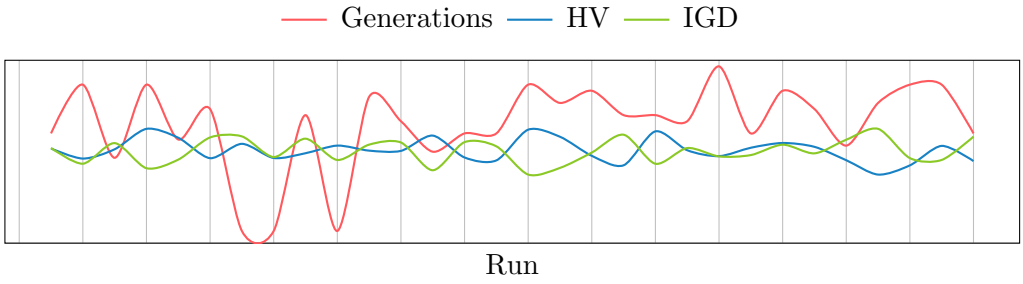


Figure 5.20: LIR9: Comparison of 30 runs of BiS-E.

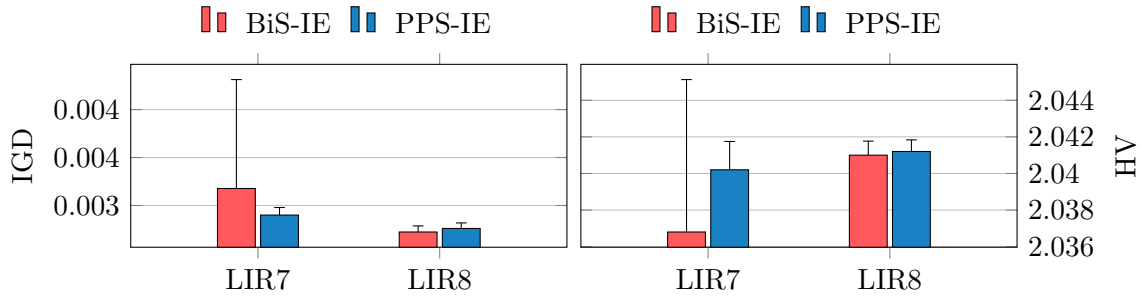
In addition to having a disjoint PF, LIR9 also has overlapping PFs (see figure 5.9). During the push phase, the population ignores constraints and is approximating the unconstrained PF. Assuming that the population reaches the unconstrained PF then most, if not all, of the disjoint segments of the constrained PF will have been found. Thus, the feasible set produced by BiS should contain feasible individuals located in all of the disjoint segments. However, this does not seem to show any performance increase, as PPS-IE performed significantly better than BiS-IE on LIR9 in regards of IGD as seen in table 5.18.

Figures 5.18 and 5.19 show that there is a high SD for several of the problems. Due to the larger feasible areas evident in LIR9-12 (see figure 5.9) compared to LIR1-4, it was expected that several feasible individuals would be found during the push phase and thus both BiS models would outperform PPS. This does however not seem to be the case for all these problems. As seen in tables 5.18 and 5.19, only LIR9 and LIR12 show a significant difference in performance between BiS-IE and PPS-IE, and BiS-E and PPS-E.

Convex and Concave PFs

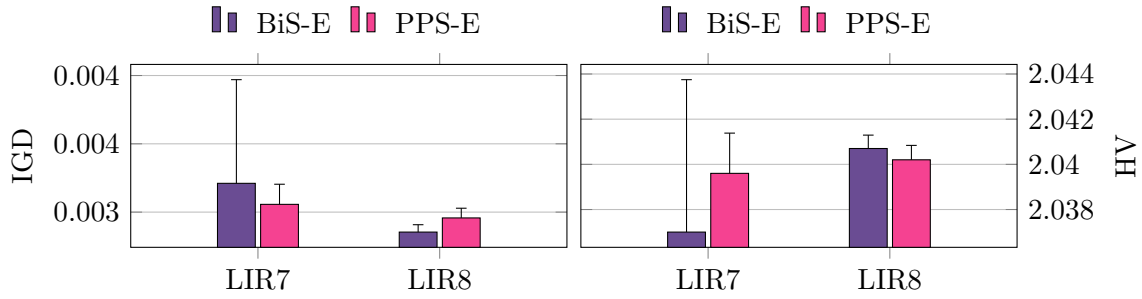
It was believed that BiS-IE and BiS-E would be superior on LIR7 and LIR8. This was hypothesised due to the sizeable feasible area close to the boundary making it more likely to identify feasible individuals during the push phase. However, they only significantly enhanced the performance on LIR8 as seen in tables 5.18 and 5.19.

MW9 has similar characteristics as LIR7: a concave unconstrained PF and a convex constrained PF (see figures 5.9 and 5.10). Figures 5.21 and 5.23 show that BiS-IE performed worse than PPS-IE on LIR7 and MW9. BiS-E on the other hand performed better than both BiS-IE and PPS-E on MW9 as seen in figure 5.23. Also, table 5.19 shows that there is a significant difference in both IGD and HV between BiS-E and PPS-E. The results may indicate that BiS is only beneficial where the constrained and unconstrained PF have the same shape. However, this can not be



(a) IGD comparison. **(b)** HV comparison.

Figure 5.21: Comparison of mean IGD and HV between BiS-IE and PPS-IE on convex problem (LIR7) and concave problem (LIR8).



(a) IGD comparison. **(b)** HV comparison.

Figure 5.22: Comparison of mean IGD and HV between BiS-E and PPS-E on convex problem (LIR7) and concave problem (LIR8).

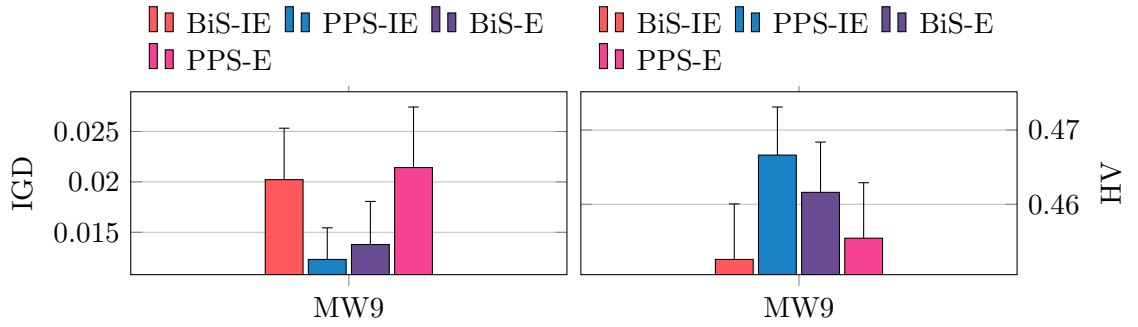
said for certain without a more thorough investigation into these characteristics of the PF. Such an investigation is outside the scope of this thesis, thus future work should look into this topic.

Figure 5.24 shows a noticeable difference between BiS-IE and PPS-IE, and BiS-E and PPS-E respectively. The models utilising BiS show better performance on these problems which is backed up by the T-tests in tables 5.18 and 5.19. In addition to the larger feasible area mentioned in section 5.4.1.1, this is likely caused by the initial constraint relaxation being set by the BiS phase. It is believed that the reduced initial constraint relaxation results in a superior initial position and more generations to explore the PF during the pull phase, as discussed in section 5.1.1.2.

5.4.1.2 Push Pull Reduced Search Space

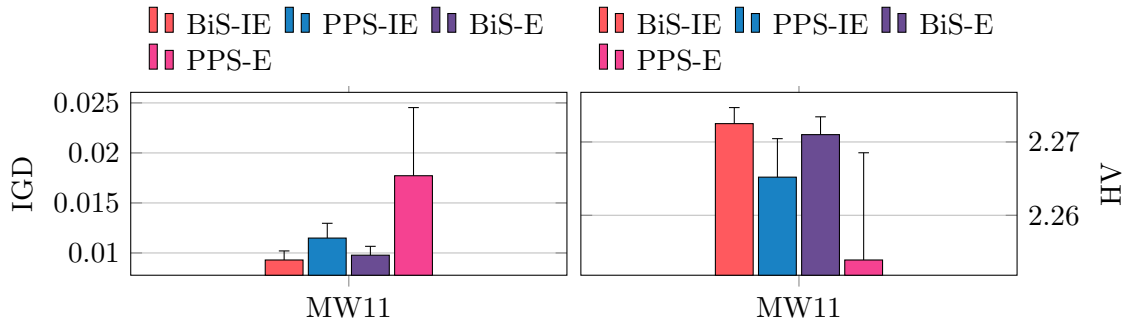
Tables 5.24 and 5.25 show T-test comparisons between PPS-RSS and the two PPS models. Problems yielding a significant difference in performance for either IGD or HV have been omitted. The tables clearly display that PPS-RSS had problems performing on par with the other models which substantiates the trend seen in tables 5.16 and 5.17. PPS-RSS was significantly worse than both PPS-IE and PPS-E for almost all problems in the test suite.

Table 5.24 gives a clear picture of PPS-RSS being outperformed. An analysis of the performance is outlined to understand these results. The discussion will follow



(a) IGD comparison on MW9. **(b)** HV comparison on MW9.

Figure 5.23: Comparison of mean IGD and HV between BiS-IE, PPS-IE, BiS-E and PPS-E on a convex problem (MW9).



(a) IGD comparison on MW11. **(b)** HV comparison on MW11.

Figure 5.24: Comparison of mean IGD and HV between BiS-IE, PPS-IE, BiS-E and PPS-E on MW11.

Problem	IGD		HV	
	P Value	Model	P Value	Model
LIR1	**0.00**	**PPS-IE**	**0.00**	**PPS-IE**
LIR2	**0.00**	**PPS-IE**	**0.00**	**PPS-IE**
LIR3	**0.00**	**PPS-IE**	**0.00**	**PPS-IE**
LIR4	**0.00**	**PPS-IE**	**0.00**	**PPS-IE**
LIR5	**0.00**	**PPS-IE**	**0.00**	**PPS-IE**
LIR6	0.56	-	**0.00**	**PPS-IE**
LIR7	**0.00**	**PPS-IE**	**0.01**	**PPS-IE**
LIR8	**0.00**	**PPS-IE**	**0.00**	**PPS-IE**
LIR9	**0.00**	**PPS-RSS**	**0.00**	**PPS-RSS**
LIR10	**0.03**	**PPS-IE**	**0.00**	**PPS-IE**
LIR12	**0.00**	**PPS-RSS**	**0.00**	**PPS-RSS**

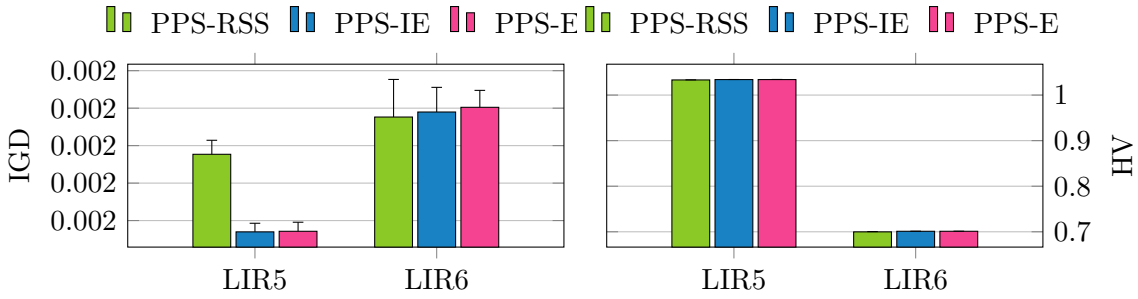
Table 5.24: T-test comparing PPS-RSS and PPS-IE for the LIR problems tested. The model column signifies which model performed significantly better ($P < 0.05$).

the same layout as in section 5.4.1.1. The line charts used in section 5.4.1.1 are not used as their goal was to analyse the impact of the new binary phase. PPS-RSS does not incorporate a new phase but changes the CHM method used. Therefore, looking at the results of all the 30 runs will not give more insight than looking at

Problem	IGD		HV	
	P Value	Model	P Value	Model
LIR4	**0.03**	**PPS-E**	**0.03**	**PPS-E**
LIR5	**0.00**	**PPS-E**	**0.00**	**PPS-E**
LIR6	0.16	-	**0.00**	**PPS-E**
LIR7	**0.00**	**PPS-E**	0.28	-
LIR8	**0.00**	**PPS-E**	**0.05**	**PPS-E**
LIR9	**0.00**	**PPS-RSS**	0.44	-
LIR10	**0.00**	**PPS-E**	**0.00**	**PPS-E**

Table 5.25: T-test comparing PPS-RSS and PPS-E for the LIR problems tested. The model column signifies which model performed significantly better ($P < 0.05$).

the bar charts highlighting the mean IGD and HV together with SD values.

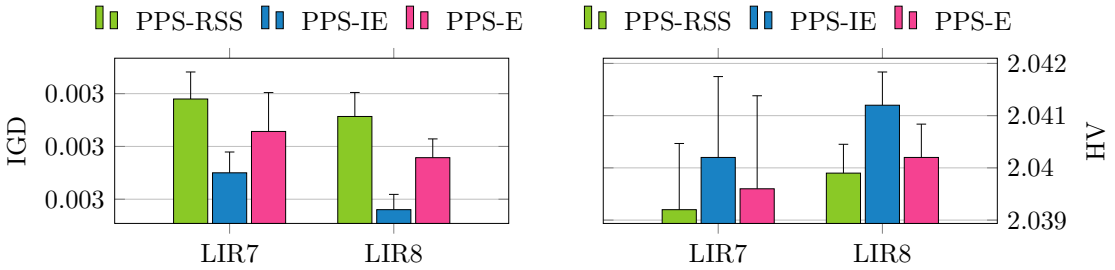


(a) IGD comparison. **(b)** HV comparison.

Figure 5.25: Comparison of mean IGD and HV between PPS-RSS, PPS-IE and PPS-E on LIR5 and LIR6.

