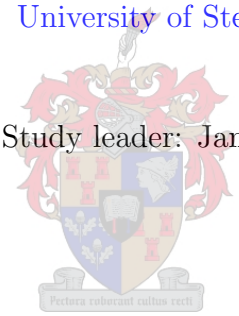
A comparative study on the value of accounting for possible relationships between decision variables when solving multi-objective problems

Esmarie Scholtz

Department of Industrial Engineering

University of Stellenbosch



Study leader: James Bekker

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering in the Faculty of Engineering at Stellenbosch University.

M. Eng (Research) Industrial

March 2014

In loving memory of my mother,
who would have been so incredibly proud.

Acknowledgements

I would like to acknowledge James Bekker for his patience, encouragement and sense of humour. I value his guidance and advice greatly.

Thank you to the National Research Foundation (NRF) for partly funding this study.

Thank you also to my family – particularly my father – for funding the remainder of my research.

I would like to acknowledge Christiaan – otherwise known as: "the best brother in the world" – for his love, his interest in my work and his cooking. These things are very much appreciated.

Last, but not least, I would like to thank Gerard Lindner for his love, support and encouragement. It means more to me than this paragraph could possibly say.

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe on any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: 25 February 2014

Abstract

The cross-entropy method for multi-objective optimisation (MOO CEM) was recently introduced by Bekker & Aldrich (2010) and Bekker (2012). Results presented by both show great promise. The MOO CEM assumes that decision variables are independent. As a consequence, the question arises: under which circumstances would an algorithm that accounts for relationships between decision variables outperform the MOO CEM? Two algorithms reported to account for relationships between decision variables, the multi-objective covariance matrix adaptation evolution strategy (MO-CMA-ES) and Pareto differential evolution (PDE), are selected for comparison. In addition, two hybrid algorithms (Hybrid 1 and Hybrid 2) based on the MOO CEM are created. These five algorithms are applied to a set of 46 continuous problems, six instances of the mission-ready resource (MRR) problem, and three instances of a dynamic, stochastic buffer allocation problem (BAP). Performance is measured using the hypervolume indicator and Mann-Whitney U-tests. One of the primary findings is that accounting for relationships between decision variables is beneficial when solving small to medium-sized problems. In these cases, the MO-CMA-ES typically outperforms the other algorithms. However, on large problems, Hybrid 1 and the MOO CEM typically perform best.

Opsomming

Die kruis-entropie metode vir meerdoelige optimering (MOO CEM) is onlangs deur Bekker & Aldrich (2010) en Bekker (2012) bekendgestel. Hul resultate is belowend. Die MOO CEM neem aan dat besluitnemingsveranderlikes onafhanklik is van mekaar. Gevolglik ontstaan die vraag: onder watter omstandighede sal 'n optimeringsalgoritme wat moontlike verhoudings tussen besluitnemingsveranderlikes in ag neem, beter vaar as die MOO CEM? Twee bestaande algoritmes, beide gerapporteer vir hul vermoë om moontlike verhoudings tussen besluitnemingsveranderlikes in ag te neem, naamlik die meerdoelige optimering kovariansiematriksaanpassing-evolusiestrategie (MO-CMA-ES) en Pareto afgeleide evolusie (PDE), word met die MOO CEM vergelyk. Twee nuwe hibriedalgoritmes (Hibried 1 en Hibried 2) word ook ter wille van dié vergelyking geskep. Die vyf algoritmes word op 'n stel van 46 kontinue probleme, ses statiese kombinatoriese gevalle en drie dinamies, stogastiese gevalle toegepas. Die prestasie van die algoritmes word deur middel van die hipervolume-aanwyser en Mann-Whitney U-toetse gemeet. 'n Primêre bevinding is dat dit voordelig is om moontlike verhoudings tussen besluitnemingsveranderlikes in ag te neem wanneer klein na medium-grootte probleme opgelos word. Vir hierdie gevalle presteer die MO-CMA-ES tipies beter as die ander algoritmes. Vir groot probleme presteer Hibried 1 en die MOO CEM beter as die ander algoritmes.

Contents

List of Figures

List of Tables

List of Algorithms

Nomenclature

Acronyms

API	Application programming interface
BAP	Buffer allocation problem
cdf	Cumulative distribution function
CEM	Cross-entropy method
CMA	Covariance matrix adaptation
CMA-ES	Covariance matrix adaptation evolution strategy
CSV	Comma separated value
DE	Differential evolution
EC	Evolutionary computation
ES	Evolution strategy
GSA	Generating set adaptation
LB	Lower bound
L_1ZDT	ZDT problems with linkages of type one
MO-CMA-ES	Multi-objective covariance matrix adaptation evolution strategy
MODE	Multi-objective differential evolution

MOO CEM	Multi-objective optimisation cross-entropy method
MRR	Mission-ready resource
MODE	Non-dominated sorting differential evolution
NSGA-II	Non-dominated sorting genetic algorithm II
PDE	Pareto differential evolution
pdf	Probability density function
SA	Simulated annealing
TS	Tabu search
UB	Upper bound
VEGA	Vector evaluated genetic algorithm
WFG	Walking fish group
WIP	Work-in-progress
ZDT	Zitzler-Deb-Thiele

Greek Symbols

α	Smoothing parameter for the CEM
β_p	Mean processing time for BAP
β_r	Mean repair time for BAP
$\delta_{i,j}$	The suitability of MRR type j for executing task i
η_j	The volume of MRR type j
γ	The $(1-\varrho)$-percentile of $f_{g-1}(\mathbf{X})$ when using the CEM
$\hat{\chi}$	Expected value of the length of a $(\mathbf{0}, \mathbf{I})$ normally distributed random vector

κ	Constant used to determine the shape of a WFG problem
λ	Number of offspring to be generated from a parent
λ_f	Mean failure count for BAP
λ_M	The number of CMA-ESs that make up the MO-CMA-ES
μ	Mean
ν	Number of parents selected when using the CMA
ω_2	Parameter used for introducing a flat region bias when constructing a WFG problem
ω_3	Parameter used for introducing a decision variable dependent bias when constructing a WFG problem
ω_5	Parameter used for shifting the optimum of an objective function to be deceptive when constructing a WFG problem
ω_6	Parameter used for changing a unimodal objective function to be multimodal when constructing a WFG problem
ϕ	Distance-scaling constant used when constructing the WFG problems
ψ_i	Shape-scaling constant for objective i when constructing WFG problems
σ	The global step size
τ_1	Parameter used for introducing a polynomial bias when constructing a WFG problem
τ_2	Parameter used for introducing a flat region bias when constructing a WFG problem

τ_3	Parameter used for introducing a decision variable dependent bias when constructing a WFG problem
τ_4	Parameter used for shifting the optimum of an objective function in linear fashion when constructing a WFG problem
τ_5	Parameter used for shifting the optimum of an objective function to be deceptive when constructing a WFG problem
τ_6	Parameter used for changing a unimodal objective function to be multimodal when constructing a WFG problem
τ_8	Parameter used for non-separable reduction when constructing a WFG problem
θ	Constant used to determine the locations of disconnected regions of a WFG problem
υ_2	Parameter used for introducing a flat region bias when constructing a WFG problem
υ_3	Parameter used for introducing a decision variable dependent bias when constructing a WFG problem
υ_5	Parameter used for shifting the optimum of an objective function to be deceptive when constructing a WFG problem
υ_6	Parameter used for changing a unimodal objective function to be multimodal when constructing a WFG problem
φ_j	The weight of MRR type j

ϖ_3	Result of a reduction function (typically a weighted sum reduction function) used for introducing a decision variable dependent bias when constructing a WFG problem
ϱ	The percentage of population t used to estimate population $t+1$

Roman Symbols

A_j	The number of MRRs of type j that are available
$\mathbf{a}$	A vector of objective function values
$\mathbf{B}$	Normalised eigenvectors of the covariance matrix $\mathbf{C}$
B_i	Buffer size i
$\mathbf{b}$	A vector of objective function values
$\mathbf{C}$	Covariance matrix
c_1	A constant used for the mutative step size control rule
c_2	A value used for the mutative step size control rule
c_3	A value used for the mutative step size control rule
C_C	Constant used to determine the number of convex regions of a WFG problem
c_c	Constant for updating $\mathbf{C}$
C_D	Constant used to determine the number of disconnected regions of a WFG problem
c_m	Constant value used for mutation in DE
c_p	Constant for updating the covariance matrix evolution path $\mathbf{p}_c$
c_r	Constant value used for recombination in DE

c_s	Constant for updating the step size evolution path$\mathbf{p}_s$
c_v	Constant used for PDE
$\mathbf{D}$	Diagonal matrix made up of the square roots of the eigenvalues of the covariance matrix $\mathbf{C}$
$\mathcal{D}$	Cross-entropy of two probability density functions
d_s	Damping parameter
$e_i(\mathbf{x})$	The i^{th} constraint associated with $\mathbf{x}$
$f^*(\mathbf{x})$	Optimal objective function value
$f_i(\mathbf{x})$	The i^{th} objective associated with $\mathbf{x}$
$g^*(\mathbf{x})$	Optimal probability density function when deriving the CEM
$g(\mathbf{x})$	Probability density function when deriving the CEM
$h(\mathbf{x})$	Probability density function when deriving the CEM
$\mathbf{I}$	Identity matrix
i	General index
J	Random value from a uniform (0,1) distribution
j	General index
J_V	Random value uniformly drawn to fall between 1 and V
K	Number of constraints
k	General index
M	Number of objective functions
m	Number of task types for the MRR problem
N	Population size

n	Number of MRR types for the MRR problem
$\mathbf{P}$	Matrix of random values used to construct L_1ZDT problems
$\mathbf{p}_c$	The covariance matrix evolution path
p_g	The goal value of the step size evolution path
p_h	Probability that a histogram will be inverted when using the MOO CEM
$\mathbf{p}_s$	The step size evolution path
p_t	Threshold value for updating the covariance matrix evolution path
$\mathbf{Q}$	Matrix of random values used to construct L_1ZDT problems
$\mathbf{R}$	Rotation matrix used in construction of R problems
$\mathbf{v}$	Reference parameter vector for CEM
R_i	The number of MRRs required for tasks of type i
r_i	Randomly chosen indices with $i = 1, 2, 3$ used for DE
S	Indicates whether or not the offspring of a parent ranked as high or higher than the parent
t	Index for the current generation
$\mathbf{u}$	Parameter vector for CEM
V	Number of decision variables
V_1	The number of decision variables used in the first objective
V_n	The number variables to be reduced using a non-separable reduction function when constructing a WFG problem

V_w	The number of variables to be reduced using the weighted sum reduction function when constructing a WFG problem
w_j	The weight assigned to the j^{th} variable when using a weighted sum reduction function when constructing a WFG problem
$\mathbf{X}$	Entire population of decision vectors $x_1, \ldots, x_N$
x_i	Decision variable i
$\mathbf{x}_m$	Temporary vector used for DE
$\mathbf{x}$	A vector or matrix of decision variables
y_i	Variables used for documenting the WFG problems
y'_i	Temporary placeholders for WFG problems, with $i = 1, 2, 3, 4$
y''_i	Temporary placeholders for WFG problems, with $i = 1, 2$
$\mathbf{z}$	A sample from a $N(0, 1)$ distribution

Terminology

a posteriori optimisation	Optimisation methods where user preferences taken into account after the results of a mathematical model is known
a priori optimisation	Optimisation methods where user preferences are incorporated into a mathematical model before the results are known
elitism	The practise of carrying the best solutions found in generation t over to generation $t+1$ in order to avoid losing these solutions

feasible set	The set of decision vectors that satisfy all constraints
metaheuristic	Algorithm designed to solve approximately a wide range of hard optimisation problems without having to deeply adapt to each problem
non-dominated vector	A decision vector $\mathbf{x}$ is said to be non-dominated regarding a subset $\mathbf{A}$ of the feasible set, if there exists no $\mathbf{x}_i$ in $\mathbf{A}$ so that $\mathbf{x}_i$ dominates $\mathbf{x}$.
Pareto front	The set of objective vectors associated with the Pareto optimal set
Pareto optimal	A decision vector that is non-dominated with regard to entire feasible set is said to be Pareto optimal
Pareto optimal set	The set of Pareto optimal vectors is referred to as the Pareto optimal set
Pareto ranking	The process of finding non-dominated fronts and sets

Other Symbols

$\prec$	For a problem where all objectives have to minimised, $\mathbf{x}_i \prec \mathbf{x}_j$ means that $\mathbf{x}_i$ dominates $\mathbf{x}_j$
$\sim$	For a problem where all objectives have to minimised, $\mathbf{x}_i \sim \mathbf{x}_j$ means that $\mathbf{x}_i$ is indifferent to $\mathbf{x}_j$
$\preceq$	For a problem where all objectives have to minimised, $\mathbf{x}_i \preceq \mathbf{x}_j$ means that $\mathbf{x}_i$ weakly dominates $\mathbf{x}_j$

Chapter 1

Introduction

The main focus of this study falls on the effect of accounting for possible relationships between decision variables on the performance of multi-objective optimisation algorithms under different circumstances.

This chapter presents the background to the research problem, poses the research question and lays out a simple methodology.

1.1 Background and research rationale

Multi-objective optimisation problems are not just encountered in the field of Industrial Engineering, but in other disciplines as well. Whereas finding the solution to a single-objective optimisation problem involves finding the single absolute optimum point, finding the solution to a multi-objective problem involves finding a set of good solutions. This means that solving multi-objective problems requires more evaluations of the problem and is therefore more computationally expensive than solving single-objective problems.

For his PhD, Bekker (2012) focused on lessening this computational burden. His multi-objective optimisation cross-entropy method (MOO CEM) algorithm performs well when compared to the multi-objective genetic algorithm implemented in Matlab 2007b. This comparison was done on a set of benchmark problems and a number of real-world problems.

As it stands, the MOO CEM selects the values of candidate solutions variables independently. This effectively ignores relationships that might exist between variables.

Decision variables for real-world problems are not necessarily completely independent.

The term *relationship* is used to indicate that variable values for such problems should preferably not be selected independently since simply improving one variable will not necessarily yield better objective function values. Improved combinations of variables need to be found in order to find better objective function values. It is used to group together three terms recently used in literature on multi-objective test problem design: *linkage* as used by Deb *et al.* (2006), *rotated variables* as used by Iorio & Li (2006), and *non-separability* as used by Huband *et al.* (2006). All three works propose multi-objective test problems for algorithm comparison with different mechanisms for introducing relationships between decision variables.

It is suspected that an algorithm that accounts for possible relationships between decision variables could outperform an algorithm that does not do so for some classes of problems.

1.2 Research question

From the background, it follows that the basic research question of this project is:

> **Would a multi-objective optimisation algorithm that accounts for possible relationships between decision variables outperform an algorithm that does not do so?**

The No Free Lunch theorems put forth by Wolpert & Macready (1997) state that even though an optimisation algorithm might outperform another algorithm on a certain class of problems, the "average performance of any pair of optimisation algorithms across all possible problems is identical". Taking this into account, there is a simple answer to the question above: yes, in some cases an algorithm that accounts for relationships between decision variables would outperform an algorithm that does not do so. However, on average, algorithm performance would be identical.

The research question is thus formulated below:

> **For which of the problems investigated does a multi-objective algorithm that accounts for possible relationships between decision variables outperform a multi-objective algorithm that does not do so?**

1.3 Methodology

In order to answer the research question, the methodology as discussed below is followed.

Figure 1.1 shows the typical procedure followed to compare two or more multi-objective optimisation algorithms as described by Huband *et al.* (2006).

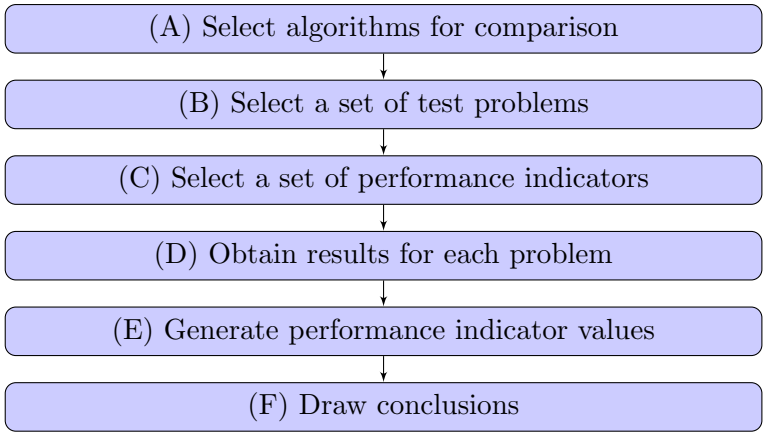


Figure 1.1: Typical methodology for comparing multi-objective optimisation algorithms.

Referring to Figure 1.1, a more detailed version of the methodology used is as follows:

1. **Select algorithms for comparison (A)** – This project builds on the research done by Bekker & Aldrich (2010) and Bekker (2012). The first algorithm is therefore the MOO CEM. In addition to the MOO CEM, two existing algorithms noted for their ability to solve problems where there are relationships between decision variables were identified, namely the multi-objective covariance matrix adaptation evolution strategy (MO-CMA-ES) and Pareto differential evolution (PDE). In addition to these, two hybrid algorithms (Hybrid 1 and Hybrid 2) were developed out of research curiosity. There are thus five algorithms in total. These algorithms are discussed in Chapter 3.

2. **Select a set of test problems (B)** – The test suite consists of 28 continuous optimisation problems with different characteristics (as discussed in Chapter

4), a static combinatorial problem (as discussed in Chapter 5) and a dynamic combinatorial problem (as discussed in Chapter 6).

3. **Select a set of performance indicators (C)** – The primary performance indicator is the hypervolume indicator. The relative run times of algorithms are also recorded. More information on the performance indicators can be found in Chapter 7.

4. **Obtain results for each problem (D) and generate performance indicator values (E)** – Keeping in mind the aim of Bekker (2012) to reduce the computational burden of solving multi-objective problems, the maximum number of function evaluations will be kept relatively small throughout. The results for the continuous, static combinatorial and dynamic combinatorial problems can be found in Appendices A, B and D respectively. The results are discussed in Chapter 8.

5. **Draw conclusions (F)** – The conclusions drawn, along with recommendations for future research, can be found in Chapter 9.

1.4 Conclusion: Introduction

This chapter presented the research rationale, posed the research question and laid out a simple methodology.

The next chapter will focus on some basics regarding multi-objective optimisation, specifically focusing on multi-objective optimisation using metaheuristics.

Chapter 2

An introduction to multi-objective optimisation

Chapter 1 provided the background to the research problem, put forth the primary research question, and laid out the basic methodology.

In this chapter, important concepts relating to multi-objective optimisation are discussed. First, a short introduction to multi-objective optimisation is presented, followed by a short history of Pareto-based multi-objective optimisation. Next, some important concepts are explained briefly, including concepts such as Pareto dominance, Pareto optimality, metaheuristics and elitism.

2.1 A short introduction to multi-objective optimisation

Multi-objective problems are found in design, manufacturing, logistics, health care and financing, to name a few areas. A multi-objective problem generally consists of a set of M objective functions $(f_1(\mathbf{x}), \ldots, f_M(\mathbf{x}))$. These objective functions are functions of a set of V decision variables $(x_1, \ldots, x_V)$ and are subject to a set of K constraints $(e_1(\mathbf{x}), \ldots, e_K(\mathbf{x}))$. The constraints are also functions of the decision variables (Zitzler, 1999).

Figure 2.1 shows how an unconstrained multi-objective problem with two decision variables and two objectives maps from the decision space to the objective space.

A multi-objective problem is only truly multi-objective if the objectives are conflicting. If the objectives do not conflict, a single optimal solution exists (Zitzler, 1999).

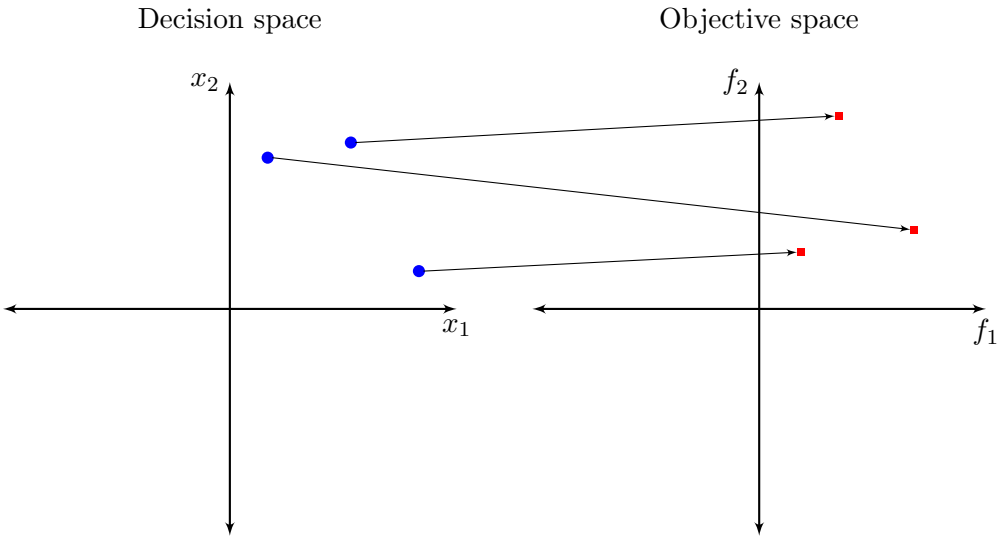


Figure 2.1: Multi-objective optimisation mapping.

Many approaches to solving multi-objective problems exist. A few such approaches are:

- **The weighted sum approach** – The various objective functions are each assigned a weight and summed together to form one objective function with a single optimal answer. Even though the weights assigned to each objective can be varied in order to find more than one solution to the problem, many points that could otherwise have been considered optimal are missed (De Weck, 2004).

- **Lexicographic ordering** – The objective function considered to be most important is optimised. The second most important objective function is then optimised without lowering the quality of the first objective value. This is done for all objectives (Coello Coello, 2006).

- **Multi-attribute utility analysis** – Optimality is measured in terms of utility to the decision-maker. Despite being efficient and widely used, this method requires extensive interviews in order to determine the relevant utility functions. It should be noted that decision makers might be influenced by the structure of the interviews (De Weck, 2004).

- **Goal programming** – Each objective function is associated with a target value. Deviations from the target values are minimised. This can be done using a weighted sum approach, or lexicographically.
- **Multi-objective metaheuristics** – Metaheurstics (such as the genetic algorithm, simulated annealing and differential evolution) are extended for use on multi-objective problems. This is usually done using the concept of Pareto dominance, as discussed in Section 2.3.

These optimisation approaches can generally be classified into three main categories (De Weck, 2004):

- Methods where decision-maker preferences are incorporated into the mathematical model before the results are known. Such methods are said to be *a priori*. The weighted sum approach, lexicographic ordering, multi-attribute utility analysis and goal programming are typical examples.
- Methods where decision-maker preferences are taken into account during the process of searching for results.
- Methods where decision makers are given results from a mathematical model, which they can then use to make an informed decision. These techniques are referred to as *a posteriori* methods. The use of multi-objective metaheuristics comes to mind.

Focus of this work falls on *a posteriori* methods. More specifically, this study focuses on Pareto-based multi-objective metaheuristics.

2.2 A short history of Pareto-based multi-objective optimisation

In 1881, an economics professor at King's College in London, Prof. FY Edgeworth, defined the optimum of a multi-criteria problem where one criterion A is to be maximised and another criterion B minimised (Coello Coello, 2006; De Weck, 2004). From Edgeworth (1881): "It is required to find a point (x, y) such that in whatever direction

we take an infinitely small step, A and B do not increase together, but that, while one increases, the other decreases."