It was expected that all models would perform close to identical on LIR5 and LIR6. This was due to the likelihood of discovering the whole constrained PF during the push phase due to the unconstrained and constrained PF being identical. However, figure 5.25a shows that PPS-RSS achieves a larger IGD than the PPS models. This is a result of the problem having large infeasible regions blocking the way to the PF (see section 2.5 and figure 5.9). The population is able to locate the constrained PF during the push phase. When the CHM kicks in during the pull phase, the population is pulled away from the PF and towards the boundary between feasible and infeasible space. PPS-RSS behaves similarly when solving LIR6. However for LIR6, PPS-RSS performed similarly to PPS-IE and PPS-E as seen in figures 5.25a and 5.25b highlighting the achieved IGD and HV on LIR5 and LIR6. This indicates that the initialisation of δ_{in} is dependent on the problem, and may hinder the population in properly exploring the constrained PF if it is too low. This behaviour of PPS-RSS during the evolutionary process is discussed in more detail in section 5.4.3.2.

Note the HV achieved by PPS-IE, PPS-E and PPS-RSS on LIR5 and LIR6. Figure 5.25b indicates that the coverage is nearly identical, even with the greater difference in IGD shown in figure 5.25a. The behaviour causing this will be discussed in section 5.4.3.2

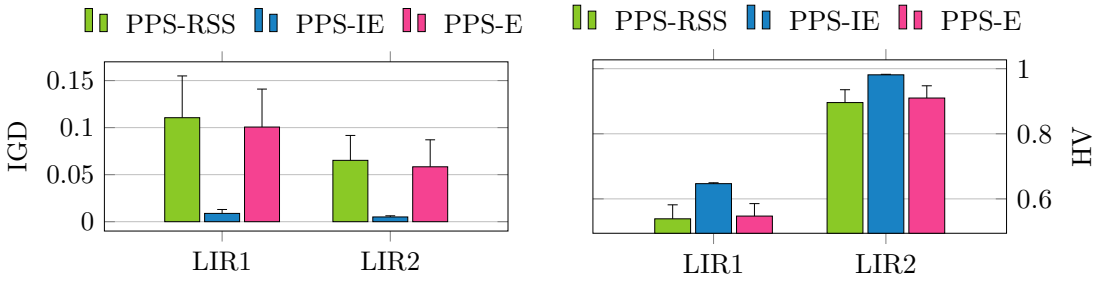


(a) IGD comparison. **(b)** HV comparison.

Figure 5.26: Comparison of mean IGD and HV between PPS-RSS, PPS-IE and PPS-E on LIR7 and LIR8.

Figure 5.26 shows that PPS-RSS is outperformed on LIR7 and LIR8. It was believed that the performance of PPS-RSS would be similar to that of the PPS models due to the similarities between how PPS-RSS and PPS-E shrink their boundaries. The reason for the poor performance is attributed to the fact that PPS-RSS shrinks the boundaries much slower than PPS-IE and PPS-E. Thus, the population requires more generations to cross the infeasible region between the unconstrained and constrained PF. Using more generations to cross this region reduces the number of generations the population explores the boundary region between feasible and infeasible space, where the constrained PF is located. Therefore, the ability of the population to properly explore the constrained PF is reduced.

Small Feasible Regions

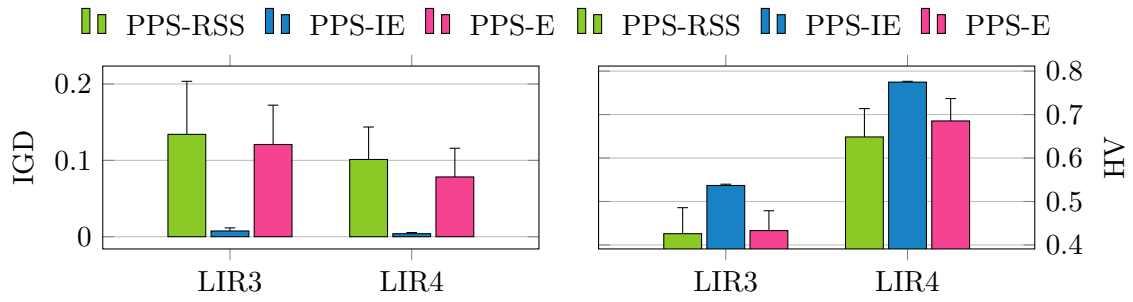


(a) IGD comparison. **(b)** HV comparison.

Figure 5.27: Comparison of mean IGD and HV between PPS-RSS, PPS-IE and PPS-E on LIR1 and LIR2.

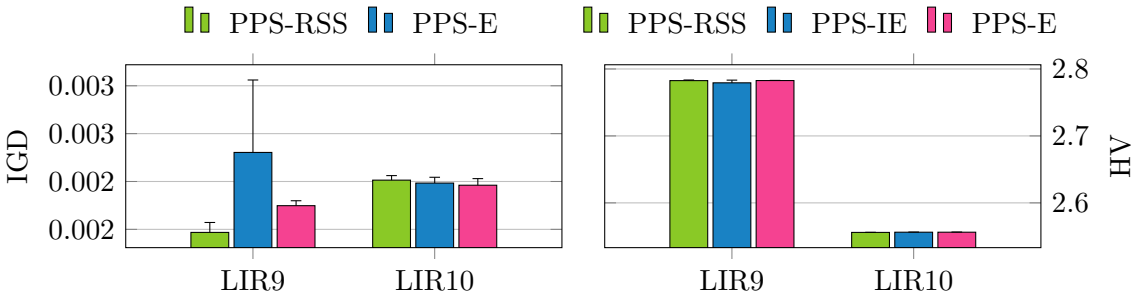
Figure 5.27 shows the performance of PPS-RSS compared to PPS-IE and PPS-E on two problems with small feasible areas. From the bar charts it is clear that PPS-IE outperforms PPS-RSS on both the problems. The performance of PPS-RSS and PPS-E is more similar. This is believed to be due to the similar approach of shrinking the boundaries. PPS-IE allows the relaxation of constraints to increase if a certain FR_p is reached (see equation (3.4)), helping it approach the small feasible space of the problems multiple times. PPS-E and PPS-RSS do not increase the constraint relaxation again, strictly shrinking it. Thus, the two methods have a harder time exploring the small feasible regions of these problems.

Disjoint PFs



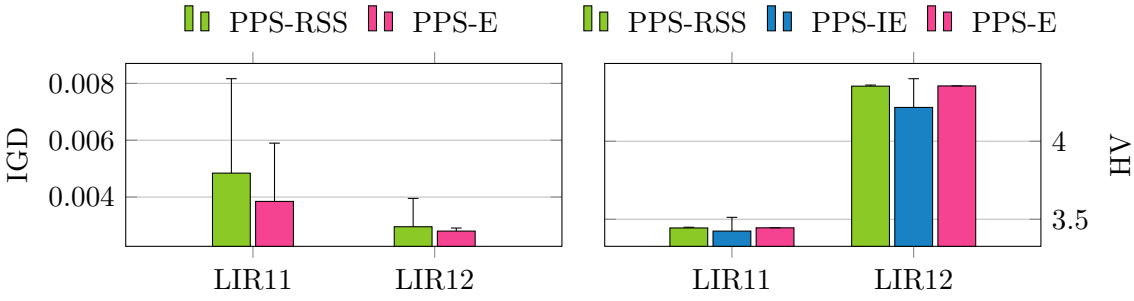
(a) IGD comparison. **(b)** HV comparison.

Figure 5.28: Comparison of mean IGD and HV between PPS-RSS, PPS-IE and PPS-E on LIR3 and LIR4.



(a) IGD comparison. **(b)** HV comparison.

Figure 5.29: Comparison of mean IGD and HV between PPS-RSS, PPS-IE and PPS-E on LIR9 and LIR10.



(a) IGD comparison. **(b)** HV comparison.

Figure 5.30: Comparison of mean IGD and HV between PPS-RSS, PPS-IE and PPS-E on LIR11 and LIR12.

Figure 5.28 shows the performance of PPS-RSS compared to PPS on two problems with disjoint PFs and small feasible regions. Also, figures 5.29 and 5.30 show the performance on several problems with disjoint PFs. Due to the poor performance of PPS-IE on LIR12, PPS-IE is not shown in figure 5.30a. PPS-IE is omitted to enhance readability in the graph and make the difference between the methods

more prominent. The figures show that PPS-RSS performs similarly to PPS-E on most of them. Figure 5.28 shows similar performance on LIR3 however a large SD is present. For LIR4 there is a more notable difference in performance which is affirmed by table 5.25 highlighting that PPS-E performed significantly better than PPS-RSS on this problem. Similarly, figure 5.30a shows a clear difference in performance on LIR11 and PPS-RSS has a large SD. For LIR12, the IGD is similar, however PPS-RSS has a large SD compared to PPS-E.

Initially, the poor performance of PPS-RSS on problems LIR3, LIR4 and LIR11 was assumed to be a result of how the infeasible regions are oriented in the objective space. For LIR3 and LIR4, the disjoint nature of their PFs is created by having one or several constraints splitting it into several pieces. Thus, these constraints will be active at certain points of the constrained PF. However, they will not be the only active constraints, and they will not be active along the whole constraint boundary. It was hypothesised that in the case that only these constraints were detected as active constraints during ACD (see section 4.2.10.1), the population would simply oscillate in-place instead of moving closer to the constrained PF. However, due to the value of Val specifically used for LIR1-4 (see table 5.11) this was not the case.

It was later hypothesised that the initialisation of the constraint boundaries was the reason for the poor performance on LIR3 and LIR4. To investigate this, a new set of experiments were performed logging the max constraint violation used for the initial constraint boundaries for eight runs on LIR3 and LIR4. Table 5.26 shows that the constraint violation is large for these two problems. The preliminary testing found that $Val = 0.26$ was appropriate for these problems. This indicates that 0.26 is a suitable estimate of the distance between the population and a boundary between feasible and infeasible regions. The values in table 5.26 clearly show a larger max constraint violation than 0.26. This indicates that using the max constraint violation of the population to initialise constraint boundaries may not always be suitable. In the case of these problems, the boundary values are too large and the shrinking of the boundaries is not sufficiently fast to make up for this.

	Mean	**SD**
LIR3	7.75e+01	1.25e+01
LIR4	7.02e+01	1.71e+01

Table 5.26: Mean max constraint violation for 10 runs of LIR3 and LIR4.

LIR9-LIR12 have either overlapping unconstrained and constrained PFs or they are close to each other (see figure 5.9). Therefore, it is highly likely that RSS is able to detect the active constraints consistently. The ACD process is among other aspects dependent on the value of Val and the distance of the population to the different borders between feasible and infeasible areas created by constraints. Assuming that the population has been able to approximate the unconstrained PF, if the problem has overlapping PFs, then the distance to the active constraints will be small. Also, for the problems with the unconstrained and constrained PF close to each other, Val should be sufficiently large. Thus, RSS detects the segments on the disconnected PF consistently, and performs best among all models on LIR9. The cause of this improved behaviour will be discussed in section 5.4.3.2.

5.4.1.3 Summary

Section 5.4.1 has discussed the results from executing the experimental plans in sections 5.2.1 and 5.2.2.

The T-tests performed show that BiS-IE and BiS-E perform similarly to PPS-IE and PPS-E respectively. It was hypothesised that the BiS models would outperform the PPS models on problems with infeasible regions separating the unconstrained and constrained PF. This was the case for LIR8 and MW11 where both BiS models outperformed their PPS counterparts. In addition, BiS-E significantly outperformed PPS-E on MW9.

After taking a closer look at the behaviour of BiS the performance was seen to both improve and reduce performance on problems with small feasible regions. For problems with disjoint PFs, the introduction of BiS reduced the performance on several of the problems. For LIR9, BiS-IE was significantly outperformed in regards to IGD and BiS-E was significantly outperformed in regards to HV. PPS-IE significantly outperformed BiS-IE on LIR12 in regards to HV, unlike PPS-E which was significantly outperformed by BiS-E on the same problem in regards to the same metric. However, PPS-E managed to significantly outperform BiS-E in regards to IGD.

The T-tests performed show that PPS-RSS is outperformed on nearly all the benchmark problems. It was hypothesised that PPS-RSS would perform on par with PPS however this was not the case. To better understand the results, a more thorough analysis was performed. The analysis shows that PPS-RSS performs similarly to PPS-E. This is credited to the fact that their CHM operate similarly. Also, the shrinking of boundaries in PPS-RSS is slower than PPS giving the population fewer generations to explore the constrained PF as more generations are used to cross infeasible regions.

5.4.2 Results Phase T3

Section 5.4.1 focuses on the end performance, looking at the archive returned at the end of each run. To further analyse the behaviour of the two Boundary Search methods during the evolutionary process, section 5.4.2 focuses on the behaviour and evolution of the population *during* the evolutionary process. Therefore, the results from section 5.4.2 highlight the population, not the archives of the different models. To evaluate the traversal of BiS models, the difference in median IGD between the *last* generation of the push phase and the *first* generation of the pull phase is calculated. For the RSS model, the difference between the *last* generation of the push phase and the *last* generation of the pull phase is used. The median is calculated for two generations and then the difference between them is calculated. The median is selected over the mean due to different runs initialising the phases at different generations which may introduce anomalies in the data when using mean values. The exact values may be found in the appendix (see table 5).

Problem	BiS-IE	PPS-IE
LIR7	61.30%	50.89%
LIR8	49.71%	30.76%
LIR11	11.91%	0.02%
LIR12	18.72%	11.29%
MW9	13.89%	0.00%
MW11	25.08%	-0.02%

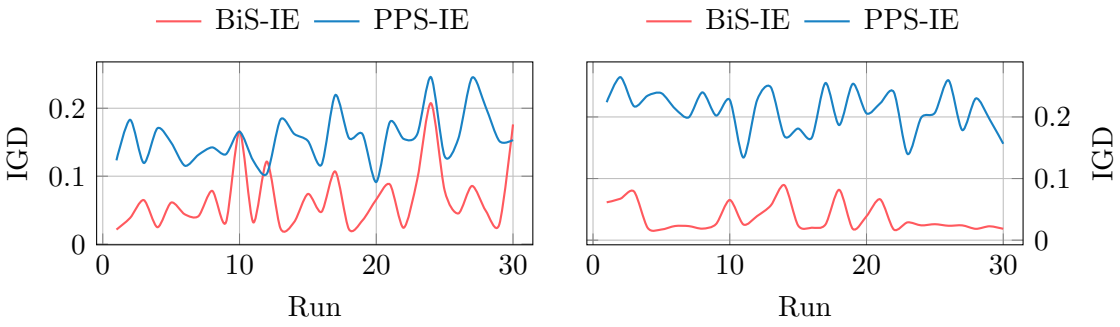
Table 5.27: Percentage increase in median IGD performance between the end of push phase and the start of the pull phase for BiS-IE and PPS-IE.

5.4.2.1 Binary Search

The difference in median IGD between the *end* of the push phase and the *start* of the pull phase is calculated and presented as percentages in table 5.27. The table contains the problems where BiS-IE had a higher increase than PPS-IE as these results give a clear comparison of the difference in incorporating BiS or not.

The results in table 5.18 highlight that BiS-IE increases end performance for LIR8, but not LIR7. Table 5.27 shows that BiS does in fact increase the convergence during the evolutionary process for both problems. The reason why the end performance is not increased when the convergence during evolution is increased is further analysed.

Figure 5.31 shows the IGD achieved during the first generation of the pull phase for all the 30 runs of LIR7 and LIR8. The performance of BiS-IE is highlighted by a red line and the performance of PPS-IE by a blue line.

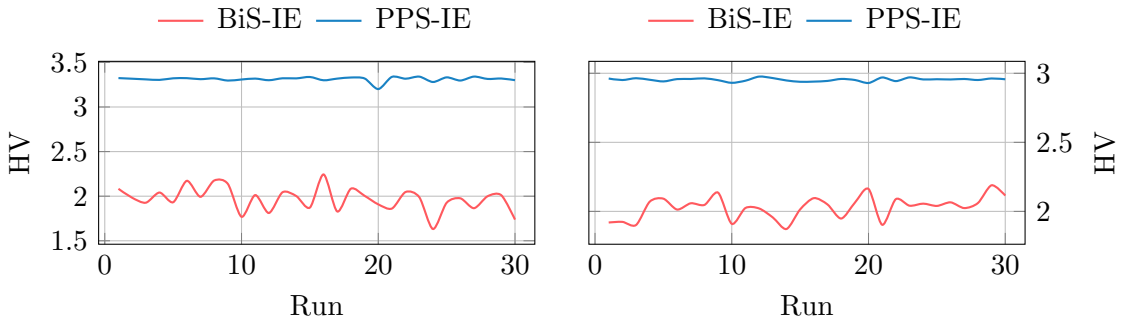


(a) LIR7 **(b)** LIR8

Figure 5.31: First generation of pull phase: Comparison of IGD between BiS-IE and PPS-IE on LIR7 and LIR8.

Figures 5.31a and 5.31b show a trend of BiS-IE achieving a better IGD than PPS-IE. This is apparent as the red line is plotted below the blue one. Figure 5.31b shows a more noticeable gap than figure 5.31a between the two models. This indicates that the effect of BiS is greater for LIR8 than LIR7. From these result, it is unclear why the ending result did not yield a better performance for both problems as figure 5.31 indicates improved performance for both problems.

It was hypothesised that the increased performance seen in figure 5.31 could come at the cost of coverage and that BiS reduced exploration. The reduced exploration



(a) LIR7 **(b)** LIR8

Figure 5.32: First generation of pull phase: comparison of HV between BiS-IE and PPS-IE on LIR7 and LIR8.

would in that case result in poor end results. Figure 5.32 shows the HV metric for LIR7 and LIR8. At first glance, the low value for BiS-IE would indicate that the increased convergence comes at the cost of coverage, as a higher score indicates better coverage. This is however not necessarily the case for these two problems because of the unconstrained PF being further away from the reference point used to calculate HV, (see section 2.8.4). As a result, a higher HV in this case could mean higher coverage, but also a larger gap to the PF, depending on the populations position in the objective space. As seen from the IGD metric, the population of BiS-IE has moved further towards the constrained PF. Detecting if the improved convergence comes at the cost of coverage can not be established from HV alone, and will be further analysed in section 5.4.3.

Analysing the data gathered from BiS-IE does not provide any answers to why the convergence of the end result was not enhanced. Understanding why PPS-IE performs well is required to determine the cause. The performance of PPS-IE listed in table 5.27 shows a large difference in IGD for LIR7, LIR8 and LIR12 between the end of the push phase and start of the pull phase. The increase in IGD was measured to 50.89%, 30.76% and 11.29% respectively. These results were unexpected as there is only a single generation separating the last generation of the push phase and first generation of the pull phase for PPS-IE. An increase similar to those for LIR11, MW11 and MW9 was expected for these problems as the population in PPS-IE does not know of the feasible areas and must discover these from exploration using MOEA/D with IϵCH. The results indicate that minimal constraint handling is required to move the population out of infeasible regions and towards the constrained PF. A lower performance increase can be seen for LIR12 than LIR7 and LIR8, which suggests that the disjoint segments increase the difficulty of the problem. However, this can not be the only reason for the difference as MW9 does not have a disjoint PF and table 5.27 does not show any indication of a large leap towards the constrained PF for MW9. In an attempt to understand the cause for this behaviour, additional tests were run for these problems to gather more information. Since the traversal for PPS-IE was so high, the allowed constraint violation at the first pull generation was believed to be an important factor. A new set of experiments were run to gather this data. This data is thus not based on the experiments presented in table 5 but

a new set of 30 runs of MW9 and LIR7. These problems were selected as both have similar PF shape characteristics (see figures 5.9 and 5.10). The results from this experiment are presented in table 5.28, which shows the allowed constraint violation of the first pull phase generation.

Problem		**BiS-IE**	**PPS-IE**
MW9	MEAN	5.24e-02	6.97e+04
	STD	4.75e-02	7.06e+03
LIR7	MEAN	3.20e-02	1.00e-01
	STD	1.44e-02	6.23e-07

Table 5.28: Mean initial allowed constraint violation threshold of BiS-IE and PPS-IE on MW9 and LIR7.

Comparing the values between BiS-IE and PPS-IE gains insight into the range of constraint violation in the objective space. PPS-IE uses the maximum achieved constraint violation during the search (see section 3.2), whereas BiS-IE selects the value from the max violation within the population after the selection strategy (see section 4.2.8.3). The results indicate that the degree of constraint violation near the boundary in MW9 and LIR7 is similar. On the other hand, the maximum possible degree of constraint violation in the objective space is much greater for MW9 than of LIR7. Calculating the difference between the mean initial allowed constraint threshold for BiS-IE and PPS-IE shows the approximate range of constraint violation for the problems. The difference is 6.80e-02 for LIR7 and 6.97e+04 for MW9. The difference for MW9 is remarkably higher than LIR7. As can be seen from the SDs on MW9, the constraint violation fluctuates heavily for PPS-IE, but remains more stable for BiS-IE. From this, it is believed that the enhancements of BiS depends on the range of constraint violation throughout the search space and not just the characteristics alone.

To visualise the effect of the different initial values used by IϵCH in BiS-IE and PPS-IE, figure 5.33 shows the progression of feasible individuals within the population for MW9. PPS-IE shows little transition from a predominantly infeasible population to a feasible one, until generation 800 when the allowed constraint relaxation is set to 0. BiS enables the population to leap towards the feasible space and IϵCH is initialised with a smaller constraint relaxation. This results in a quicker transition population with predominantly feasible individuals in BiS-IE. The same thing can not be said for PPS-IE, as it struggles to reduce the allowed constraint violation threshold enough to discover feasible individuals. PPS-IE peaks to 1 at generation 800 caused by the constraint violation threshold being forced to zero.

To further support the discussion, the same comparison as table 5.27 is presented in table 5.29, but between BiS-E and PPS-E. The results support the claim that BiS enhances the convergence for some problems and that PPS-E is able make significant progress in the one generation gap between the last generation of the push phase and the first generation of the pull phase. Two runs using BiS-E and PPS-E on MW11 were extracted to further examine the difference between utilising the constraint violation near the boundary and the maximum constraint violation in the objective space. The problems were evaluated based on IGD of the end result, where the

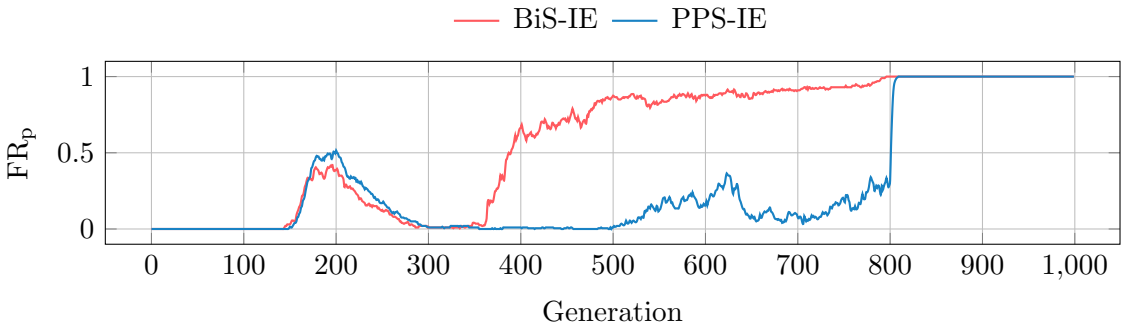


Figure 5.33: Comparison of median FR_p between BiS-IE and PPS-IE on MW9

Problem	BiS-E	PPS-E
LIR7	61.46%	53.00%
LIR8	49.91%	41.96%
LIR11	13.57%	2.62%
LIR12	18.95%	11.44%
MW9	12.61%	-0.01%
MW11	25.34%	-0.01%

Table 5.29: The percentage increase in median IGD performance between the last generation of push phase and the first generation of the pull phase for BiS-E and PPS-E.

lowest IGD was selected. These runs were selected to compare two problems that presented the best result in terms of convergence. Using a single run enables the highlighting of the generations where the models switch phases. Either IGD or HV could have been used as a selection preference, as BiS-E performed significantly better in both metrics. From this run, the progress of FR_p through the evolutionary process was plotted in figure 5.34.

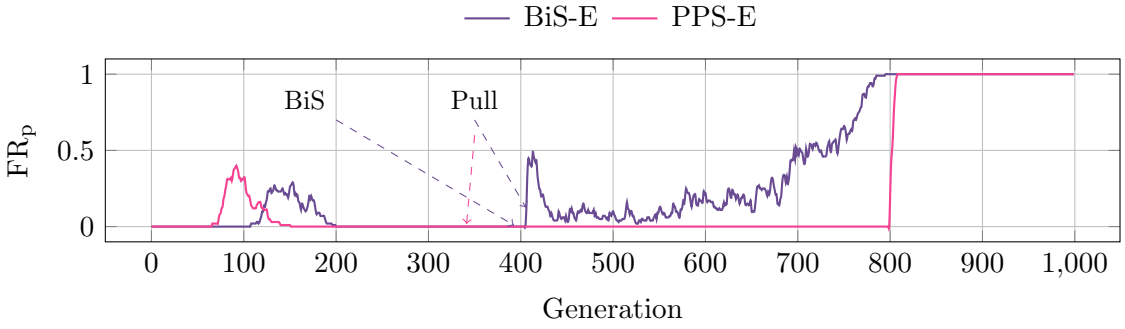


Figure 5.34: Comparison of FR_p between BiS-E and PPS-E on MW11.

To increase readability and understanding of the following discussion it is advised to examine figure 5.10c to get an overview of the objective space of MW9. As seen in figure 5.34, small spikes at generations 100 to 200 are evident for PPS-E and BiS-E. This increase in FR_p is caused by the population passing through feasible space towards the unconstrained PF. The feasible areas discovered are not overlapping with the unconstrained PF so the population moves out of this feasible space and into infeasible space towards the unconstrained PF. The two models then move the

populations through infeasible space until the end of the push phase, marked by the dotted arrows in the figure. Similarly to how IϵCH is not able to reduce the constraint relaxation fast enough for MW9 in figure 5.33, ϵCH is unable to decrease the relaxation enough for the FR_p to increase. This can be seen by the flat pink plot after the initialisation of the pull phase, highlighted by the pink arrow. The result is minimal exploration within feasible space before any constraint violation is prohibited at generation 800. The purple plot visualising the behaviour of BiS-IE shows an increase in FR_p shortly after the BiS phase. For this problem, the BiS phase lasted 10 generations and it is apparent that the inclusion of it is crucial for increasing FR_p and solving the problem, as a significant increase in performance can be seen for both BiS-IE and BiS-E in table 5.23. The use of a lower initial constraint violation for ϵCH seems to give the population a more efficient search range and it is successfully able to gradually traverse into feasible space.

5.4.2.2 Push Pull Reduced Search Space

Problem	PPS-RSS	PPS-IE	PPS-E
LIR1	55.37%	64.22%	56.22%
LIR2	60.45%	65.63%	62.12%
LIR3	54.14%	62.72%	54.23%
LIR4	56.03%	65.11%	59.80%
LIR7	65.81%	65.98%	65.76%
LIR8	51.86%	52.00%	51.93%
LIR11	9.16%	9.37%	9.14%
LIR12	36.29%	36.65%	36.65%
MW9	-5.00%	14.27%	12.30%
MW11	-3.00%	30.71%	30.11%
MW13	0.66%	-4.39%	-4.88%

Table 5.30: The percentage increase in median IGD between the end of the push and pull phase for PPS-RSS, PPS-IE and PPS-E.

Table 5.30 shows the difference between the *last* generation of the push phase and the *last* generation of the pull phase as a percentage for PPS-RSS, PPS-IE and PPS-E. Note the contrarily, tables 5.27 and 5.29 illustrate the difference between the *last* generation of the push phase and the *first* generation of the pull phase is highlighted. Again, it is clear that the behaviour of PPS-RSS resembles that of PPS-E more than PPS-IE.

PPS-RSS performs similarly to the PPS models on several problems. For these problems, the state that the population is in at the end of the push phase may affect the overall performance seen in table 5.30. Runs with low performance at the end of the push phase will likely have a larger increase in performance at the end of the pull phase than those where the performance at the end of the push phase is good. Future work should look closer into this aspect of the experiments. A possibility would be to log the results at the end of the push phase together with the performance increase at the end of the pull phase to gain more insight into the effect of RSS.