A similar definition, known as the Pareto optimum, can be seen as a generalisation of Edgeworth's definition (Coello Coello, 2006) and was proposed by Vilfredo Pareto in 1896: "The optimum location of the resources of a society is not attained so long as it is possible to make at least one individual better off in his own estimation while keeping others as well off as before in their own estimation."

David Schaffer is generally credited with the first multi-objective evolutionary algorithm (put forth in 1985): the vector evaluated genetic algorithm (VEGA). VEGA is not a Pareto-based algorithm but made use of subpopulations for selection (Coello Coello, 2006).

In 1989, more than a century after the concept was first introduced, David Goldberg suggested using Pareto optimality for selection in multi-objective evolutionary algorithms (Coello Coello, 2006). This has since been a popular method for solving multi-objective problems – especially two- and three-objective problems.

2.3 Pareto dominance and other important concepts

When solving single-objective optimisation problems, there exists a single optimal solution. This is not the case when solving multi-objective problems. Let us look at the example of minimising the total cost of a system while minimising its stockouts. From single-objective optimisation, there exists a solution where total cost is at a minimum, a solution where the number of stockouts is at a minimum and a trade-off curve where neither the total cost, nor the number of stockouts are minimal but where neither of the objectives can be improved upon without detriment to the other. This is an informal explanation of the Pareto front. In order to define it formally, we need several definitions.

The first important definition is that of Pareto dominance. From Zitzler (1999):

Definition 2.1. Pareto dominance Consider a case where all objective functions are to be minimised. For any two objective vectors, $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{a}$ is said to be equal to $\mathbf{b}$ ($\mathbf{a} = \mathbf{b}$) if $a_i = b_i$ for $i = 1, \ldots, M$. Similarly:

$\mathbf{a} \leq \mathbf{b}$ if $a_i \leq a_i$ for $i = 1, \ldots, k$, and

$\mathbf{a} < \mathbf{b}$ if $a_i \leq a_i$ for $i = 1, \ldots, k$ and $\mathbf{a} \neq \mathbf{b}$.

For any two decision vectors $\mathbf{x}_i$ and $\mathbf{x}_j$,

$\mathbf{x}_i$ is said to dominate $\mathbf{x}_j$ ($\mathbf{x}_i \prec \mathbf{x}_j$) if $\mathbf{f}(\mathbf{x}_i) < \mathbf{f}(\mathbf{x}_j)$,

$\mathbf{x}_i$ is said to weakly dominate $\mathbf{x}_j$ ($\mathbf{x}_i \preceq \mathbf{x}_j$) if $\mathbf{f}(\mathbf{x}_i) \leq \mathbf{f}(\mathbf{x}_j)$,

$\mathbf{x}_i$ is said to be indifferent to $\mathbf{x}_j$ ($\mathbf{x}_i \sim \mathbf{x}_j$) if $\mathbf{f}(\mathbf{x}_i) \not\leq \mathbf{f}(\mathbf{x}_j)$ and $\mathbf{f}(\mathbf{x}_j) \not\leq \mathbf{f}(\mathbf{x}_i)$.

The definitions of the $\geq$ and $>$, and $\succ$ and $\succeq$ relations are similar. The $\succ$ and $\succeq$ relations are used when objectives are to be maximised.

The dominance relations in Definition 2.1 give rise to the notion of non-dominance:

Definition 2.2. A non-dominated vector A decision vector $\mathbf{x}$ is said to be non-dominated regarding a subset $\mathbf{A}$ of the feasible set, if there exists no $\mathbf{x}_i$ in $\mathbf{A}$ so that $\mathbf{x}_i \prec \mathbf{x}$.

The set of decision vectors that satisfy all constraints is referred to as the *feasible set* (Zitzler, 1999). If a decision vector $\mathbf{x}$ is non-dominated with regard to the entire feasible set, $\mathbf{x}$ is said to be *Pareto optimal*. The collection of all Pareto optimal solutions is called the *Pareto optimal set* and the associated objective function vectors are referred to as the *Pareto front*. Figure 2.2 shows a Pareto front for two minimised objectives.

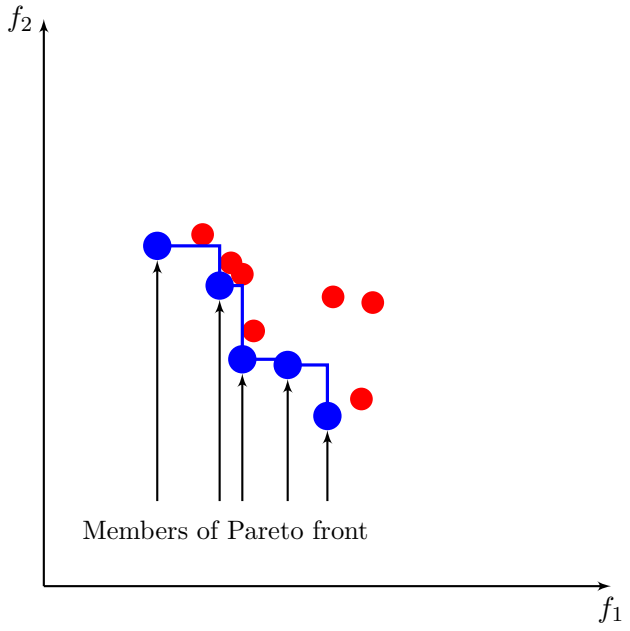


Figure 2.2: Pareto front explained for two minimised objectives.

Pareto ranking involves finding all non-dominated fronts and sets, including the Pareto optimal set and front. As mentioned above, the Pareto optimal set and front (also referred to as the first set and front in this work) are non-dominated in terms of the entire population. The second front is non-dominated in terms of the solutions excluding the first front. The third front is non-dominated in terms of the solutions excluding the first and second fronts, etc.

Pareto ranking was used for selection for all the algorithms discussed in Chapter 3.

2.4 Preference order ranking

As the number of objectives increases, the size of the Pareto optimal set and front increases, making Pareto ranking less effective for selection (Di Pierro *et al.*, 2007).

Das (1999) proposed a more stringent selection criterion called preference ordering. Di Pierro *et al.* (2007) showed that the size of the Pareto optimal set does not increase as dramatically when using preference order ranking as is the case when Pareto ranking is used. However, preference order ranking is less effective at maintaining diversity than Pareto ranking (Di Pierro *et al.*, 2007).

A preference order algorithm was implemented in Matlab and experimented with for solving the mission-ready resource problem, described in Chapter 5.

2.5 What are metaheuristics?

As mentioned at the start at of this chapter, the focus of this study falls on Pareto-based metaheuristics for multi-objective optimisation. Wheras the previous section discussed some important concepts regarding Pareto optimality, this section covers some basic concepts regarding metaheuristics.

Boussaïd *et al.* (2013) provide this simple definition of a metaheuristic: "A metaheuristic is an algorithm designed to solve approximately a wide range of hard optimisation problems without having to deeply adapt to each problem." They define hard optimisation problems as "problems that cannot be solved to optimality, or to any guaranteed bound, by an exact (deterministic) method within a 'reasonable' amount of time."

Metaheuristics are typically contrasted with problem-specific heuristics, with the Greek prefix "meta" indicating that such heuristics are higher-level heuristics than their problem-specific counterparts (Boussaïd *et al.*, 2013). Unlike problem-specific exact algorithms, which guarantee the optimality of the solutions found, the solutions found by metaheuristics are not regarded as optimal but rather as "near-optimal" (Talbi, 2009). In contrast to approximation algorithms, metaheuristics also do not quantify how far away the optimal solutions are from the solutions obtained (Talbi, 2009).

Successful metaheuristics are able to find a balance between *exploration* and *exploitation*. Exploration identifies areas in the search space resulting in high-quality solutions. Exploitation intensifies the search in such areas (Boussaïd *et al.*, 2013).

A popular classification scheme for metaheuristics differentiates algorithms based on whether they are single-solution-based or population-based. Examples of a few single-solution-based metaheuristics are simulated annealing (SA), tabu search (TS), the greedy randomised adaptive search procedure for combinatorial optimisation, and variable neighbourhood search (Boussaïd *et al.*, 2013). SA has its roots in the Metropolis algorithm and is regarded as the first metaheuristic, whereas TS uses a memory structure to escape local minima (for minimisation problems) and enforce exploration.

Population-based searches, on the other hand, include (Boussaïd *et al.*, 2013):

- evolutionary computation (EC) – a general term for algorithms that are inspired by Darwin's theory of evolution. It includes:
  - evolutionary algorithms, a subset of EC that uses biological evolution mechanisms specifically. The selection, recombination and mutation mechanisms are especially popular. Evolutionary algorithms include (to name a few):
    - the genetic algorithm
    - genetic programming
    - evolutionary programming
    - the memetic algorithm
    - evolution strategy (ES), upon which the covariance matrix adaptation evolution strategy (CMA-ES) – discussed in Section 3.2.2 – is built
    - differential evolution (DE), which is discussed in Section 3.3.2

  - the cultural algorithm
  - swarm intelligence, which includes (amongst others):
    * particle swarm optimisation
    * bacterial foraging optimisation
    * bee colony optimisation

- the cross-entropy method, discussed in Section 3.1.2.

This is just one way of classifying metaheuristics. Other popular classification methods include (Talbi, 2009):

- whether or not an algorithm is inspired by nature
- whether or not an algorithm makes use of a memory mechanism
- whether the algorithm is deterministic or stochastic
- whether the algorithm is iterative or greedy.

As pointed out earlier in this section, good metaheuristics are able to find a balance between exploration and exploitation. Algorithms should therefore be able to maintain a diverse population, without losing the best solutions. Mechanisms for accomplishing this are discussed next.

2.5.1 Fitness functions vs objective functions

Instead of using objective functions to select good solutions, metaheuristics sometimes make use of *fitness functions* in order to evaluate the performance of different solutions.

Fitness functions can be defined in many different ways. For the single-objective case, the fitness function is a function of the objective function and constraint functions (Zitzler, 1999). For an unconstrained problem, the fitness function is simply a function of the objective function.

Defining fitness functions is more complicated when using multi-objective optimisation. Zitzler (1999) identifies three main multi-objective approaches to fitness assignment:

- **Aggregating objectives** – All objective functions and constraint functions are aggregated into a single fitness function.

- **Alternating between objectives** – Instead of aggregating objective functions, good solutions are selected based on performance on a single-objective (or the fitness associated with that single-objective and the constraint functions). The specific objective used to rate performance is alternated.
- **Pareto-based approaches** – Each objective may be associated with a fitness function made up of that objective and the constraint functions. Good solutions are selected using the concept of non-domination discussed above.

Out of the multi-objective optimisation algorithms investigated in this study (and discussed in Chapter 3), only the multi-objective covariance matrix adaptation evolution strategy (MO-CMA-ES) makes use of fitness function by default. If a solution violates the box constraints of a problem, the solution is fixed to adhere to the box constraints and the objective functions are evaluated as usual, but the fitness associated with each objective is decreased proportionately to the size of the violation. Otherwise the fitness associated with each objective is simply equal to the value of the objective function. The MO-CMA-ES is Pareto-based, and good solutions are selected by Pareto ranking the fitness function values associated with all the objectives.

For the combinatorial mission-ready resource (MRR) problem (discussed in Chapter 5), fitness functions were used for constraint handling.

2.5.2 Elitism

Elitism is the practice of carrying the best solutions found in generation t over to generation $t+1$ in order to avoid losing these solutions during the selection or recombination processes. The idea was first suggested by De Jong (1975) for the single-objective case.

Zitzler (1999) discusses some of the complexities involved in extending the principle to the multi-objective case. Where single-objective problems have one best solution in every generation, multi-objective problems have a set of Pareto optimal solutions. Should all these solutions be kept? And for how long? Should these solutions have an influence on the selection process? And if so, when and how?

Zitzler (1999) showed that algorithms that make use of an elitist set as part of the population outperforms algorithms that do not do so. The experiments were done using six different algorithms on the six ZDT problems (discussed in Chapter 4).

De Jong (1975) found that elitism can be beneficial when solving unimodal single-objective problems, but might lead to early convergence on local optimums when solving multi-modal single-objective problems.