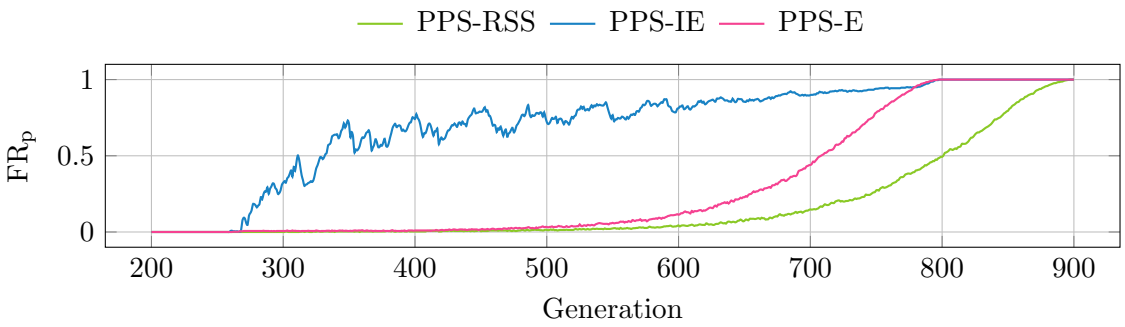


Figure 5.35: Comparison of median FR_p between PPS-RSS, PPS-IE and PPS-E on LIR7.

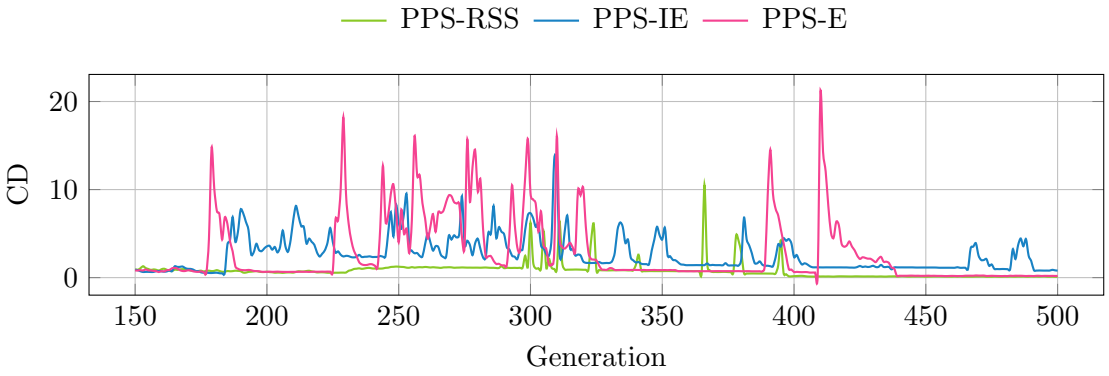


Figure 5.36: Comparison of median CD between PPS-RSS, PPS-IE and PPS-E on LIR7.

Figures 5.35 and 5.36 show the traversal of PPS-RSS, PPS-IE and PPS-E. Figure 5.35 illustrates how quick PPS-IE is to approach the constrained PF moving from the unconstrained one. PPS-RSS and PPS-E are slower, with PPS-RSS being the slowest. This is due to the CHM in PPS-RSS reducing the allowed constraint violation at a lower speed than the two other methods. Thus, PPS-RSS spends more time crossing the infeasible space between the constrained and unconstrained PF.

Figure 5.36 highlights an interesting difference between the models: PPS-RSS has more stable CD during the search than the two PPS methods. Note that the graph aggregates the CD of 30 runs, and not one specific run. In other words, the runs of PPS-IE and PPS-E do not necessarily experience such a change for each run. Still, PPS-RSS displays a more stable CD over the 30 runs. The exact reason for this is unknown, but it is hypothesised that this is due to the slow reduction in constraint relaxation. Reducing the constraint relaxation slowly may result in a controlled sweep of the search space when traversing between the unconstrained and constrained PF. Similar results were found for other problems in the test suite.

5.4.2.3 Summary

Section 5.4.2 has discussed the results from executing the experimental plan in section 5.2.3.

The BiS models have proved to introduce faster traversal during the evolutionary process for certain problems. The model did not improve the traversal for LIR1-4,

where few runs performed BiS. The results indicate that BiS performs well when the maximum possible constraint violation for a problem is large. BiS determines the initial allowed constraint violation for the CHM from the approximated boundary, which allows a better traversal between the unconstrained and constrained PF.

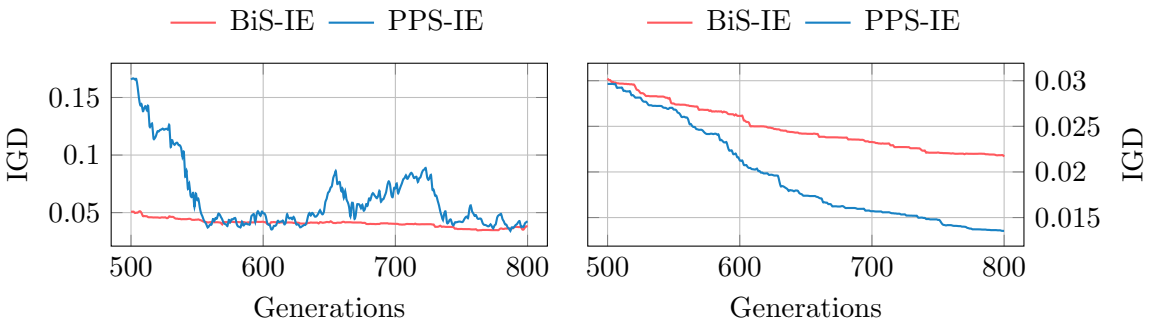
RSS has a slower traversal through the objective space than both PPS models. The slow traversal results in a more gradual and controlled evolution of the population.

5.4.3 Results Phase T4

To evaluate if Boundary Search may introduce a bias towards certain areas of the search space, IGD, CD and FR_p metrics are analysed. The metrics highlight properties of the population during the search. The development of the archive will also be presented with the population to highlight differences in their behaviour. Finally, plots from runs will be visualised to highlight certain concepts identified from the metrics.

5.4.3.1 Binary Search

For the discussion of BiS, three problems where BiS-IE performed significantly worse, significantly better and not significantly different are selected. MW5, MW9 and MW11 have been selected based on the results in table 5.21. The results of BiS-IE are selected over BiS-E as BiS is part of both models, and only one is required for this discussion. To increase readability, the graphs start at the 500th generation.

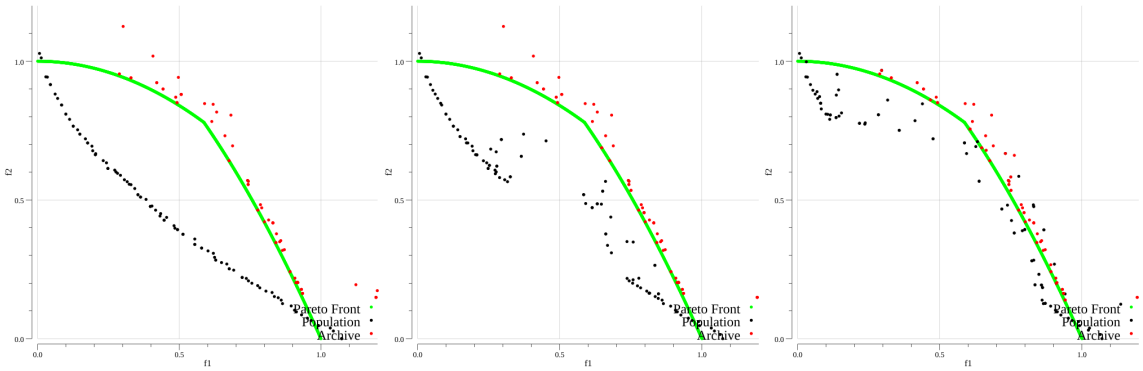


(a) Population IGD **(b)** Archive IGD

Figure 5.37: Comparison of median IGD of the population and the archive between BiS-IE and PPS-IE on MW9.

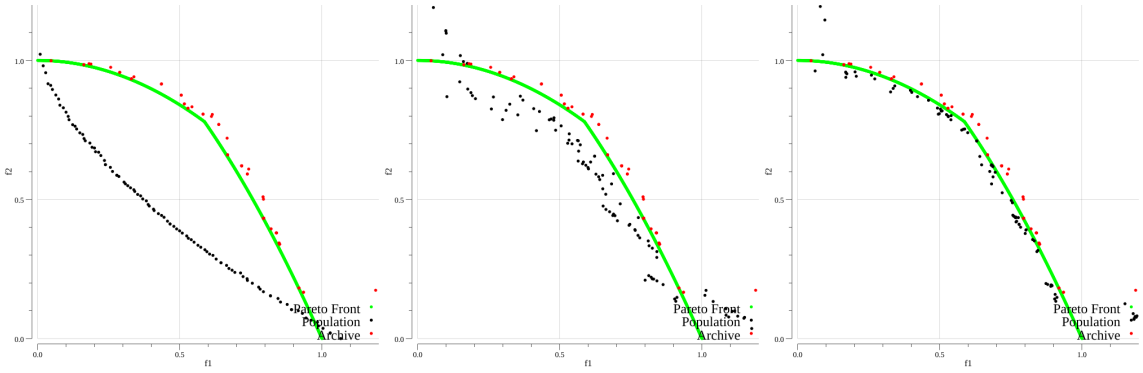
MW9 introduces a high likelihood of finding a diverse feasible set of individuals during the push phase. This can be seen from the large continuous feasible area surrounding the constrained PF in figure 5.10c. In table 5.21 it was shown that PPS-IE performed significantly better than BiS-IE on this particular problem. The problem shows an interesting trait: the population of PPS-IE is generally further away from the PF than BiS-IE. Figure 5.37a highlights this by the blue plot having a higher IGD than the red plot for the majority of the generations. The archive of PPS-IE is closer than BiS-IE to the PF and the distance is progressively reduced as seen in figure 5.37b, where the blue plot shows a lower IGD than the red plot.

Thus, even though PPS-IE is exploring the search space seemingly further away from the optimal solutions, the end result is significantly better than BiS-IE. A notable trend in the chart is the low vertical change of BiS-IE, indicating that change in the population happens at a low frequency. This may be due to the population crowding together around a small region.



(a) Generation 509. **(b)** Generation 516. **(c)** Generation 528.

Figure 5.38: PPS-IE MW9.



(a) Generation 380. **(b)** Generation 383. **(c)** Generation 386.

Figure 5.39: BiS-IE MW9.

The higher initial IGD for PPS-IE in figure 5.37a is caused by the greater distance to the PF than BiS-IE, however the distance is gradually reduced. The gradual reduction allows greater exploration of the search space and enables PPS-IE to spread the population along the PF. Figure 5.38 shows this effect over the span of 19 generations (generation 509 to 528). The population is gradually traversing from the unconstrained PF to the constrained. Figure 5.38a shows that the individuals in the middle of the population start the traversal towards the constrained PF. This is due to the constraint violation being higher in the centre of the unconstrained PF. Gradually the individuals close to the edges move closer towards the constrained PF as seen in figures 5.38b and 5.38c. Figure 5.39 shows the development of the population for BiS-IE over the span of 6 generations (generation 380 to 386). Figure 5.39a shows having converged towards the unconstrained PF. BiS is performed during the following generations and the population performs a leap towards the constrained

PF as shown in figures 5.39b and 5.39c. The population of BiS-IE reaches the constrained PF in only 6 generations, compared to PPS-IE not converging even after 19 generations.

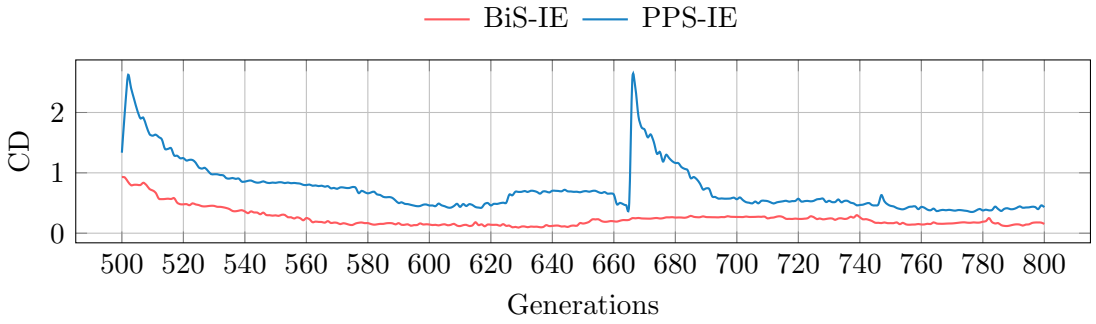


Figure 5.40: Comparison of the median CD between BiS-IE and PPS-IE on MW9.

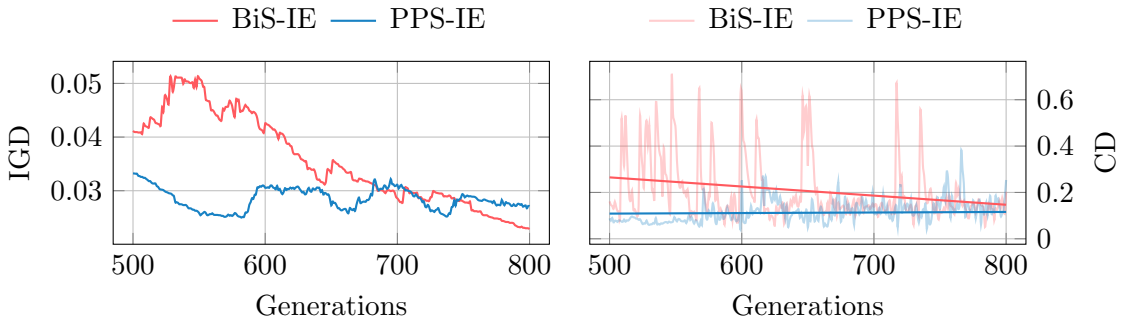
To determine if the rapid convergence on MW9 causes clustering of the population, figure 5.40 is analysed. The figure shows the progress of CD, which represents the total distance between all individuals of the population. From figure 5.37a the IGD is comparable between BiS-IE and PPS-IE around generation 600. Figure 5.40 shows that the CD is higher for PPS-IE than BiS-IE for the same generations where the IGD is comparable in figure 5.37a. This indicates that there is a higher spread within the population and the individuals thus cover a larger part of the objective space. A lower CD indicates that BiS-IE may be prone to stagnation in a local optima for this problem. This may explain the lack of improvement in IGD for BiS-IE shown in figures 5.37a and 5.37b. The reason seems to be BiS, as PPS-IE does not have the same problem. The constrained PF is continuous and therefore exploration over a larger area is required to cover it completely. BiS introduces rapid convergence towards the boundary. However, there is also less exploration along the constrained PF. The lower exploration for BiS-IE is possibly caused by the lower initial constraint violation introduced, potentially hindering the ability of the population to explore the edges of the PF.