2.5.3 Maintaining population diversity

It is important to maintain population diversity in order to avoid premature convergence to a local optimum (Zitzler, 1999). Several methods for maintaining population diversity have been developed. Such methods include (Zitzler, 1999):

- **Fitness sharing** – The fitness function values of individuals that are surrounded by many similar individuals are degraded. This encourages search in less well-explored areas. The distance between individuals can be calculated in the individual space, the decision space or the objective space. This is the most commonly used technique for maintaining population diversity.
- **Restricted mating** – Only individuals within a specified distance from one another are allowed to mate.
- **Isolation by distance** – Individuals are assigned a conceptual location and only individuals that are conceptually located close to one another are allowed to mate.
- **Overspecification** – Individuals are made up of active and inactive parts. As the search progresses, The active parts may become inactive and the inactive parts, active. Information hidden on individuals thus aide in maintaining a diverse population.
- **Reinitialisation** – In order to avoid premature convergence, the population (or parts thereof) is reset at a specific point in time or when the search starts stagnating.
- **Crowding** – New individuals replace similar individuals.

These techniques for maintaining population diversity are simply popular techniques; other methods may also be used.

2.6 Conclusion: An introduction to multi-objective optimisation

In this chapter, important concepts relating to multi-objective optimisation were discussed. Multi-objective optimisation was introduced briefly, followed by a short history of Pareto-based multi-objective optimisation. Finally, important concepts such as Pareto dominance, non-domination, Pareto fronts, evolutionary algorithms and population diversity were explained briefly.

Chapter 3 will focus on the algorithms selected for comparison.

Chapter 3

The multi-objective optimisation algorithms under investigation

Chapter 1 presented the research question: for which of the problems investigated does a multi-objective algorithm that accounts for possible relationships between decision variables outperform an multi-objective algorithm that does not do so?

Chapter 2 briefly introduced the reader to some key concepts relating to multi-objective optimisation.

This chapter presents the algorithms selected for comparison. These algorithms include the cross-entropy method for multi-objective optimisation (MOO CEM), the multi-objective covariance matrix adaptation evolution strategy (MO-CMA-ES), Pareto differential evolution (PDE) and two hybrid algorithms that were developed out of research curiosity (Hybrid 1 and Hybrid 2).

The single-objective versions of the MOO CEM, MO-CMA-ES and PDE will each be presented before their multi-objective counterparts are discussed.

3.1 The cross-entropy method for multi-objective optimisation

As mentioned in Chapter 1, the MOO CEM – and the single-objective cross-entropy method (CEM) that inspired it – works based on the assumption that decision variables are independent. This assumption forms the basis of this study.

This section motivates the investigation of the MOO CEM, presents the CEM and subsequently the MOO CEM.

3.1.1 Why investigate the performance of the cross-entropy method for multi-objective optimisation?

The cross-entropy method (CEM) was expanded for multi-objective optimisation and introduced in Bekker & Aldrich (2010) and Bekker (2012). It was shown to outperform the non-dominated sorting genetic algorithm II (NSGA-II) on a set of continuous benchmark problems. It also performed well when applied to stochastic, discrete problems.

The cross-entropy method for multi-objective optimisation (MOO CEM) forms the basis of this study in a few ways:

- The results presented by Bekker & Aldrich (2010) and Bekker (2012) show much promise, outperforming the popular NSGA-II on a variety of problems. Based on this promise, the MOO CEM deserves further investigation.
- The MOO CEM is a novel algorithm and has thus far only officially been compared to one published multi-objective optimisation algorithm – the NSGA-II.
- The nature of the CEM is such that it assumes that decision variables are independent of one another. Subsequently its multi-objective counterpart, the MOO CEM, also assumes independence of decision variables (as does the NSGA-II). As a result, the question of whether or not the MOO CEM would be outperformed by algorithms that do not assume independence of decision variables arises. Referring to Chapter 1, the reader is reminded that the main research question relates directly to this.

3.1.2 The cross-entropy method for single-objective optimisation

The CEM was first introduced in Rubinstein (1999) and has since been used to solve a variety of continuous, discrete and combinatorial problems. It aims to find the optimal function value $f^*(\mathbf{x})$ to some objective function $f(\mathbf{x})$ and the decision variable values $\mathbf{x}^*$ associated with $f^*(\mathbf{x})$. In the continuous case, the expected value of $f(\mathbf{x})$ is

$$\mathbb{E}[f(\mathbf{X})] = \int f(\mathbf{x})g(\mathbf{x})d\mathbf{x}, \tag{3.1}$$

where $f(\mathbf{x})$ is the sample performance and $g(\mathbf{x})$ is the probability density of $\mathbf{X}$.

How would a random search algorithm go about searching for the optimal answer? For each variable, the random search algorithm would select a value for the decision variable by drawing from a uniform probability density function (pdf). It would then evaluate the combination of variables and, irrespective of the function value achieved, would once again select decision variable values from a uniform pdf. After a set number of iterations, the random search algorithm might or might not have come across the optimal solution. The question then arises: could the odds of finding the optimal solution not be better if a search algorithm was more likely to draw decision variable values that result in good objective function values?

The CEM works on this principle. Every decision variable domain is associated with a pdf. These pdfs are used to draw decision variables from and are updated based on the objective function value performance of different decision variables. If, for example, $x_1 = b$ is associated with good objective function values, whereas $x_1 = c$ is not, the pdf for x_1 is adjusted so that $x_1 = b$ will be drawn more often than $x_1 = c$. Ideally, the values of $\mathbf{x}$ associated with $f^*(\mathbf{x})$ should all be drawn with a probability of one by the time the algorithm terminates.

The CEM has its roots in importance sampling, which involves choosing a sampling distribution that favours important samples (Rubinstein & Kroese, 2008). Instead of using importance sampling, the Kullback-Liebler distance (or relative entropy or cross-entropy) can be used to choose a sampling distribution that favours important samples.

In short, the CEM is concerned with estimating the ideal pdf (including the parameters of the pdf) for each decision variable, in order to converge to an optimal answer. It does this by using the cross-entropy (or Kullback-Liebler distance) of the two pdfs. The cross-entropy $\mathcal{D}$ is a measure of the distance between two pdfs $g(\mathbf{x})$ and $h(\mathbf{x})$ (Cover & Thomas, 2006; Rubinstein, 1999; Rubinstein & Kroese, 2004, 2008) and is denoted by

$$\mathcal{D} = \mathbb{E}_g \left[\ln \frac{g(\mathbf{x})}{h(\mathbf{x})} \right] \tag{3.2}$$

$$= \int g(\mathbf{x}) \ln \frac{g(\mathbf{x})}{h(\mathbf{x})} d\mathbf{x} \tag{3.3}$$

$$= \int g(\mathbf{x}) \ln g(\mathbf{x}) d\mathbf{x} - \int g(\mathbf{x}) \ln h(\mathbf{x}) d\mathbf{x}. \tag{3.4}$$

If two pdfs are identical ($g(\mathbf{x}) = h(\mathbf{x})$), then $\mathcal{D} = 0$. We want to choose the pdf from which we sample $h(\mathbf{x})$ in such a way that the cross-entropy distance between $h(\mathbf{x})$ and the optimal pdf to be sampled from $g^*(\mathbf{x})$ is at a minimum (Rubinstein, 1999; Rubinstein & Kroese, 2004, 2008).

If we assume that the pdf we are sampling from and the ideal pdf are from the same parametric family of distributions, then we can concern ourselves simply with choosing the parameter of the sampling pdf such that the cross-entropy distance is at a minimum (Rubinstein, 1999; Rubinstein & Kroese, 2004, 2008).

The importance sampling density is denoted as $h(\mathbf{x}, \mathbf{v})$ where $\mathbf{v}$ is the reference parameter vector. Since the pdf we are trying to approximate comes from the same parametric family of distributions, from (3.1), we denote the optimal sampling distribution as $f(\mathbf{x})h(\mathbf{x}, \mathbf{u})$ where $\mathbf{u}$ is the parameter vector of the pdf. Minimising the cross-entropy distance with respect to $\mathbf{v}$ is given by

$$\min_v \left[\mathcal{D} = \int f(\mathbf{x})h(\mathbf{x}, \mathbf{u}) \ln f(\mathbf{x})h(\mathbf{x}, \mathbf{u}) d\mathbf{x} - \int f(\mathbf{x})h(\mathbf{x}, \mathbf{u}) \ln f(\mathbf{x})h(\mathbf{x}, \mathbf{v}) d\mathbf{x} \right]. \tag{3.5}$$

Since there is no $\mathbf{v}$ in the first term, this is equivalent to

$$\max_v \left[\mathcal{D} = \int f(\mathbf{x})h(\mathbf{x}, \mathbf{u}) \ln f(\mathbf{x})h(\mathbf{x}, \mathbf{v}) d\mathbf{x} \right]. \tag{3.6}$$

From (3.2), (3.6) is equivalent to

$$\max_v \left[\mathcal{D}(\mathbf{v}) = \mathbb{E}_{\mathbf{u}} \left[f(\mathbf{X}) \ln h(\mathbf{X}, \mathbf{v}) \right] \right]. \tag{3.7}$$

According to Rubinstein & Shapiro (1990), $\mathcal{D}$ is generally convex and differentiable with respect to $\mathbf{v}$. The solution to (3.7) can therefore be found by differentiating with respect to $\mathbf{v}$ and setting the result equal to zero:

$$\mathbb{E}_u \left[f(\mathbf{X}) \nabla \ln h(\mathbf{X}, \mathbf{v}) \right] = \mathbf{0}. \tag{3.8}$$

Alternatively, the sample estimator equivalent can be set to zero:

$$\frac{1}{N} \sum_{k=1}^{N} f(\mathbf{X}_k) \nabla \ln h(\mathbf{X}_k, \mathbf{v}) = \mathbf{0}. \tag{3.9}$$

If the distribution of $\mathbf{X}$ is assumed to be from the exponential family of distributions (which includes, among other distributions, the exponential, normal, Poisson, gamma, chi-squared and geometric distributions), then – for a one-dimensional exponential family, parameterised by the mean – the sample estimator of the optimal reference parameter (when using cross-entropy) is always (Rubinstein & Kroese, 2008):

$$\hat{v}_i = \frac{\sum_{k=1}^{N} f(\mathbf{X}_k) X_{ki}}{\sum_{k=1}^{N} f(\mathbf{X}_k)}, \tag{3.10}$$

where i indicates the i^{th} variable.

Up to here, we have derived a sample estimator for the optimal reference parameter if we wanted to minimise the cross-entropy of two pdfs. Next, we will see how this can be used for single-objective optimisation.

Keeping in mind that, as discussed above, the ultimate goal of the CEM algorithm is to find the optimal function value $f^*(\mathbf{x})$ of some function $f(\mathbf{x})$, the probability that $f(\mathbf{x})$ will be greater than some value γ when maximising, can be treated as a rare-event probability

$$\mathbb{P}_{\mathbf{u}}(f(\mathbf{X}) \geq \gamma) = \mathbb{E}_{\mathbf{u}}(I_{\{f(\mathbf{X}) \geq \gamma\}}), \tag{3.11}$$

where the random state $\mathbf{X}$ has a pdf $h(\cdot, \mathbf{u})$. $I_{\{f(\mathbf{X}) \geq \gamma\}}$ is an indicator function

$$I_{\{f(\mathbf{X}) \geq \gamma\}} = \begin{cases} 1, & \text{if} \quad f(\mathbf{X}) \geq \gamma \\ 0, & \text{if} \quad f(\mathbf{X}) < \gamma \end{cases}. \tag{3.12}$$

Then, the optimal reference vector can be found by solving

$$\max_{v} \left[\hat{\mathcal{D}}(\mathbf{v}) = \frac{1}{N} \sum_{k=1}^{N} I_{\{f(\mathbf{X}_k) \geq \gamma_t\}} \ln h(\mathbf{X}_k, \mathbf{v}) \right]. \tag{3.13}$$

At generation t, γ_t is the $(1 - \varrho)$-quantile of $f_{t-1}(\mathbf{X})$. The answer to (3.13) can be estimated by

$$\hat{v}_i = \frac{\sum_{k=1}^{N} I_{\{f(\mathbf{X})\geq\gamma\}} X_{ki}}{\sum_{k=1}^{N} I_{\{f(\mathbf{X})\geq\gamma\}}}, \tag{3.14}$$

when the pdfs are assumed to be from exponential families.

Algorithm 1 shows the basic CEM algorithm (Rubinstein, 1999), (Rubinstein & Kroese, 2004), (Rubinstein & Kroese, 2008).

Algorithm 1 The basic CEM algorithm.

1: Choose an initial parameter vector $\mathbf{v_0}$.
2: Set generation $t \to 1$.
While stopping criteria not met,
3: Generate sample $\mathbf{X_1}, \ldots, \mathbf{X_1}$ from the pdf $h(\cdot, v_{t-1})$.
4: Evaluate the sample.
5: Find the $(1-\varrho)$-quantile of the sample performance.
6: Denote the solution to (3.14) by $\tilde{\mathbf{v}}_t$.
7: Smooth $\tilde{\mathbf{v}}_t$: $\hat{\mathbf{v}}_t = \alpha\tilde{\mathbf{v}}_t + (1-\alpha)\hat{\mathbf{v}}_t$, where α is called the smoothing parameter and typically ranges from 0.7 to 1.
end while

3.1.3 The cross-entropy method for multi-objective optimisation

The cross-entropy method for multi-objective optimisation (MOO CEM) using inverting histograms was introduced in Bekker & Aldrich (2010) and is tested extensively in Bekker (2012). Since the CEM discussed above aims to find the single optimal reference parameter that would lead to a single optimal solution being drawn, the challenge of expanding the CEM for use in multi-objective optimisation lies in this: a Pareto optimal set very often contains more than one solution.

In order to overcome this challenge, Bekker & Aldrich (2010) suggest finding a near optimal set of parameter vectors that would result in a Pareto optimal set. This is done using inverting histograms drawn up from the *Elite*. The term *Elite* refers to a set comprising of solutions that have ranked in the first three non-dominated fronts in one of the last few generations.

The MOO CEM is inherently elitist as the best solutions of the current generation are automatically kept for the next generation.

As with the CEM, pdfs for each variable are independent of pdfs for other decision variables. Using the *Elite*, a histogram is constructed for each variable. The histogram

expresses the number of times that values of variable i that are part of the *Elite* set have fallen into automatically determined bins. Such a histogram is shown in Figure 3.1. There are seven bins in total. The first bin has a frequency of zero as it stretches from the lower limit of the range of the variable to the minimum value observed for that variable. The last bin is similar and contains values between the maximum value observed for the variable and the upper limit of the range of the variable. Due to rounding errors the last bin might often not be empty. The remaining five bins are of equal size.

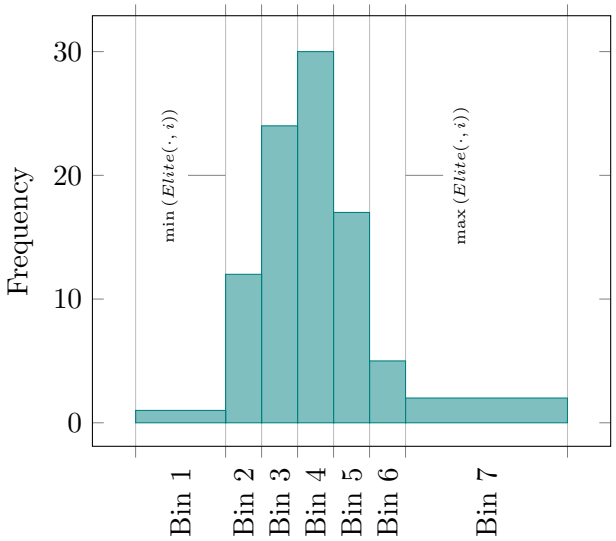
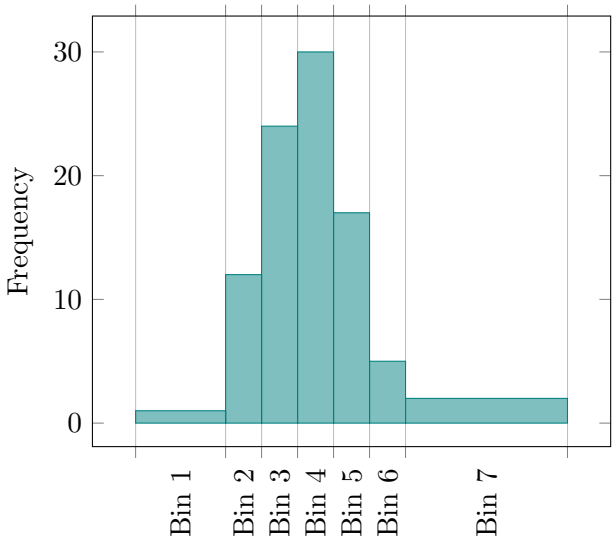


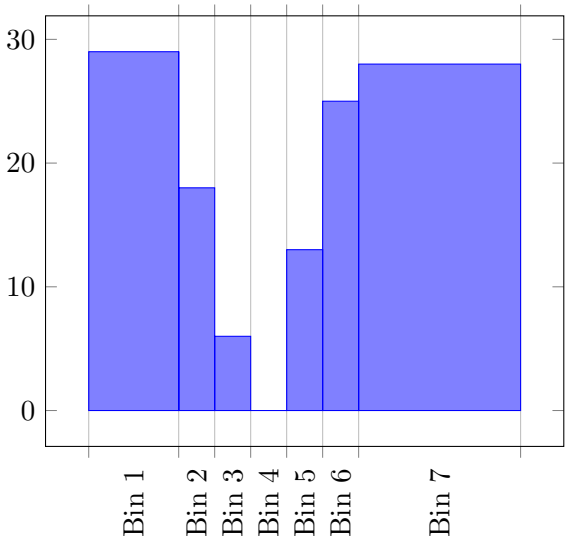
Figure 3.1: Example of a histogram for the decision variable x_i.

The MOO CEM (shown in Algorithm 2) makes use of two mechanisms to maintain population diversity. First, the *Elite* is accumulated over a few generations. At each generation the first three non-dominated fronts, and not just the first non-dominated front, are added to the existing *Elite*. After a set number of generations, the entire set of accumulated solutions is ranked and the first two non-dominated fronts are retained.

In addition to this, inversion of the histograms is also used to avoid early convergence on a local minimum (or maximum when maximising). With some probability p_h (typically $0.1 \leq p_h \leq 0.3$) the histogram for variable i is inverted. Inversion involves subtracting the frequencies of histogram i from the maximum frequency occurring in histogram i. The inverted histogram is shown in Figure 3.2.



(a) Histogram to be inverted



(b) Inverted histogram

Figure 3.2: Example of an inverted histogram for the decision variable x_i.

New decision variable vectors are created by scaling the frequencies of the histogram. If 20% of the *Elite* values for variable i fell between a and b, then 20% of the values for variable i of the new decision variable vectors will fall between a and b. To ensure that values fall exactly between a and b (b and c, etc.), Bekker & Aldrich (2010) use truncated normal distributions. The mean of each bin serves as mean to the truncated

normal distribution.

Algorithm 2 The basic MOO CEM algorithm.

Choose an initial vector $\mathbf{v_0}$.
Create an initial population using truncated normal distribution with means $\mathbf{v_0}$.
Evaluate initial population.
Rank initial population to find *Elite* – the *Elite* is made up of the first three non-dominated fronts.
While stopping criteria not met,
  For each variable i
    Create histogram i using *Elite*.
    If $rand(0,1) \leq p_h$
      Invert histogram.
    end if
    For each bin j in histogram i
      Create $\lfloor \frac{Frequency\ of\ bin \times Population\ size}{Total\ size\ of\ Elite} \rfloor$ new decision variable vectors using a truncated normal distribution with μ = *mean of bin* j and σ = *UB of bin* j − *LB of bin* j
    end for
  end for
  Evaluate new decision variable vectors.
  Rank new decision variable vectors.
  Add best ranked decision variable vectors to *Elite*.
  If number of generations equals some predetermined value,
    Rank *Elite*.
    Keep only the first two non-dominated fronts.
  end if
end while
Rank *Elite*.
Keep only the first non-dominated front.

3.2 The multi-objective covariance matrix adaptation evolution strategy

The multi-objective covariance matrix adaptation evolution strategy (MO-CMA-ES) is one of two algorithms selected from literature for its reported ability to manage relationships between variables (the other being Pareto differential evolution (PDE)).

In this section the selection of the MO-CMA-ES is motivated, background to the

single-objective covariance matrix adaptation evolution strategy (CMA-ES) is provided, and the CMA-ES and MO-CMA-ES algorithms are discussed.

3.2.1 Why investigate the performance of the MO-CMA-ES?

The MO-CMA-ES was introduced in Igel *et al.* (2007a). Although it is older than the MOO CEM, it can still be considered a novel algorithm. The MO-CMA-ES remains relatively uninvestigated, despite the promise shown by two variants of the MO-CMA-ES in Igel *et al.* (2007a) where it was compared to the NSGA-II and the non-dominated sorting differential evolution algorithm (NSDE). Furthermore, the general consensus is that its single-objective counterpart, the CMA-ES, is very effective (Boussaïd *et al.*, 2013).

Investigating the MO-CMA-ES ties in directly with the main research question: the algorithm is reported to keep track of relationships between decision variables. It is therefore a good candidate for comparison to the MOO CEM.

3.2.2 The covariance matrix adaptation evolution strategy for single-objective optimisation

The presentation of the covariance matrix adaptation evolution strategy (CMA-ES) differs slightly from the presentation of the CEM (in Section 3.1.2) and differential evolution (in Section 3.3.2). This is because the development of the CMA-ES is clearly visible when looking at preceding papers by the same authors. The researcher considers the CMA-ES to be more easily understandable in light of its background. A short summary of the research leading up to the CMA-ES is presented next, following which the CMA-ES for single-objective optimisation will briefly be explained.

3.2.2.1 Background to the covariance matrix adaptation evolution strategy for single-objective optimisation

The covariance matrix adaptation evolution strategy (CMA) was first presented in Hansen & Ostermeier (1996). It is the result of a research effort that started with Ostermeier *et al.* (1994a) presenting what they refer to as a 'derandomised approach to self adaptation of evolution strategies'. The work builds on mutative step size control: a method developed in the 1970s, based on the idea that the mutation step size parameter in an evolution strategy should not simply be set beforehand, but rather be adjusted

by the algorithm itself as it progresses. The original mutative step size control rule involves taking the product of four values, of which three have elements of randomness. For one offspring $\mathbf{x}^{(t+1)}$ mutated from one parent $\mathbf{x}^{(t)}$,

$$\mathbf{x}^{(t+1)} = \mathbf{x}^{(t)} + c_1 c_2 c_3 \mathbf{z}, \tag{3.15}$$

where all vectors are of size V, and V indicates the number of decision variables.

In (3.15), c_1 is a constant, whereas c_2 and c_3 both follow some distribution and each element of $\mathbf{z}$ is drawn from a $N(0, 1)$ distribution. 'Derandomised' refers to the fact that Ostermeier *et al.* (1994a) adapted the mutative step size control rule to involve less randomness. This allows the algorithm to benefit from updating the step size: changes to the step size would correspond directly to the size of the mutations that follow.

Ostermeier *et al.* (1994b) expanded on Ostermeier *et al.* (1994a) by changing the step size update rule to take into account information from not only the previous generation, but all preceding generations. This concept is referred to as using an *evolution path*.