IGD	BiS-IE	PPS-IE
MEAN	**1.3022e-02**	1.4226e-02
STD	3.1653e-03	3.0128e-03

HV	BiS-IE	PPS-IE
MEAN	**3.7836e-01**	3.7637e-01
STD	4.3910e-03	4.3443e-03

Table 5.31: IGD and HV results on MW5 from section 5.4.1.

Table 5.31 shows the IGD and HV gathered from the results of section 5.4.1, where BiS-IE proved to be slightly better than PPS-IE. However, no significant difference could be detected for either metric in table 5.21. In MW5, the two PFs are overlapping (see figure 5.10a). Figure 5.41 shows the change in IGD and CD for BiS-IE and PPS-IE when solving MW5. Trendlines are highlighted in figure 5.41b as the CD is fluctuating and difficult to read. An interesting development can be seen in figure 5.41a where PPS-IE initially is ahead in terms of IGD. During the evolutionary process, BiS-IE is able to pass PPS-IE around generation 700 and as



(a) Population IGD. **(b)** Population CD.

Figure 5.41: Comparison of median IGD of the population, median CD of the population between BiS-IE vs PPS-IE on MW5.

the populations reach generation 800, BiS-IE has achieved a better IGD than PPS-IE. Figure 5.41b shows that the CD of BiS-IE was high during the same generations as IGD was high.

It is uncertain if the same clustering effect is present in MW5 as in MW9. Displaying the potential clustering as a plot similar to figures 5.38 and 5.39 for this particular problem is difficult as it requires zooming in on the segments, which is not implemented. However, the problem contains disconnected segments, and crowding together is thus a natural reaction from any population, as the individuals aim to fit into the feasible segments of the PF.

To show an indication of BiS-IE clustering within segments, the FR_p of BiS-IE is presented in figure 5.42. The figure illustrates the number of individuals contained within feasible segments. The FR_p for BiS-IE is rising at a faster rate than PPS-IE. At generation 800, the last generation before the constraint relaxation is set to 0, the FR_p for PPS-IE was 0.79. BiS-IE was able to achieve an FR_p of 0.79 at generation 593, before dipping slightly, then reaching 0.79 again at 631. It is apparent that BiS places more individuals within feasible space early in the evolution, allowing more generations to explore the feasible segments. This could also be the reason for the early low IGD seen in PPS-IE as well as the increasing CD as the population approaches generation 800.

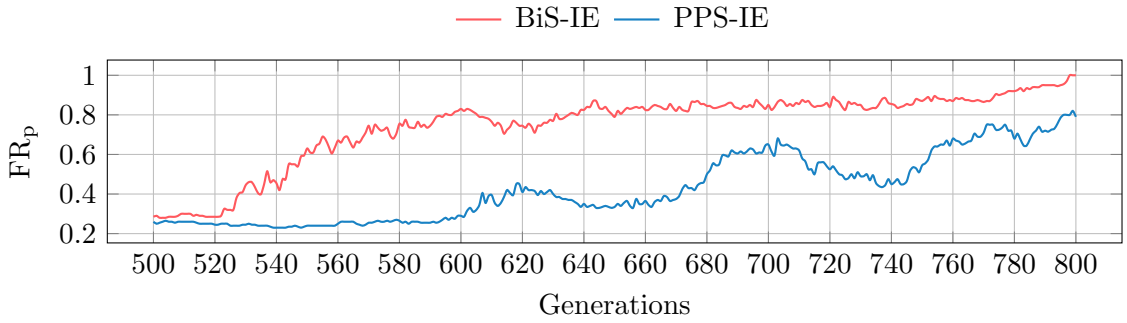
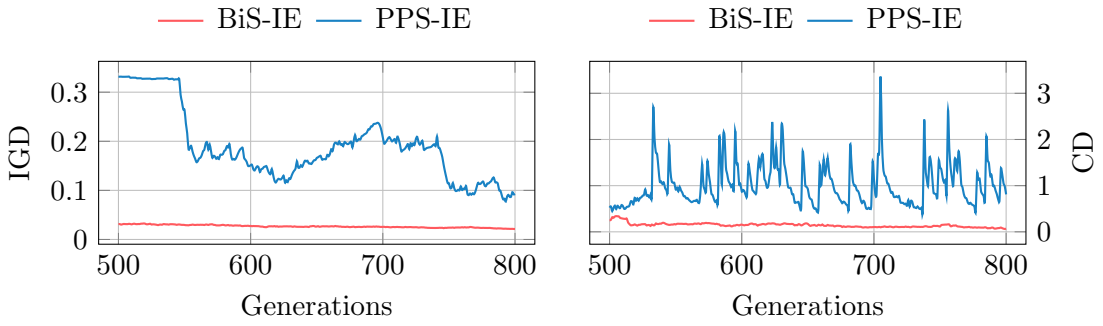


Figure 5.42: Comparison of median FR_p between BiS-IE and PPS-IE on MW5.

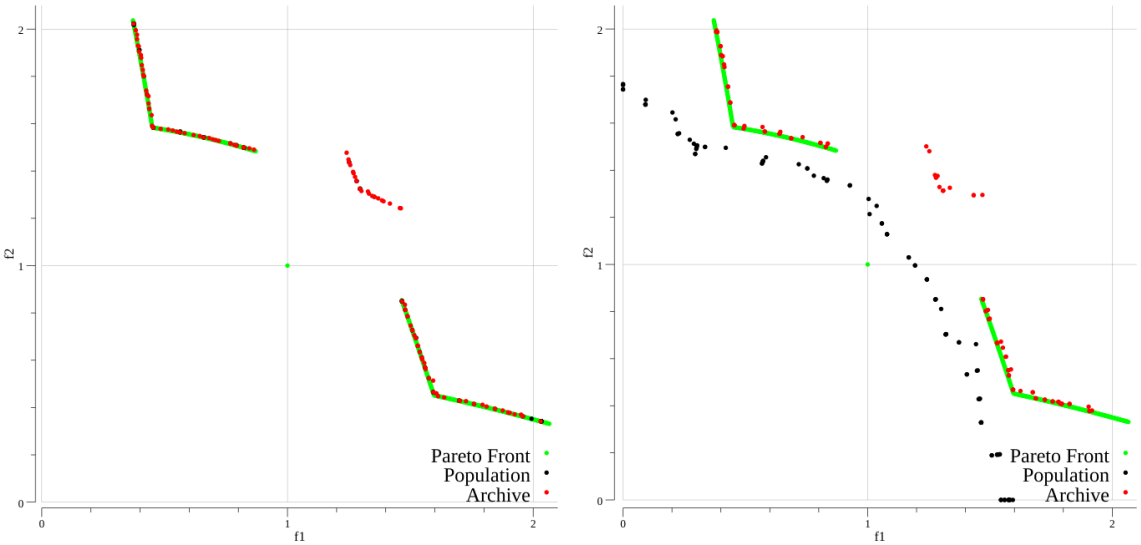
Two relatively large segments with a small feasible area in between make up the PF in MW11 (see figure 5.10e. As seen in figure 5.43a, at generation 500



(a) Population IGD. **(b)** Population CD.

Figure 5.43: Comparison of median IGD of the population and median CD of the population on MW11.

BiS-IE is ahead of PPS-IE in terms of IGD. PPS-IE has difficulties discovering both segments when traversing back from the unconstrained PF, resulting in a poor distribution among the two feasible segments.BiS on the other hand has numerous archived feasible individuals available to guide the population back to both segments. Quickly moving multiple individuals towards both of the segments increases the exploration inside the feasible areas. This can be seen particularly well when looking at the CD in figure 5.43b. The figure shows that the accumulated distance between individuals is very low, as most of the individuals are contained within feasible segments. Figure 5.44 shows this at generation 800. BiS-IE has spread its population within the feasible segments, where PPS-IE is still exploring the infeasible area.



(a) BiS-IE generation 800. **(b)** PPS-IE generation 800.

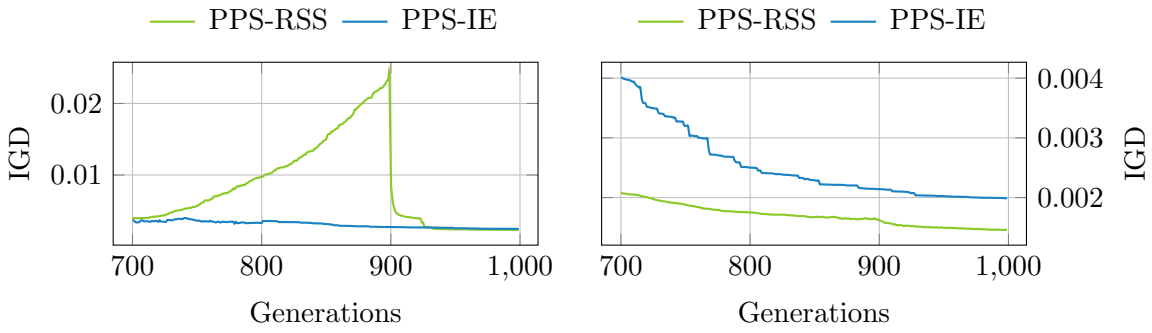
Figure 5.44: Plots of BiS-IE and PPS-IE at generation 800 for MW11.

Results on MW5, MW9 and MW11 indicate that some connection from the PF shape. The constrained PF in MW9 is continuous and requires an even spread in the population to achieve a diverse solution set. The constrained PF in MW5 is disconnected with several segments and identifying the location o all the segments

is required. Lastly, MW11 is disconnected with few segments which require a combination of the two. It may seem from this that the fewer the segments, the better the performance of BiS. There is, however, a requirement of the feasible segments not extending too far from where the individuals end up after BiS. The continuous PF seemed to be difficult to cover completely. It is believed that this is caused by a clustering effect introduced by BiS.

5.4.3.2 Reduced Search Space

Tables 5.24 and 5.25 show that PPS-RSS is significantly outperformed on nearly all problems. However, when solving LIR9 PPS-RSS showed a significant increase in convergence. To gain a better understanding of why PPS-RSS performed well on this specific problem, the behaviour of the population during the evolutionary process was analysed.



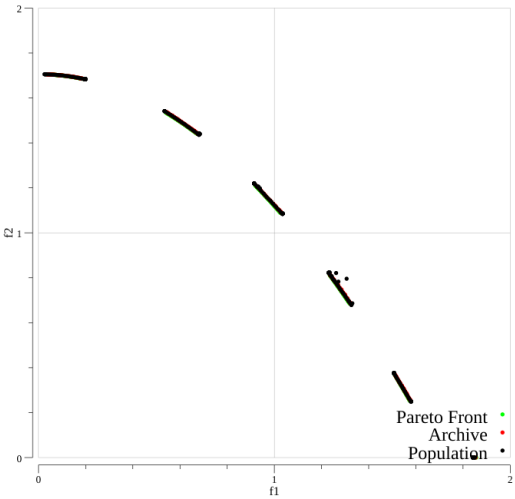
(a) Population IGD **(b)** Archive IGD

Figure 5.45: Comparison of median IGD of the population and median IGD of the archive between PPS-RSS and PPS-IE on LIR9.

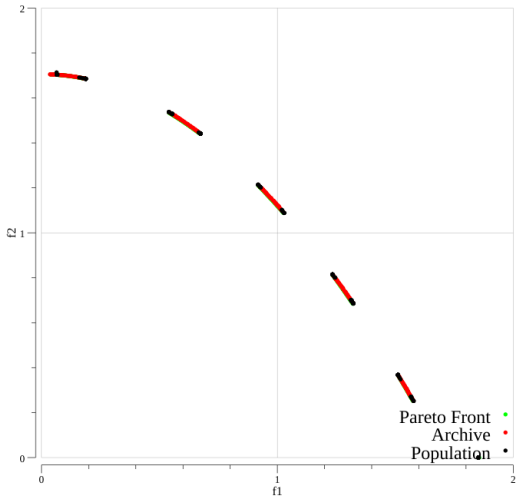
Figure 5.45 shows the IGD achieved by the population and the archive of PPS-RSS on LIR9, during the last 300 generations. Figure 5.45b shows that PPS-RSS outperforms PPS-IE, as discussed in section 5.4.1.2. More interestingly, figure 5.45a shows that the IGD for PPS-RSS increases for the last 300 generations of the run. It is not before generation 900 that the IGD decreases again.

Initially, it was hypothesised that the results in figure 5.45 were a product of a mistake in the implementation of the model. However, further analysis of these results suggests an interesting interaction between the boundary created for the feasible space and the disjoint nature of the constrained PF.

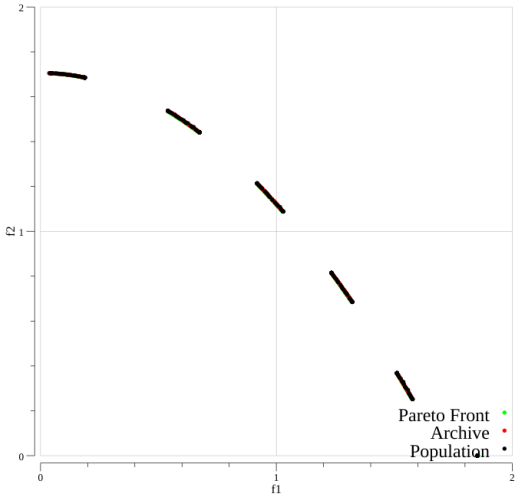
The evolutionary process of PPS-RSS is visualised in figure 5.46. Figure 5.46a shows that at generation 700 the population is covering the entire constrained PF. As the evolutionary process continues, δ_{in} is shrinking and eventually becomes so small that the population is forced out to the edges of each segment as shown in figure 5.46b. This is what creates the increase in IGD shown in figure 5.45a. It is not until generation 900 when $FESc$ number of function evaluations is reached (see section 4.4) that δ_{in} is ignored. At this point, the population begins to cover each segment of the disconnected PF fully again as shown in figure 5.46c. The



(a) Generation 700.



(b) Generation 899.

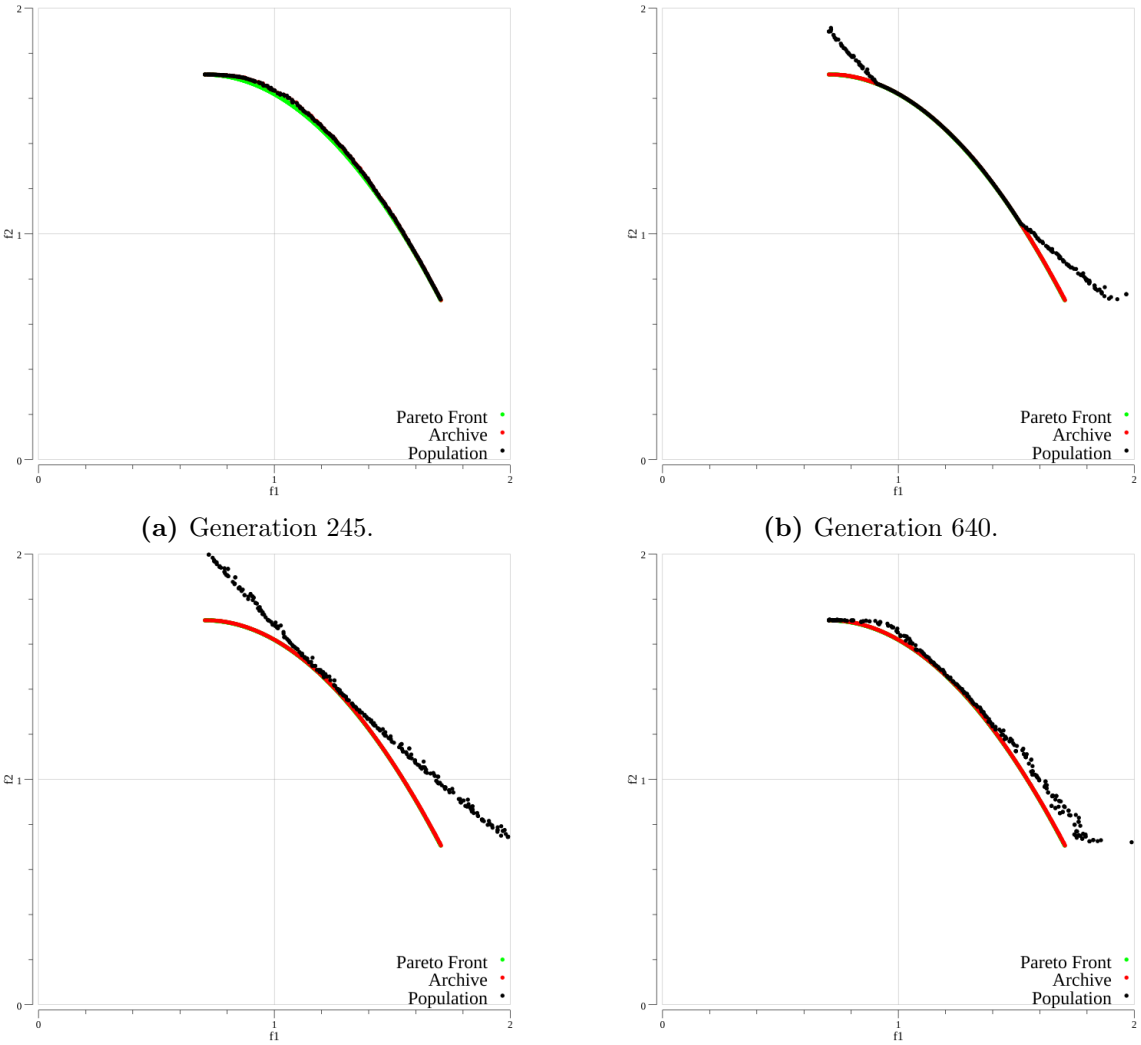


(c) Generation 920.

Figure 5.46: Boundaries shrinking over the PF on LIR9.

consequences of this behaviour is uncertain. However, figure 5.45b shows that during this event, the archive keeps improving the IGD. It is not unlikely that shrinking the search space along the tangent of the PF creates a form of scanning effect, forcing the population to search the whole segment for optimal solutions.

Similar behaviour can be seen for other benchmarks. LIR6 is another benchmark where RSS seems to enhance the performance. LIR6 does not have a disjoint PF, however, the unconstrained and constrained PF are the same. For this problem, the same interaction with the boundary for the feasible space is encountered as illustrated in figure 5.47. Figure 5.47a shows the population before δ_{in} becomes sufficiently to start pulling the population away from the PF. Following this, the population moves towards the infeasible space as shown in figures 5.47b and 5.47c. Finally, figure 5.47d shows when FES_c generations is reached and the population begins approximating the PF again. Figure 5.48 shows the IGD increasing until generation 900. This behaviour is similar to the one seen in figure 5.45a and fits with the plots in figure 5.47 where the population moves away from the constrained



(a) Generation 245.

(b) Generation 640.

(c) Generation 899.

(d) Generation 900.

Figure 5.47: Boundaries shrinking over the PF on LIR6.

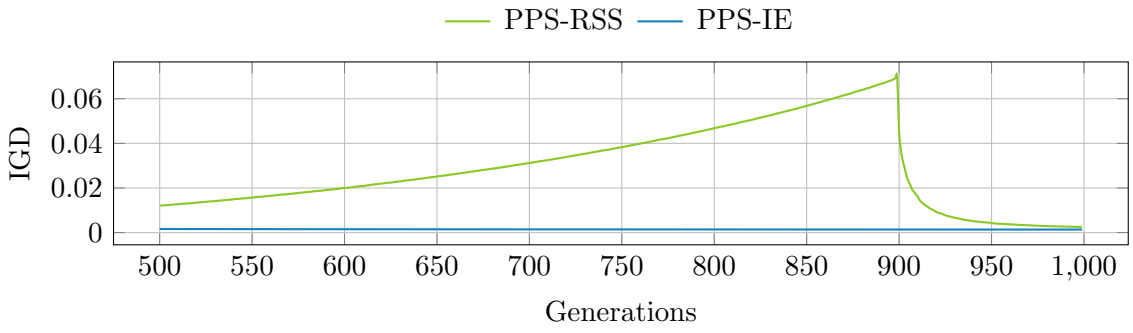


Figure 5.48: Comparison of median IGD of the population between PPS-RSS and PPS-IE on LIR6.

PF until generation 900. It can clearly be seen from this that RSS has a tendency to favour the boundary between feasible and infeasible space. However, as previously

mentioned in section 5.4.1, RSS depends largely on the constraints determined as active during ACD and the initial constraint violation used to initialise δ_{out} and δ_{in}.

5.4.3.3 Summary

Section 5.4.3 has discussed the results from executing the experimental plan described in section 5.2.4. After analysing the IGD, FR_p and CD of the population during the evolutionary process the following was found:

It is hypothesised that BiS may be more prone to stagnating in local minima. This is due to the clustering of the population after BiS has been performed. In addition, the reduced initial constraint violation used for constraint handling after BiS may reduce the ability of the population to properly explore the search space and the entire PF. BiS has shown bias towards feasible areas discovered in the push phase, thus detecting undiscovered regions is less prioritised.

RSS has shown bias towards the border between the feasible and infeasible space. If δ_{in} becomes too small, the population may be forced closer to the border between feasible and infeasible space. Thus, the population be alienated from part of, or the entire, constrained PF on problems with disjoint PFs or problems where the constrained PF is not located on the border between feasible and infeasible space.

Chapter 6

Evaluation and Conclusion

The following chapter presents a discussion and evaluation of the goal and RQs in section 6.1. Furthermore, section 6.2 presents an overview of the contributions of this thesis. Finally, future work is elaborated in section 6.3 concluding this thesis.

6.1 Evaluation

The objective of this thesis was to *Investigate how landscape information can increase the performance of PPS when solving CMOPs.* Four RQs were formulated to reach the goal. These RQs are discussed below:

RQ1 *What landscape information extracted during the evolutionary search can benefit PPS to increase convergence to and coverage of the constrained PF?*

During the course of this thesis, RQ1 was taken into account during the literature review, the design of the model and the experimental testing. From the literature review, three main approaches were investigated: surrogate models, Fitness Landscape Analysis and Boundary Search (BS). Further research focused on the use of BS. Two specific methods were evaluated during the experimental testing: BiS and RSS.

BiS exploits information regarding the best known feasible individuals in regards to Pareto dominance. The goal is to approximate the boundary of infeasible and feasible space by moving pairs of infeasible and feasible individuals closer together. The method has shown the ability to increase both convergence and coverage of the constrained PF. However, the applicability is highly dependent of the problem at hand.

The intention of the two PBPS models is to exploit the ability of BiS to quickly focus the search on the border between feasible and infeasible regions. Results show that this method is beneficial when the population has passed the constrained PF during the push phase, and found feasible individuals close to it. By reducing the number of generations used to cross infeasible regions and approximate the boundary between feasible and infeasible space, the population is allowed to explore the constrained PF for more generations. Thus, the performance in regards to IGD and HV may be improved, as was the case for LIR8. However for most problems, the

performance was neither significantly increased nor decreased by the introduction of BiS.

RSS exploits information regarding which constraints are active at the constrained PF. By focusing on these constraints and the area around their borders, the search space may be drastically reduced.

The results in section 5.4.1 show that RSS has problems performing on par with the original PPS framework. Most notably is the difference between PPS-RSS and PPS-IE. This is attributed to the difference in constraint relaxation and the fact that IϵCH allows for an increase in constraint relaxation, whilst RSS does not.

Having different approaches to the constraint relaxation complicate drawing any definitive conclusions about the usefulness of knowledge regarding the active constraints of a problem. If PPS-RSS used ϵCH (see equation (2.12)) and IϵCH (see equation (3.4)) when comparing results with PPS-IE and PPS-E respectively, it may have been easier to see the effects of focusing only on active constraints. Thus, no conclusion is drawn about the usefulness of active constraint knowledge.

RQ2 *How do different problem characteristics affect the performance of BS?*

The benchmarks used for the experimental testing have several different characteristics. This work chose to focus on the following ones: small feasible areas, disjoint PFs, and convex and concave PFs. The benchmarks exhibit other characteristics as well, however they were not the focus of this thesis. Note that when discussing the results, other characteristics such as large infeasible areas may be drawn attention to due to them possibly affecting performance.

The results from the experimental testing show that BiS performs well on problems where the unconstrained and constrained PF are separated by an infeasible region. BiS allows the population to quickly traverse the space between the two fronts. The method is however dependent on finding feasible individuals to create pairs with the individuals in the population (see section 4.2.8.1). BiS performed better on problems where the feasible area around the constrained PF was sufficiently large. The reason for this is that larger areas making it likely for several feasible individuals to be found during the push phase. Problems with small feasible areas may pose a greater challenge due to the reduced likelihood of finding feasible individuals during the push phase. The experiments show that BiS has potential to perform well, even when few feasible individuals are available for pairing. However, with small feasible areas, the likelihood of finding feasible individuals is reduced. Thus, BiS may not be executed for these problems. Therefore, BiS may not be suited for problems with small feasible regions.

BiS produced varying results on problems with disjoint PFs. The max constraint violation used as a result of performing BiS may in some cases be too low. As a result, the population is unable to fully discover all feasible segments along the PF. This was most apparent for problems with small feasible regions or problems where the disjoint parts of the PF were far away from each other. Additionally, problems where the shape of the constrained and unconstrained PFs had opposing shapes, as seen in LIR7, gave poor results.

PPS-RSS performed poorly compared to the other models on most problems. First, this was evident on problems where the unconstrained and constrained PF are

separated by a large infeasible area. The reason for the poor performance is due to the slow shrinking of constraint relaxation. The slow shrinking forces the population to use more generations traversing the infeasible regions before it may explore the constrained PF. PPS-RSS did not seem to have difficulty solving problems or excel at solving them due to small feasible regions or disjoint PFs. Rather, the experiments indicate that the reason for the poor performance is the necessity for specialised parameter tuning. The parameters of the model are required to fit the problem as the ACD method and the shrinking of constraint boundaries are highly dependent on the problem at hand. The parameter sweep in this thesis focused on finding parameters that would fit several different problems.

RQ3 *How does the use of BS affect the traversal to the constrained PF through infeasible space?*

In terms of traversing to the constrained PF through infeasible space it was shown that BiS had a positive impact on most problems. The problems where no enhanced traversal was detected, were LIR1-4, where the small feasible area was not discovered during the majority of the runs. This is therefore dependent on the size of the regions. For problems with medium and large sized feasible regions, BiS was consistently initiated and would allow BiS to produce a large leap through the infeasible region. For problems with small feasible regions, the effect of BiS is less consistent. When no feasible individuals are found during the push phase, BiS is not initiated and thus has no effect on the performance of the framework. BiS is an appropriate solution to problems with a large difference between the maximum constraint violation in the search space and the constraint violation close to the boundary. The initial allowed constraint violation determined by BiS allows ϵCH and IϵCH to start the gradual decrease of allowed constraint violation, with values gathered from the approximated boundary. This could be beneficial for problems where parameterisation of the IϵCH and ϵCH is difficult, caused by large variations in constraint violation throughout the objective space.

RSS exhibits similar traversal to PPS-E. However, RSS moves slower through the infeasible area spending several generations traversing towards the constrained PF. This slow evolution reduces the number of generations the population has at exploring the constrained PF. Despite this, it may also allow for a thorougher sweep of the landscape.