Continuing this line of research, Hansen *et al.* (1995) compared three adaptation strategies. In the previous works (discussed above), univariate normal distributions are used for mutation. However, in this paper, Hansen *et al.* (1995) started using the multivariate normal distribution to do mutation. The multivariate normal distribution is a generalisation of the univariate normal distribution. It is denoted as $N(\mu, \mathbf{C})$ with μ representing a vector of means of all variables and $\mathbf{C}$ denoting a $V \times V$ covariance matrix. Focus falls on finding a mechanism for adapting $\mathbf{C}$ and as a result the mutation distributions of evolution strategies. Such a mechanism should work independently of the coordinate system of the decision variables as the quality of the mutation distribution should not depend on this specific position of decision variables in the coordinate system (Hansen & Ostermeier, 1996; Hansen *et al.*, 1995). This will ensure invariance with respect to the rotation of objective functions which will make the evolution strategy more suited to solve problems with complex fitness functions (Hansen & Ostermeier, 1996). Hansen *et al.* (1995) refer to the most successful of these strategies as the generating set adaptation (GSA). The GSA is able to reliably adapt the mutation distribution independent of the given coordinate system.

Hansen & Ostermeier (1996) introduced the covariance matrix adaptation (CMA) as a mechanism for adapting mutation distributions. This built directly on both Ostermeier *et al.* (1994b), with the use of an evolution path, and Hansen *et al.* (1995) because the CMA is very similar to the GSA. There are two differences between the CMA and the GSA (Hansen & Ostermeier, 1996):

- While both make use of a factor called the global step size, the global step size for the CMA can be calculated from other information used in the CMA, whereas the global step size for the GSA is a random factor.
- For its evolution path, the GSA weights information from all generations equally. The CMA, on the other hand, uses exponentially decreasing weights: the older the information, the less importance it carries.

Hansen & Ostermeier (1996) found that the CMA outperforms the GSA and should therefore be preferred.

Where Hansen & Ostermeier (1996) used $(1, 10)$-evolution strategies (strategies where one parent produces ten offspring) for their experiments, Hansen & Ostermeier (1997) combine intermediate recombination with the CMA in order to improve the robustness of the CMA-ES.

Hansen & Ostermeier (2001) combined the CMA with weighted recombination. This paper was found to be the most thorough presentation of the CMA-ES.

Whereas previous implementations were all serial, Hansen *et al.* (2003) adapt the CMA-ES so that it can be implemented for parallel execution. It should be noted that there are some problems with the adjustment of the global step size when population size is greater than $10V$.

3.2.2.2 Basic algorithm for the covariance matrix adaptation evolution strategy for single-objective optimisation

As discussed in Chapter 2, mutation forms a key part of any evolution strategy. Many evolution strategies have some mechanism which automatically adapts the mutation distribution, be it in a random (mutative) or controlled (calculated) fashion. The CMA is essentially a way of controlling the adaptation of the mutation distribution in order to favour previously successful mutation values (Hansen & Ostermeier, 2001).

3.2 The multi-objective covariance matrix adaptation evolution strategy

For maximum effectiveness, the mutation distribution should not be dependent on the coordinate system of the decision variables (Hansen & Ostermeier, 1996, 2001), and should take into account information gained in all the preceding generations (Hansen & Ostermeier, 1996, 2001; Ostermeier *et al.*, 1994b).

The CMA-ES as laid out below (and shown in Algorithm 3) is mostly from Hansen & Ostermeier (2001) which uses weighted recombination. Note that, for better flow from this section to Section 3.2.3, weights are assumed to be equal and all information regarding weights (such as constants and subscripts) presented in Hansen & Ostermeier (2001) are disregarded.

Algorithm 3 The basic CMA-ES algorithm.

1: Set the number of offspring λ, and the number of parents ν.
2: Initialise constants.
3: Initialise evolution paths $\mathbf{p}_c^{(0)}$ and $\mathbf{p}_s^{(0)}$, global step size $\sigma^{(0)}$, covariance matrix $\mathbf{C}^{(0)}$ and parent vector $\langle \mathbf{x} \rangle^{(0)}$.
4: Find $\mathbf{B}^{(0)}$ and $\mathbf{D}^{(0)}$ from $\mathbf{C}^{(0)}$.
5: Create a set of offspring of size λ using (3.16).
While stopping criteria not met,
6: Evaluate population offspring.
7: Choose the best ν members of population (comprising of both parents and offspring).
8: Calculate evolution paths $\mathbf{p}_c^{(t+1)}$ (3.19) and $\mathbf{p}_s^{(t+1)}$ (3.21), global step size $\sigma^{(t+1)}$ ((3.22)), covariance matrix $\mathbf{C}^{(t+1)}$ (3.20) and the mean of the ν best parents $\langle \mathbf{x} \rangle^{(t+1)}$.
9: Find $\mathbf{B}^{(t+1)}$ and $\mathbf{D}^{(t+1)}$ from $\mathbf{C}^{(t+1)}$.
10: Create a set of offspring of size λ using (3.16).
end while.

For each offspring $\mathbf{x}_k^{(t+1)}$ with $k = 1, \ldots, \lambda$, the update rule for the standard CMA-ES is (Hansen & Ostermeier, 2001; Igel *et al.*, 2007a)

$$\mathbf{x}_k^{(t+1)} = \langle \mathbf{x} \rangle^{(t)} + \sigma^{(t)} N(\mathbf{0}, \mathbf{C}^{(t)}) \tag{3.16}$$

$$= \langle \mathbf{x} \rangle^{(t)} + \sigma^{(t)} \mathbf{B}^{(t)} \mathbf{D}^{(t)} \mathbf{z}_k^{(t+1)} \tag{3.17}$$

where $\langle \mathbf{x} \rangle^{(t)}$ is the mean of the ν best individuals from the current generation t.

The step size $\sigma^{(t)}$ scales all samples drawn from a multivariate normal distribution $N(\mathbf{0}, \mathbf{C}^{(t)})$. Note that the covariance matrix $\mathbf{C}^{(t)}$ is a symmetric, positive, definite matrix of size $V \times V$. When initialising, $\mathbf{C}^{(0)} = \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. $\mathbf{B}^{(t)}$ and $\mathbf{D}^{(t)}$ can be determined from $\mathbf{C}^{(t)}$

$$\mathbf{C}^{(t)} = \mathbf{B}^{(t)}\mathbf{D}^{(t)}(\mathbf{B}^{(t)}\mathbf{D}^{(t)})^T, \tag{3.18}$$

where $\mathbf{B}^{(t)}$ is an orthogonal $V \times V$ matrix.

The columns of $\mathbf{B}^{(t)}$ are the normalised eigenvectors of the covariance matrix $\mathbf{C}^{(t)}$. $\mathbf{D}^{(t)}$ is a $V \times V$ diagonal matrix. For $i \neq j$, elements of $\mathbf{D}^{(t)}$ equal zero. The diagonal elements of $\mathbf{D}^{(t)}$ $(i = j)$ are the square roots of the eigenvalues of the covariance matrix $\mathbf{C}^{(t)}$. The orthogonal matrix $\mathbf{B}^{(t)}$ is responsible for the rotation of the mutation distribution, whereas $\mathbf{D}^{(t)}$ is responsible for the scaling.

Members of $\mathbf{z}_k^{(t+1)}$ are drawn from independent univariate $(0, 1)$ normal distributions. This is equivalent to sampling from a multivariate $(\mathbf{0}, \mathbf{I})$ normal distribution. From (3.16), note that sampling using $\sigma^{(t)}\mathbf{B}^{(t)}\mathbf{D}^{(t)}\mathbf{z}_k^{(t+1)}$ is equivalent to sampling from $(\mathbf{0}, {\sigma^{(t)}}^2\mathbf{C}^{(t)})$. Figure 3.3 shows how adapting $\mathbf{C}$ influences the multivariate normal pdf from which samples are effectively drawn. The multivariate pdfs shown are for two decision variables, x_1 and x_2. It is assumed that $\sigma = 1$. For the two top figures $\mathbf{C} = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, while $\mathbf{C} = \left(\begin{smallmatrix} 0.25 & 0.3 \\ 0.3 & 1 \end{smallmatrix}\right)$ for the two bottom figures.

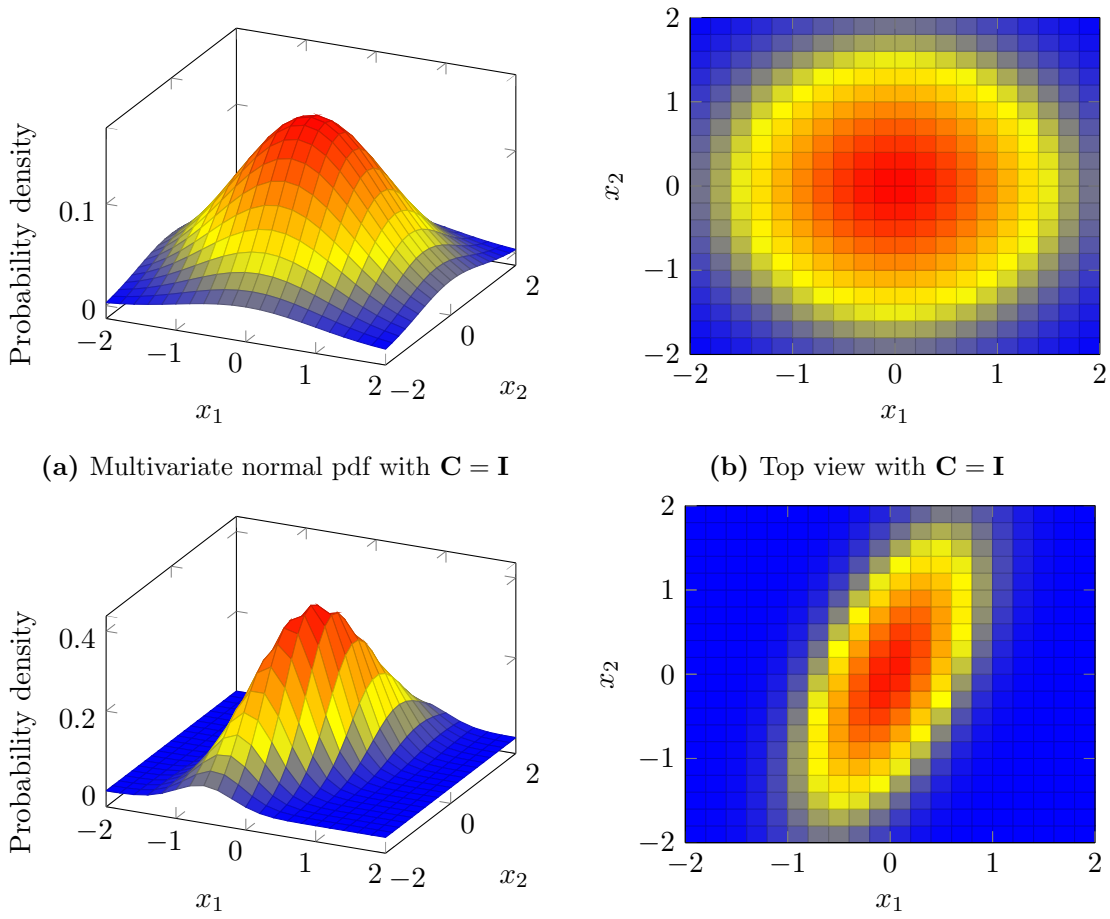
In order to update $\mathbf{C}^{(t)}$, the CMA makes use of an evolution path as suggested in Ostermeier *et al.* (1994b). The evolution path is a cumulation of information gained throughout the progress of the algorithm. Instead of weighting information equally, the CMA uses exponential weights with recent information weighing heavier than older information (Hansen & Ostermeier, 1996). The covariance matrix evolution path for the next generation $\mathbf{p}_c^{(t+1)}$ is calculated using the covariance matrix evolution path for the current generation $\mathbf{p}_c^{(t)}$:

$$\mathbf{p}_c^{(t+1)} = (1 - c_p)\mathbf{p}_c^{(t)} + \sqrt{c_p(2 - c_p)}\left(\langle\mathbf{x}\rangle^{(t+1)} - \langle\mathbf{x}\rangle^{(t)}\right), \tag{3.19}$$

where c_p is a constant with a value between 0 and 1. The initial value of the covariance matrix evolution path $\mathbf{p}_c^{(0)}$ equals $\mathbf{0}$. From $\mathbf{p}_c^{(t+1)}$, the covariance matrix for the next generation $\mathbf{C}^{(t+1)}$ is calculated using $\mathbf{C}^{(t)}$:

$$\mathbf{C}^{(t+1)} = (1 - c_c)\mathbf{C}^{(t)} + c_c\mathbf{p}_c^{(t+1)}\left(\mathbf{p}_c^{(t+1)}\right)^T, \tag{3.20}$$

where the change rate of the covariance matrix c_c is a constant with a value between 0 and 1.



(a) Multivariate normal pdf with $\mathbf{C} = \mathbf{I}$

(b) Top view with $\mathbf{C} = \mathbf{I}$

(c) Multivariate normal pdf adjusted $\mathbf{C}$

(d) Top view with adjusted $\mathbf{C}$

Figure 3.3: The effect of adapting $\mathbf{C}$ on the multivariate normal pdf from which samples for the CMA-ES are effectively drawn.

A second evolution path, the step size adaptation path $\mathbf{p}_s$, is used to update the global step size σ (Hansen & Ostermeier, 2001),

$$\mathbf{p}_s^{(t+1)} = (1 - c_s)\mathbf{p}_s^{(t)} + \sqrt{c_s(2 - c_s)}\mathbf{B}^{(t)} \left(\mathbf{D}^{(t)}\right)^{-1} \left(\mathbf{B}^{(t)}\right)^{-1} \left(\langle \mathbf{x} \rangle^{(t+1)} - \langle \mathbf{x} \rangle^{(t)}\right), \quad (3.21)$$

where c_s is a constant with a value between 0 and 1. The initial value of the step size adaptation path $\mathbf{p}_s^{(0)}$ equals $\mathbf{0}$. The global step size is updated with (Hansen & Ostermeier, 2001)

$$\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{||\mathbf{p}_s^{(t+1)}|| - \hat{\chi}}{d_s \hat{\chi}}\right). \tag{3.22}$$

The damping parameter $d_s \geq 1$ determines the range of σ, $||\mathbf{p_s}^{(t+1)}||$ is the norm of $\mathbf{p_s}^{(t+1)}$, and $\hat{\chi}$ is the expected value of the length of a ($\mathbf{0}$, $\mathbf{I}$) normally distributed random vector. It is calculated as

$$\hat{\chi} = \frac{\sqrt{2}\Gamma\left(\frac{V+1}{2}\right)}{\Gamma\left(\frac{V}{2}\right)}, \tag{3.23}$$

and can be approximated as

$$\hat{\chi} \approx \sqrt{V}\left(1 - \frac{1}{4V} + \frac{1}{21V^2}\right). \tag{3.24}$$

3.2.3 The covariance matrix adaptation evolution strategy for multi-objective optimisation

As mentioned in Section 3.2.1, the covariance matrix adaptation evolution strategy for multi-objective optimisation (MO-CMA-ES) was introduced in Igel *et al.* (2007a). The CMA-ES is adapted so that, instead of using the mean of the best ν parents as above, it works with a single parent: the best solution from the previous generation. To accommodate this change, the update rules for the step size and covariance matrix are adapted. This elitist CMA-ES forms the basis of the MO-CMA-ES.

The MO-CMA-ES laid out by Igel *et al.* (2007a) is made up of multiple elitist CMA-ESs – each with one parent generating one offspring. Although it is easy to expand this strategy so that each parent produces more than one offspring, this study uses the "one parent producing one offspring" strategy as this is how the basic MO-CMA-ES functions.

The update rule for the evolution path of global step size p_s differs from (3.21) and works as follows:

$$p_s^{(t+1)} = (1 - c_s)p_s^{(t)} + c_s S, \tag{3.25}$$

where $S = 1$ if the offspring of the parent in question was ranked (using Pareto ranking) as high or higher than the parent, otherwise $S = 0$. Note that p_s is scalar

here compared to (3.21) where it was a vector. The global step size is updated using (Igel *et al.*, 2007a)

$$\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{p_s - p_g}{d_s(1-p_g)}\right). \tag{3.26}$$

The goal value of the step size evolution path p_g lies between zero and 0.5. This combination of evolution path and global step size updating ensures that σ increases if it is associated with a high success rate and decreases if not.

Similarly, the update rules for the covariance matrix differs from (3.20). If the value of p_s is smaller than some threshold value p_t, then

$$\mathbf{p}_c^{(t+1)} = (1-c_p)\mathbf{p}_c^{(t)} + \sqrt{c_p(2-c_p)}\left(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\right) \tag{3.27}$$

and

$$\mathbf{C}^{(t+1)} = (1-c_c)\mathbf{C}^{(t)} + c_c\mathbf{p}_c^{(t+1)}\left(\mathbf{p}_c^{(t+1)}\right)^T. \tag{3.28}$$

On the other hand, if $p_s \geq p_t$, then

$$\mathbf{p}_c^{(t+1)} = (1-c_p)\mathbf{p}_c^{(t)} \tag{3.29}$$

and

$$\mathbf{C}^{(t+1)} = (1-c_c)\mathbf{C}^{(t)} + c_c\left(\mathbf{p}_c^{(t+1)}\left(\mathbf{p}_c^{(t+1)}\right)^T + c_p(2-c_p)\mathbf{C}\right). \tag{3.30}$$

It is important to note that, although the update rules work based on whether or not an offspring outperforms its parent, elitism is incorporated in the MO-CMA-ES in that the next generation for the MO-CMA-ES is made up of the λ_M best individuals of the current population (offspring and parents considered together).

The parameter λ_M denotes the number of CMA-ESs that will make up the MO-CMA-ES. If each parent creates one offspring, λ_M is equal to half the size of the population. This relatively large number of parents helps maintain population diversity as the parents are often made up of more than one non-dominated front.

As each parent k produces only one offspring, the update rule for generating new offspring is:

$$\mathbf{x}_k^{(t+1)} = \mathbf{x}_k^{(t)} + \sigma^{(t)} N(\mathbf{0}, \mathbf{C}^{(t)}) \tag{3.31}$$

$$= \mathbf{x}_k^{(t)} + \sigma^{(t)} \mathbf{B}^{(t)} \mathbf{D}^{(t)} \mathbf{z}_k^{(t+1)}. \tag{3.32}$$

Algorithm 4 The basic MO-CMA-ES algorithm.

1: Set the number of offspring and parents for multi-objective optimisation λ_M.
2: Initialise constants.
For $k = 1$ to λ_M
3: Initialise evolution paths $\mathbf{p}_{c_k}^{(0)}$ and $p_{s_k}^{(0)}$, global step sizes $\sigma_k^{(0)}$, covariance matrices $\mathbf{C}_k^{(0)}$ and parent vectors $\mathbf{x}_k^{(0)}$.
4: Find $\mathbf{B}_k^{(0)}$ and $\mathbf{D}_k^{(0)}$ from $\mathbf{C}_k^{(0)}$.
end for
While stopping criteria not met,
5: Create a set of offspring: one offspring k for each parent k using (3.31).
6: Evaluate offspring.
7: Pareto rank the population (comprising both offspring and parents).
For $k = 1$ to λ_M
8: Calculate evolution paths $\mathbf{p}_c^{(t+1)}$ – (3.27) or (3.29) – and $\mathbf{p}_s^{(t+1)}$ (3.25), global step size $\sigma^{(t+1)}$ (3.26), covariance matrix $\mathbf{C}^{(t+1)}$ – (3.28) or (3.30) – and the mean of the μ best parents $\langle \mathbf{x} \rangle^{(t+1)}$.
9: Find $\mathbf{B}_k^{(t+1)}$ and $\mathbf{D}_k^{(t+1)}$ from $\mathbf{C}_k^{(t+1)}$.
end for
10: Choose the best λ_M members of population to make up parents for next generation. (Use indices to keep track of which evolution paths, step sizes and covariance matrices go with which members of the population).
end while.
11: Return non-dominated solutions found during the search.

Igel *et al.* (2007a) presents two variations on the basic MO-CMA-ES described here. Both these variations make use of a secondary selection criterion in order to rank solutions on the same level of non-dominance. The first variation uses crowding distance, whereas the second makes use of the contributing hypervolume in order to rank solutions further.

For the purposes of this study, the basic MO-CMA-ES (as shown in Algorithm 4) is preferred to the two variations. This is because the more complicated selection mechanisms of the two variations might make it more difficult to draw conclusions about the effect of the MO-CMA-ES taking into account relationships between decision

variables. For the same reason, alternative selection strategies suggested by Igel *et al.* (2007b), Voß *et al.* (2010), and Loshchilov *et al.* (2011) were not included.

3.3 Pareto differential evolution

Pareto differential evolution (PDE) is the second optimisation algorithm selected from literature based on its reported ability to handle relationships between decision variables.

This section motivates the selection of PDE, discusses its single-objective counterpart, differential evolution (DE), and lays out the PDE algorithm.

3.3.1 Why investigate Pareto differential evolution?

Differential evolution (DE) is a very popular algorithm for solving continuous optimisation problems (Boussaïd *et al.*, 2013). It has the advantage of having very few input parameters (Boussaïd *et al.*, 2013).

Pareto differential evolution (PDE), as presented by Abbass *et al.* (2001), is one of a few multi-objective versions of DE. Other notable multi-objective differential evolution algorithms include: another version of Pareto differential evolution by Madavan (2002), Pareto-based multi-objective differential evolution (MODE) presented by Xue *et al.* (2003), a multi-objective differential evolution algorithm by Babu & Jehan (2003), and the NSDE presented by Iorio & Li (2005).

The NSDE variation has been used to successfully solve a set of test problems (the R problems, discussed in Chapter 4) included in the test suite for this study. Also recall from Section 3.2.1 that the NSDE was compared to the MO-CMA-ES. It was selected because it was reported to be able to solve problems with relationships between decision variables (Igel *et al.*, 2007a).

Unfortunately all the multi-objective variations of DE cannot be included in this study. The PDE algorithm as presented by Abbass *et al.* (2001) was preferred to the other variations because it was the first multi-objective optimisation DE algorithm to be presented. The differences between the variations are not such that a superior performing variation could be identified from literature.

3.3.2 Differential evolution for single-objective optimisation

Stork & Price (1997) presented the differential evolution (DE) algorithm (shown in Algorithm 5). The algorithm was developed in an effort to satisfy four main requirements:

- An optimisation algorithm must be able to handle non-differentiable, nonlinear and multimodal functions as well as combinations thereof.
- It must be possible to parallelise the optimisation algorithm.
- The optimisation algorithm should be easy to use, requiring few control parameters.
- The optimisation algorithm should converge to the global minimum consistently.

Algorithm 5 The basic DE algorithm.

Set the number of offspring λ, and the number of parents ν (equal to λ).
Initialise constants.
Randomly select an initial set of offspring of size λ.
Evaluate offspring.
While stopping criteria not met,
  For $i = 1$ to λ
    Randomly select r_1, r_2 and r_3 with $i \neq r_1 \neq r_2 \neq r_3$.
    Use (3.33) to create a mutated vector.
    Use (3.34) to create an offspring $\mathbf{x}_i^{(t+1)}$.
    If $\mathbf{x}_i^{(t+1)}$ is better than $\mathbf{x}_i^{(t)}$.
      $\mathbf{x}_i^{(t+1)}$ makes out part of the population for next generation.
    else
      $\mathbf{x}_i^{(t+1)} \leftarrow \mathbf{x}_i^{(t)}$, the parent for this generation outlives its offspring to form part of the next population.
    end if
  end for
end while.

The inspiration for DE came from the Nelder and Mead method (introduced in Nelder & Mead (1965)). The Nelder and Mead method is essentially a local optimisation algorithm, and its main influence on DE is the use of information gained from the search space to adapt members of the population (Stork & Price, 1997). (Compare this to an algorithm that adapts members of the population using a purely random mutation variable.)