RQ4 *How can BS introduce a bias towards certain areas of the objective space?*

BiS showed indication of the population crowding together. The reduced max violation created difficulties in fully covering the PF for continuous PFs. As the disjoint PFs require the population to crowd together within the feasible segments, it is unclear if this behaviour is causing negative effects in problems with disjoint PFs. However, the results indicate that this may be the case, where problems with less segments seem to be more easily solved. From this, bias towards feasible areas discovered in the push phase is evident and detecting undiscovered segments is neglected by BiS.

RSS showed interesting behaviour due to the δ_{in} becoming too small for some problems. This led to the population being forced towards the border between

feasible and infeasible space, ignoring the fact that the PF and the border may not overlap. PPS-RSS was able to perform well on the problems where this occurred, but this was largely a result of the population locating the constrained PF before being alienated from it. If the population locates the constrained PF before δ_{in} becomes too small, this bias towards the border between infeasible and feasible space may reduce performance.

6.2 Contributions

From this work, the use of landscape information has successfully been incorporated into PPS to create three new algorithmic models. These models have proved to be able to solve several CMOPs. Also, they show the ability to perform competitively with the original PPS models:PPS-IE and PPS-E.

Two types of landscape information have been utilised to perform two different forms of Boundary Search: the location of best feasible individuals in terms of Pareto dominance to perform Binary Search (BiS) and the location of active constraints to utilise the Reduced Search Space (RSS) method.

The use of BiS has shown to be advantageous when solving problems where the unconstrained and constrained PF are separated by infeasible regions. Using the last known feasible individuals allows the population to perform a leap in the search space, reducing the generations spent traversing infeasible regions.

The use of RSS has shown to perform similarly to PPS-E. Focusing on active constraints allows the search space to be reduced, however there seems to be little performance increase using this information.

In addition to analysing the performance change by implementing these two BS methods, it was also found that:

- BiS improves convergence of the population for problems where the constrained and unconstrained PFs are not overlapping.
- BiS may not be suitable for problems with small feasible regions due to the low likelihood of BiS being performed.
- Using the max constraint violation of the population after BiS to initialise IϵCH may allow the population to better explore the PF. However, for disjoint PFs this may reduce the exploration capabilities of the population, alienating it from detecting undiscovered segments. For continuous PFs, this may reduce the coverage of the PF.
- The slow reduction in constraint relaxation may enable a thorough sweep of the search space as the population moves towards feasible space.
- RSS may alienate the population from the constrained PF if it is not located on the border between feasible and infeasible space.

6.3 Future Work

The following section presents potential future work used to further research the use of landscape information in solving CMOPs.

6.3.1 Surrogate Model

During the structured literature review, surrogate models were researched. However, after the literature review surrogate models were not implemented into the model in favour of implementing two BS methods. Surrogate models have shown their usefulness in increasing efficiency [Díaz-Manríquez et al., 2016]. The models showed difficulties finding good results on the MW problems. Implementing a surrogate model to extract underlying information about the problem could prove beneficial when trying to solve inherently difficult objective functions. By enhancing efficiency, the computational cost for the same number of generations may be reduced and thereby enabling a longer search by increasing the number of generations possibly enhancing performance. The PPS framework is suited for this by building the surrogate based on information gathered during the push phase, then utilising the model fully during the pull phase. The research should look at the different classifications of surrogate models, defined by Díaz-Manríquez et al. [2016] and how they perform together with PPS.

6.3.2 Fitness Landscape Analysis

FLA was also explored during the literature review. There has been little research on the use of FLA during the run of an algorithm. The common approach is to perform two separate runs, using the first one to accumulate information which may be used for the second run. Due to the majority of metaheuristic-approaches aiming to solve the problem in one run, and the complexity of FLA, FLA has not been extensively used in the field [Malan and Engelbrecht, 2013]. The biphasic nature of PPS allows the population to explore the landscape unhindered during the push phase. RQ1 has been addressed by looking at the use of landscape information regarding the location of the last known feasible space (BiS) and which constraints are active (RSS). However, this topic should be further analysed. Analysing how the biphasic nature of PPS can be combined with FLA to produce a landscape analysis exploited in the pull phase is an interesting topic that should be researched. This can for instance be used to focus the search in specific areas, or to decide which CHM should be used to better fit the problem at hand when entering the pull phase. Analysing the use of FLA in conjunction with PPS may unveil new ways to use FLA as part of the search directly and not as a separate search of the landscape before attempting to solve the problem.

6.3.3 Combining Different Landscape Information

This thesis has looked at the use of two different types of landscape information: the location of feasible regions and the location of active constraints. However, different landscape information has not been combined in this work. A possible approach

building on the work of this thesis is to use the archive of feasible individuals when performing ACD. For instance, instead of using a predefined value for Val, the distance between the population and the archive could be used instead. Another approach could be to look at the distance from the archive to the constraints, rather than the working population. Another possibility is to incorporate active constraints when pairing individuals, grouping them into feasible and infeasible based on the violation of only active constraints and not all constraints. Also, other landscape information should be identified as potential additions to the model and used in combination.

6.3.4 Parameter Setting

This work has used one set of parameters for all benchmarks tested, with the exception of LIR1 to LIR4. This was due to the large difference in Val needed to identify an active constraint. It would be interesting to see how the two BS methods perform when the parameters have been tuned for each problem. Specifically, for RSS, the preferred value of Val is highly problem dependent. The location of the population when the search stagnates in the push phase may vary from problem to problem, and thus the appropriate Val value will change. Investigating the effect of RSS with tuned parameters could provide greater insight into the potential of the method.

6.3.5 Additional Experiments

During the experiments of this work, new questions continuously emerged from the results of previous experiments. Sections 5.4.2 and 5.4.3 shows that BiS-IE is able to increase the convergence in less generations than PPS-IE for certain problems. However the results in section 5.4.1 do not show that BiS-IE achieves better end results. Thus more experiments with reduced function evaluations should be conducted to see if there exists a higher difference between the two models.

Bibliography

Back, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*.

Bandaru, S., Ng, A. H., and Deb, K. (2017). Data mining methods for knowledge discovery in multi-objective optimization: Part A - Survey.

Baug, E., Norstein, A., and Haddow, P. C. (2019). MAIM: A Novel Island Based Evolutionary Classification Algorithm. Technical report.

Bonyadi, M. R. and Michalewicz, Z. (2014). On the edge of feasibility: A case study of the particle swarm optimizer. In *Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014*, pages 3059–3066. Institute of Electrical and Electronics Engineers Inc.

Bonyadi, M. R., Michalewicz, Z., and Barone, L. (2013). The travelling thief problem: The first step in the transition from theoretical problems to realistic problems. In *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, pages 1037–1044.

Bosman, A. S., Engelbrecht, A., and Helbig, M. (2017). Search space boundaries in neural network error landscape analysis. In *2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016*. Institute of Electrical and Electronics Engineers Inc.

Burke, E. K., Hyde, M. R., Kendall, G., and Woodward, J. (2007). Automatic heuristic generation with genetic programming: Evolving a jack-of-all-trades or a master of one. In *Proceedings of GECCO 2007: Genetic and Evolutionary Computation Conference*, pages 1559–1565, New York, New York, USA. ACM Press.

Calborean, H. A. and Vinţan, L. (2011). Multi-Objective Optimization of Advanced Computer Architectures using Domain-Knowledge. Technical report.

Coello, C., Lamont, G., and Veldhuizen, D. V. (2007). *Evolutionary algorithms for solving multi-objective problems*.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):311–338.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. Technical Report 2.

Del Ser, J., Osaba, E., Molina, D., Yang, X. S., Salcedo-Sanz, S., Camacho, D., Das, S., Suganthan, P. N., Coello Coello, C. A., and Herrera, F. (2019). Bio-inspired computation: Where we stand and what's next. *Swarm and Evolutionary Computation*, 48:220–250.

Díaz-Manríquez, A., Toscano, G., Barron-Zambrano, J. H., and Tello-Leal, E. (2016). A review of surrogate assisted multiobjective evolutionary algorithms.

Fan, Z. (2017). Lir testsuite source code. `http://imagelab.stu.edu.cn/Content.aspx?type=content&Content_ID=238`.

Fan, Z., Li, W., Cai, X., Huang, H., Fang, Y., You, Y., Mo, J., Wei, C., and Goodman, E. (2019a). An improved epsilon constraint-handling method in MOEA/D for CMOPs with large infeasible regions. *Soft Computing*, 23(23):12491–12510.

Fan, Z., Li, W., Cai, X., Li, H., Wei, C., Zhang, Q., Deb, K., and Goodman, E. (2019b). Push and pull search for solving constrained multi-objective optimization problems. *Swarm and Evolutionary Computation*, 44:665–679.

Handoko, S. D., Kwoh, C. K., and Ong, Y. S. (2010). Feasibility structure modeling: An effective chaperone for constrained memetic algorithms. *IEEE Transactions on Evolutionary Computation*, 14(5):740–758.

Jahr, R., Calborean, H., Vintan, L., and Ungerer, T. (2012). Boosting design space explorations with existing or automatically learned knowledge. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7201 LNCS, pages 221–235.

Jain, H. and Deb, K. (2014). An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, Part II: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, 18(4):602–622.

Jevne, H. K., Haddow, P. C., and Gaivoronski, A. A. (2012). Evolving constrained mean-VaR efficient frontiers. In *2012 IEEE Congress on Evolutionary Computation, CEC 2012*.

Leguizamón, G. and Coello, C. C. (2009). Boundary search for constrained numerical optimization problems. *Studies in Computational Intelligence*, 198:25–49.

Leguizamón, G. and Coello Coello, C. A. (2009). Boundary search for constrained numerical optimization problems with an algorithm inspired by the ant colony metaphor. *IEEE Transactions on Evolutionary Computation*, 13(2):350–368.

Li, J. P., Wang, Y., Yang, S., and Cai, Z. (2016). A comparative study of constraint-handling techniques in evolutionary constrained multiobjective optimization. In *2016 IEEE Congress on Evolutionary Computation, CEC 2016*, pages 4175–4182. Institute of Electrical and Electronics Engineers Inc.

Lim, D., Ong, Y. S., Gupta, A., Goh, C. K., and Dutta, P. S. (2016). Towards a new Praxis in optinformatics targeting knowledge re-use in evolutionary computation: simultaneous problem learning and optimization. *Evolutionary Intelligence*, 9(4):203–220.

Liu, H. L., Chen, L., Deb, K., and Goodman, E. D. (2017). Investigating the effect of imbalance between convergence and diversity in evolutionary multiobjective algorithms. *IEEE Transactions on Evolutionary Computation*, 21(3):408–425.

Liu, Z. Z. and Wang, Y. (2019). Handling constrained multiobjective optimization problems with constraints in both the decision and objective spaces. *IEEE Transactions on Evolutionary Computation*, 23(5):870–884.

Ma, Z. and Wang, Y. (2019). Evolutionary constrained multiobjective optimization: Test suite construction and performance comparisons. *IEEE Transactions on Evolutionary Computation*, 23(6):972–986.

Malan, K. M. and Engelbrecht, A. P. (2013). A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences*, 241:148–163.

Metkar, S. J. and Kulkarni, A. J. (2014). Boundary Searching Genetic Algorithm: A Multi-objective Approach for Constrained Problems. In *Advances in Intelligent Systems and Computing*, volume 247, pages 269–276. Springer Verlag.

Monsef, H., Naghashzadegan, M., Jamali, A., and Farmani, R. (2019). Comparison of evolutionary multi objective optimization algorithms in optimum design of water distribution network. *Ain Shams Engineering Journal*, 10(1):103–111.

Neri, F. and Tirronen, V. (2010). Recent advances in differential evolution: A survey and experimental analysis. *Artificial Intelligence Review*, 33(1-2):61–106.

Paraskevopoulos, D. C., Laporte, G., Repoussis, P. P., and Tarantilis, C. D. (2017). Resource constrained routing and scheduling: Review and research prospects.

Pilat, M. and Neruda, R. (2011). ASM-MOMA: Multiobjective memetic algorithm with aggregate surrogate model. In *2011 IEEE Congress of Evolutionary Computation, CEC 2011*, pages 1202–1208.

Pitzer, E. and Affenzeller, M. (2012). A Comprehensive Survey on Fitness Landscape Analysis. pages 161–191. Springer, Berlin, Heidelberg.

Roald, L. A. and Molzahn, D. K. (2019). Implied Constraint Satisfaction in Power System optimization: The Impacts of Load Variations. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing, Allerton 2019*, pages 308–315. Institute of Electrical and Electronics Engineers Inc.

Rosales-Perez, A., Coello, C. A., Gonzalez, J. A., Reyes-Garcia, C. A., and Escalante, H. J. (2013). A hybrid surrogate-based approach for evolutionary multiobjective optimization. In *2013 IEEE Congress on Evolutionary Computation, CEC 2013*, pages 2548–2555.

Sallam, K. M., Sarker, R. A., and Essam, D. L. (2017). Reduced search space mechanism for solving constrained optimization problems. *Engineering Applications of Artificial Intelligence*, 65:147–158.

Shi, L. and Rasheed, K. (2010). A Survey of Fitness Approximation Methods Applied in Evolutionary Algorithms. pages 3–28. Springer, Berlin, Heidelberg.

Takahama, T. and Sakai, S. (2005). Constrained optimization by ϵ constrained particle swarm optimizer with ϵ-level control. In *Advances in Soft Computing*, number AISC, pages 1019–1029.

Talukder, A. K. A., Deb, K., and Rahnamayan, S. (2016). Maintaining diversity in the bounded pareto-set: A case of opposition based solution generation scheme. In *GECCO 2016 Companion - Proceedings of the 2016 Genetic and Evolutionary Computation Conference*, pages 945–951. Association for Computing Machinery, Inc.

Tanabe, R. and Oyama, A. (2017). A note on constrained multi-objective optimization benchmark problems. In *2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings*, pages 1127–1134. Institute of Electrical and Electronics Engineers Inc.

Tanweer, M. R., Suresh, S., and Sundararajan, N. (2016). Dynamic mentoring and self-regulation based particle swarm optimization algorithm for solving complex real-world optimization problems. *Information Sciences*, 326:1–24.

Van Aardt, W. A., Bosman, A. S., and Malan, K. M. (2017). Characterising neutrality in neural network error landscapes. In *2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings*, pages 1374–1381. Institute of Electrical and Electronics Engineers Inc.

Wang, M., Li, B., Zhang, G., and Yao, X. (2018). Population Evolvability: Dynamic Fitness Landscape Analysis for Population-Based Metaheuristic Algorithms. *IEEE Transactions on Evolutionary Computation*, 22(4):550–563.

Wang, Y., Liu, Z.-Z., Ma, Z., and Wang, B.-C. (2020). Wcci2020 cmop competition. `https://wcci2020.org/competitions/`.

Woldesenbet, Y., . . . , G. Y. I. T. o., and 2009, u. (2009). Constraint handling in multiobjective evolutionary optimization. *ieeexplore.ieee.org*.

Zhang, Q. and Li, H. (2007). MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731.

Zitzler, E. and Thiele, L. (1998). Parallel Problem Solving from Nature. Technical report.

Appendices

.1 Appendix A

Problem		BiS-IE	PPS-IE	BiS-E	PPS-E	PPS-RSS
LIR1	MEAN	1.1045e-02	**8.8040e-03**	8.1249e-02	1.0060e-01	1.1054e-01
	STD	7.9447e-03	4.1645e-03	5.9163e-02	4.0359e-02	4.4406e-02
LIR2	MEAN	5.4212e-03	**5.0375e-03**	7.4021e-02	5.8354e-02	6.5206e-02
	STD	2.8561e-03	1.2346e-03	5.6062e-02	2.8691e-02	2.6475e-02
LIR3	MEAN	2.5340e-02	**7.5015e-03**	1.1112e-01	1.2076e-01	1.3406e-01
	STD	6.8518e-02	4.0834e-03	5.1968e-02	5.1519e-02	6.9498e-02
LIR4	MEAN	2.3902e-02	**3.9279e-03**	8.4656e-02	7.8273e-02	1.0115e-01
	STD	6.7865e-02	1.5441e-03	5.0944e-02	3.7611e-02	4.2529e-02
LIR5	MEAN	1.5674e-03	**1.5398e-03**	1.5667e-03	1.5429e-03	1.9538e-03
	STD	6.2003e-05	4.5873e-05	4.3362e-05	4.8351e-05	7.4783e-05
LIR6	MEAN	2.1965e-03	2.1795e-03	2.2184e-03	2.2046e-03	**2.1527e-03**
	STD	1.2423e-04	1.3155e-04	1.0416e-04	9.0404e-05	2.0067e-04
LIR7	MEAN	3.1774e-03	**2.9001e-03**	3.2108e-03	3.0568e-03	3.1797e-03
	STD	1.1356e-03	7.8821e-05	7.5886e-04	1.4729e-04	1.0240e-04
LIR8	MEAN	**2.7237e-03**	2.7605e-03	2.8538e-03	2.9574e-03	3.1135e-03
	STD	6.3532e-05	5.7866e-05	5.3935e-05	7.1103e-05	9.0984e-05
LIR9	MEAN	2.9368e-02	2.3036e-03	1.7827e-03	1.7474e-03	**1.4681e-03**
	STD	9.2398e-02	7.5685e-04	5.3131e-05	5.1536e-05	1.0368e-04
LIR10	MEAN	1.9999e-03	1.9834e-03	1.9797e-03	**1.9611e-03**	2.0142e-03
	STD	7.4514e-05	6.0702e-05	7.4345e-05	6.8928e-05	4.7319e-05
LIR11	MEAN	1.0355e-02	1.0262e-02	4.9267e-03	**3.8463e-03**	4.8417e-03
	STD	2.1019e-02	2.8192e-02	2.7023e-03	2.0510e-03	3.3240e-03
LIR12	MEAN	9.2983e-02	4.7764e-02	2.8288e-03	**2.8018e-03**	2.9553e-03
	STD	5.5371e-02	6.0262e-02	1.0025e-04	1.0748e-04	9.9693e-04

Table 1: IGD results of BiS-IE, PPS-IE, BiS-E, PPS-E and PPS-RSS on LIR problems. Best performance is highlighted for each problem.

Problem		BiS-IE	PPS-IE	BiS-E	PPS-E	PPS-RSS
LIR1	MEAN	6.4321e-01	**6.4678e-01**	5.6248e-01	5.4716e-01	5.3873e-01
	STD	1.6697e-02	2.7374e-03	5.8156e-02	3.8310e-02	4.3014e-02
LIR2	MEAN	9.8076e-01	**9.8116e-01**	8.8204e-01	9.0986e-01	8.9592e-01
	STD	3.6760e-03	1.5973e-03	8.5820e-02	3.7632e-02	3.9486e-02
LIR3	MEAN	5.2058e-01	**5.3681e-01**	4.4402e-01	4.3296e-01	4.2586e-01
	STD	6.4708e-02	2.9635e-03	4.9291e-02	4.5763e-02	5.9883e-02
LIR4	MEAN	7.5277e-01	**7.7470e-01**	6.7843e-01	6.8531e-01	6.4851e-01
	STD	7.5395e-02	1.8649e-03	5.8162e-02	5.1531e-02	6.5302e-02
LIR5	MEAN	1.0340e+00	**1.0340e+00**	1.0340e+00	1.0340e+00	1.0332e+00
	STD	2.0930e-04	1.8233e-04	1.7173e-04	2.7266e-04	2.0096e-04
LIR6	MEAN	7.0103e-01	7.0112e-01	7.0108e-01	**7.0113e-01**	6.9985e-01
	STD	2.0075e-04	1.5016e-04	1.1829e-04	1.5522e-04	5.6906e-04
LIR7	MEAN	2.0368e+00	**2.0402e+00**	2.0370e+00	2.0396e+00	2.0392e+00
	STD	8.3316e-03	1.5486e-03	6.7472e-03	1.7812e-03	1.2675e-03
LIR8	MEAN	2.0410e+00	**2.0412e+00**	2.0407e+00	2.0402e+00	2.0399e+00
	STD	7.6797e-04	6.3639e-04	5.9258e-04	6.3808e-04	5.5204e-04
LIR9	MEAN	2.7394e+00	2.7793e+00	2.7825e+00	**2.7827e+00**	2.7826e+00
	STD	1.3591e-01	4.0266e-03	2.2780e-04	2.2968e-04	9.2848e-04
LIR10	MEAN	2.5562e+00	2.5562e+00	2.5562e+00	**2.5562e+00**	2.5559e+00
	STD	2.3280e-04	1.9190e-04	2.2572e-04	2.2732e-04	1.4858e-04
LIR11	MEAN	3.4283e+00	3.4241e+00	3.4451e+00	**3.4451e+00**	3.4442e+00
	STD	5.0879e-02	8.8182e-02	2.9840e-04	2.7036e-04	4.6463e-03
LIR12	MEAN	4.0776e+00	4.2165e+00	**4.3543e+00**	4.3543e+00	4.3529e+00
	STD	1.6975e-01	1.8410e-01	5.3795e-05	5.2563e-05	7.0742e-03

Table 2: HV results of BiS-IE, PPS-IE, BiS-E, PPS-E and PPS-RSS on LIR problems. Best performance is highlighted for each problem.

Problem		BiS-IE	PPS-IE	BiS-E	PPS-E	PPS-RSS
MW5	MEAN	**1.3022e-02**	1.4226e-02	1.3678e-02	1.5440e-02	2.8637e-02
	STD	3.1653e-03	3.0128e-03	5.0086e-03	3.0914e-03	5.4167e-03
MW6	MEAN	-	**5.7230e-01**	-	-	-
	STD	-	1.6805e-01	-	-	-
MW10	MEAN	-	-	-	-	-
	STD	-	-	-	-	-
MW9	MEAN	2.0151e-01	1.9068e-01	**1.8612e-01**	2.6728e-01	2.8065e-01
	STD	5.1044e-03	3.1373e-03	4.2815e-03	6.0034e-03	9.7723e-03
MW11	MEAN	**9.2972e-03**	1.1489e-02	9.7784e-03	1.7721e-02	1.2079e-01
	STD	9.0567e-04	1.4772e-03	8.8466e-04	6.8115e-03	1.0526e-01
MW13	MEAN	2.0221e-02	**1.2308e-02**	1.3782e-02	2.1423e-02	3.8627e-02
	STD	6.8724e-02	5.0990e-02	3.0443e-02	2.9623e-01	1.3489e-01

Table 3: IGD results of BiS-IE, PPS-IE, BiS-E, PPS-E and PPS-RSS on MW problems. Best performance is highlighted for each problem.

Problem		**BiS-IE**	**PPS-IE**	**BiS-E**	**PPS-E**	**PPS-RSS**
MW5	MEAN	**3.7836e-01**	3.7637e-01	3.7673e-01	3.7457e-01	3.5261e-01
	STD	4.3910e-03	4.3443e-03	7.6587e-03	4.4631e-03	8.1659e-03
MW6	MEAN	-	-	-	-	-
	STD	-	-	-	-	-
MW10	MEAN	-	-	-	-	-
	STD	-	-	-	-	-
MW9	MEAN	4.5257e-01	**4.6663e-01**	4.6161e-01	4.5543e-01	4.3263e-01
	STD	7.4792e-03	6.4746e-03	6.7706e-03	7.4763e-03	9.8116e-03
MW11	MEAN	**2.2725e+00**	2.2652e+00	2.2710e+00	2.2539e+00	2.0251e+00
	STD	2.1890e-03	5.2507e-03	2.4293e-03	1.4626e-02	1.7317e-01
MW13	MEAN	2.6029e+00	2.6424e+00	**2.6600e+00**	-	2.3439e+00
	STD	2.6198e-01	2.0884e-01	1.2866e-01	-	3.0052e-01

Table 4: HV results of BiS-IE, PPS-IE, BiS-E, PPS-E and PPS-RSS on MW problems. Best performance is highlighted for each problem.

.2 Appendix B

Problem	State	BiS-E	BiS-IE	PPS-E	PPS-IE	PPS-RSS
	Push End	6.7853e-01	6.7953e-01	6.7994e-01	6.7858e-01	6.7577e-01
LIR1	Pull Start	6.7766e-01	6.7963e-01	6.7997e-01	6.7883e-01	6.7608e-01
	Pull End	7.7875e-02	3.7187e-02	8.5589e-02	3.8286e-02	3.0488e-01
	Push End	6.8200e-01	6.8159e-01	6.8219e-01	6.8215e-01	6.8173e-01
LIR2	Pull Start	6.8204e-01	6.8169e-01	6.8223e-01	6.8220e-01	6.8185e-01
	Pull End	8.9656e-02	2.8349e-02	7.6180e-02	2.8225e-02	3.4103e-01
	Push End	6.7734e-01	6.7788e-01	6.7651e-01	6.7624e-01	6.7806e-01
LIR3	Pull Start	6.7755e-01	6.7798e-01	6.7675e-01	6.7637e-01	6.7833e-01
	Pull End	1.2980e-01	6.2230e-02	1.1941e-01	5.0824e-02	6.4845e-01
	Push End	6.8090e-01	6.8041e-01	6.8056e-01	6.8078e-01	6.8096e-01
LIR4	Pull Start	6.8097e-01	6.8040e-01	6.8071e-01	6.8082e-01	6.8096e-01
	Pull End	8.6667e-02	2.8982e-02	7.8573e-02	3.8150e-02	6.6647e-01
	Push End	6.6196e-01	6.6274e-01	6.6038e-01	6.6163e-01	6.6220e-01
LIR7	Pull Start	4.9537e-02	4.2009e-02	1.4339e-01	1.5201e-01	1.4412e-01
	Pull End	3.3094e-03	3.0336e-03	3.3380e-03	3.1632e-03	3.8119e-03
	Push End	5.2268e-01	5.2216e-01	5.2238e-01	5.2290e-01	5.2349e-01
LIR8	Pull Start	2.2378e-02	3.0526e-02	1.1147e-01	2.1388e-01	9.8379e-02
	Pull End	3.0673e-03	2.8666e-03	3.2346e-03	3.0331e-03	3.6513e-03
	Push End	1.7694e-01	1.7563e-01	1.7943e-01	1.7466e-01	1.8158e-01
LIR11	Pull Start	8.8607e-02	5.6299e-02	1.4646e-01	1.7465e-01	1.8212e-01
	Pull End	8.6606e-02	8.6775e-02	8.6572e-02	8.6738e-02	8.8061e-02
	Push End	3.7148e-01	3.7179e-01	3.7051e-01	3.6650e-01	3.7055e-01
LIR12	Pull Start	1.8247e-01	1.8077e-01	2.5000e-01	2.4447e-01	3.7016e-01
	Pull End	3.7242e-03	4.0594e-03	3.8166e-03	3.8696e-03	3.3414e-03
	Push End	1.3710e-01	1.4017e-01	1.3853e-01	1.3514e-01	1.3496e-01
MW13	Pull Start	2.7885e-01	2.7022e-01	1.3825e-01	1.3526e-01	1.3497e-01
	Pull End	1.7345e-01	1.7188e-01	1.8038e-01	1.7615e-01	1.2980e-01
	Push End	1.6382e-01	1.7627e-01	1.8154e-01	1.7366e-01	1.8280e-01
MW9	Pull Start	4.0312e-02	4.0395e-02	1.8252e-01	1.7411e-01	1.8332e-01
	Pull End	2.7909e-02	2.9475e-02	5.0248e-02	3.7414e-02	7.2069e-02
	Push End	3.2558e-01	3.2148e-01	3.2131e-01	3.2250e-01	3.2098e-01
MW11	Pull Start	7.5224e-02	7.1553e-02	3.2199e-01	3.2262e-01	3.2118e-01
	Pull End	1.5761e-02	1.4872e-02	1.7739e-02	1.7568e-02	3.4906e-01

Table 5: Median IGD values from end of push phase, start of pull phase and end of pull phase. High difference between start and end of pull shows great traversal between constrained PF and unconstrained PF.

Marius Aarsnes & Erling Wisløff

Informed Push & Pull Search for Solving Constrained Multi-Objective Optimisation

NTNU
Norwegian University of
Science and Technology