Compared to the CEM and the CMA, there is little background information or theory required to understand DE. DE comprises three major, classic evolutionary computation operators: mutation, recombination and selection. Mutation in DE involves adding a weighted difference between two parent vectors to a third parent:

$$\mathbf{x}_m = \mathbf{x}_{r_1}^{(t)} + c_m(\mathbf{x}_{r_2}^{(t)} - \mathbf{x}_{r_3}^{(t)}), \tag{3.33}$$

where r_1, r_2 and r_3 are randomly chosen indices to parents from the previous generation.

If the index to the main parent is i, then $i \neq r_1 \neq r_2 \neq r_3$. The weight of the difference between $\mathbf{x}_{r_2}^{(t)}$ and $\mathbf{x}_{r_3}^{(t)}$ is denoted by c_m, a constant with a value between 0 and 2.

Recombination occurs when this mutated vector is mixed with another parent vector (the main parent $\mathbf{x}_i^{(t)}$) to form an offspring $\mathbf{x}_i^{(t+1)}$. For the j_{th} element of $\mathbf{x}$,

$$\mathbf{x}_{i,j}^{(t+1)} = \begin{cases} \mathbf{x}_{mutated,j} & J \leq c_r \text{ or } j = J_V \\ \mathbf{x}_{i,j}^{(t)} & \text{else} \end{cases} \tag{3.34}$$

where J comes from a uniform (0,1) distribution and c_r is a recombination constant (ranging from 0 to 1).

J_V is drawn uniformly to fall between 1 and V. Its function is to ensure that at least one element of population member i is mutated.

Selection is simple: if the offspring $\mathbf{x}_i^{(t+1)}$ outperforms the main parent $\mathbf{x}_i^{(t)}$, the offspring would be included as parent for the next generation, otherwise the main parent would be retained as part of the population for the next generation (Stork & Price, 1997).

3.3.3 Differential evolution for multi-objective optimisation

Probably due to its combination of effectiveness and simplicity, several authors have extended the DE algorithm for single-objective optimisation for use on multi-objective problems. Four early extensions were the works by Abbass *et al.* (2001), Madavan (2002), Xue *et al.* (2003) and Babu & Jehan (2003).

This study uses the work by Abbass *et al.* (2001) – their Pareto differential evolution (PDE) is the first multi-objective DE algorithm found in literature. Pseudocode for the PDE algorithm is shown in Algorithm 6.

Algorithm 6 The basic PDE algorithm.

Set the number of offspring λ.
Initialise constants.
Randomly select an initial set of offspring of size λ.
While stopping criteria not met,
    Evaluate offspring.
    Pareto rank offspring and find the non-dominated solutions.
    While number of non-dominated solutions $> c_v$
        Remove the point that has the lowest average Euclidean distance between itself and the two points closest to it in the decision space from the set of possible parents.
    end while
    For $i = 1$ to λ
        Randomly select r_1, r_2 and r_3 with $r_1 \neq r_2 \neq r_3$ from possible parents.
        Use (3.35) to create a mutated vector.
        Use (3.36) to create an offspring $\mathbf{x}_i^{(t+1)}$.
        If $\mathbf{x}_i^{(t+1)}$ dominates $\mathbf{x}_{r_1}^{(t)}$.
            $\mathbf{x}_i^{(t+1)}$ makes out part of the population for next generation.
        else
            $\mathbf{x}_i^{(t+1)} \leftarrow \mathbf{x}_{r_1}^{(t)}$, the parent for this generation outlives its offspring to form part of the next population.
        end if
    end for
end while.
Return non-dominated solutions found during the search.

The following differentiates the single-objective DE by Stork & Price (1997) from the PDE presented by Abbass *et al.* (2001):

1. The DE uses uniform distributions for initialisation, whereas the PDE uses univariate normal distributions.

2. The constant c_m in the DE algorithm is replaced by a normal (0,1) random variable in the PDE algorithm.

3. For DE, an offspring outperforms its parent if it has a lower function value when minimising (and a higher function value for maximising). The multi-objective version makes use of elitism and Pareto ranking is used to determine whether or not an offspring outperforms its parent. The non-dominated front makes up the parents for the next generation.

4. The PDE algorithm only uses non-dominated solutions for each succeeding generation. Because of this, the number of parents is not always equal to the number of offspring required. To make up for this Abbass *et al.* (2001) do not select a main parent in the same way as is done by DE. Rather, only the three randomly selected parents are used to form an offspring with the first parent $\mathbf{x}_{r_1}^{(t)}$ acting as main parent.

5. In order to maintain diversity, Abbass *et al.* (2001) thin out the non-dominated set if it becomes larger than some predetermined value c_v. This is done using the average nearest neighbour distances of the members of the non-dominated set.

Due to these differences the mutation operator differs slightly from (3.33):

$$\mathbf{x}_m = \mathbf{x}_{r_1}^{(t)} + N(0,1)(\mathbf{x}_{r_2}^{(t)} - \mathbf{x}_{r_3}^{(t)}), \tag{3.35}$$

and the recombination operator differs from (3.34):

$$\mathbf{x}_{i,j}^{(t+1)} = \begin{cases} \mathbf{x}_{mutated,j}, & J \leq c_r \text{ or } j = J_V \\ \mathbf{x}_{r_1,j}^{(t)}, & \text{else} \end{cases} . \tag{3.36}$$

3.4 The two hybrid algorithms

In addition to the three established multi-objective algorithms discussed above (MOO CEM, MO-CMA-ES and PDE), the researcher developed two hybrid algorithms (Hybrid 1 and Hybrid 2). Both these algorithms exist due to research curiosity, as will be expanded on below, and make use of the MOO CEM to some extent. This is because the MOO CEM forms the basis for the research. Comparing these algorithms to the three established algorithms provides information about not only their mechanisms but also the workings of the other three algorithms.

3.4.1 Combining the cross-entropy method with clustering in the search space

As discussed earlier, both the CEM and the MOO CEM assume that decision variables are independent of one another. The research question to be answered by this work relates to whether or not an algorithm which provides for some form or relationship between variables would perform better on different test problems than others.

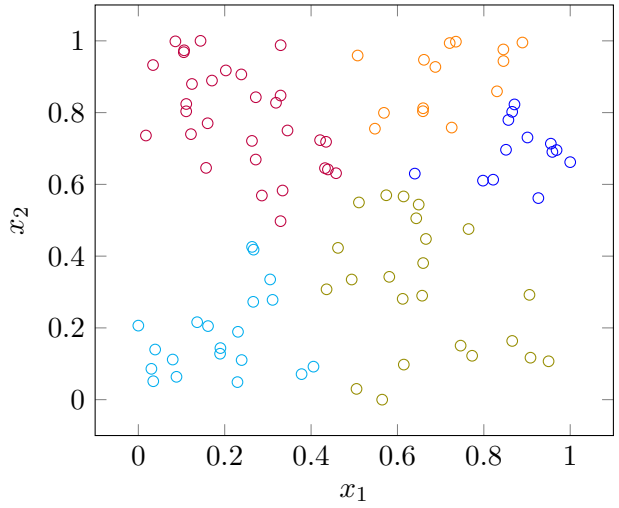


Figure 3.4: A sample of clusters produced by the k-means algorithm.

The idea of using clustering in the search space builds on this: points that are close together in the search space, are close in terms of all (or at least most) of the variables, as illustrated in Figure 3.4. There are two variables x_1 and x_2, a hundred data points and five clusters. Data for both variables were scaled to be between 0 and 1.

Would clustering the *Elite* before creating histograms for each cluster be more effective than the original MOO CEM? Or would this rob the population of too much diversity?

Hybrid 1 (shown in Algorithm 7) simply clusters the decision variables of the *Elite* before creating histograms and offspring for each cluster separately. In order to try and preserve some diversity, each cluster has an equal amount of offspring irrespective of its size.

The clustering is done using the Matlab k-means clustering algorithm. This clustering method was the fastest of the methods that were experimented with. Even though more accurate algorithms might exist, accuracy tends to come at a cost. Keeping in mind that clustering has to happen tens or hundreds or thousands of times each time the algorithm runs, the trade-off between accuracy and time does not favour accuracy in this case. To some extent the lack of accuracy might be a bonus: it might help to maintain some diversity. If a cluster contains some points that would perhaps have fitted better with another cluster, the information available to create a new set of offspring for that cluster is less local than what would have been the case without the

said "misplaced point".

Algorithm 7 The basic Hybrid 1 algorithm.

1: Choose an initial mean vector $\mathbf{v_0}$.
2: Create an initial population using truncated normal distributions with means $\mathbf{v_0}$.
3: Evaluate initial population.
4: Rank initial population to find *Elite* – the *Elite* is made up of the first three non-dominated fronts.
While stopping criteria not met,
5: Cluster decision variables in *Elite*,
For each cluster k,
For each variable i,
6: Create histogram i using cluster k.
If $rand(0,1) \leq p_h$
7: Invert histogram.
end if
For each bin j in histogram i,
8: Create $\lfloor \frac{Frequency\ of\ bin \times Population\ size}{Total\ size\ of\ Elite \times Number\ of\ clusters} \rfloor$ new decision variable vectors using a truncated normal distribution with μ = *mean of histogram* i and σ = *UB of bin* j − *LB of bin* j.
end for
end for
end for
9: Evaluate new decision variable vectors.
10: Rank new decision variable vectors.
11: Add best ranked decision variable vectors to *Elite*.
If number of generations equals some predetermined value,
12: Rank *Elite*.
13: Keep only the first two non-dominated fronts.
end if
end while
14: Rank *Elite*.
15: Keep only the first non-dominated front.

3.4.2 Hybrid 2: combining the cross-entropy method and covariance matrix adaptation

Early runs of the MO-CMA-ES and MOO CEM algorithms indicated that the performance of the MO-CMA-ES decreased as the number of variables increased, whereas the performance for the MOO CEM was not as affected by an increase in the number

of variables. Early runs, on small numbers of variables, also indicated that the MO-CMA-ES outperformed the MOO CEM in terms of the hypervolume (see Chapter 7) achieved.

As mentioned above, Hybrid 2 (shown in Algorithm 8) is the product of research curiosity: what would happen if we combined the two algorithms? How much would be sacrificed or gained in terms of performance? How much would be sacrificed or gained in terms of run time?

The original Hybrid 2 used the MOO CEM to determine means for the multivariate normal distribution employed by the MO-CMA-ES. Initial experiments on the continuous test problems (discussed in Chapter 4) showed that this combination of the two algorithms showed little promise.

The version of Hybrid 2 presented here outperformed its original version on early test runs. It is quite simple: half the population of each generation is created using the inverting histograms from the MOO-CEM, the other half using the MO-CMA-ES.

3.5 Conclusion: The multi-objective optimisation algorithms under investigation

This chapter focused on the algorithms selected for comparison. Single-objective versions (the CEM, CMA-ES and DE algorithms) of the selected multi-objective algorithms were laid out, before each of the primary algorithms were discussed. In addition to the three multi-objective algorithms found in literature (MOO CEM, MO-CMA-ES and PDE), two hybrid algorithms (Hybrid 1 and Hybrid 2) were also presented. These hybrids are the product of research curiosity and were developed in response to specific questions that came up as the researcher was working on the study.

The next chapter will focus on the unconstrained, continuous test problems used for this study. Chapter 5 will discuss the static combinatorial problem investigated, and Chapter 6 will lay out the dynamic, stochastic problem studied.

Algorithm 8 The basic Hybrid 2 algorithm.

Choose an initial mean vector $\mathbf{v_0}$.
Set the number of offspring and parents for multi-objective optimisation $\lambda_M = 0.5N$.
Initialise constants.
For $k = 1$ to λ_M
    Initialise evolution paths $\mathbf{p}_{c_k}^{(0)}$ and $p_{s_k}^{(0)}$, global step sizes $\sigma_k^{(0)}$, covariance matrices $\mathbf{C}_k^{(0)}$ and parent vectors $\mathbf{x}_k^{(0)}$.
    Find $\mathbf{B}_k^{(0)}$ and $\mathbf{D}_k^{(0)}$ from $\mathbf{C}_k^{(0)}$.
end for
Create an initial population using truncated normal distributions with means $\mathbf{v_0}$.
Evaluate initial population.
Rank initial population to find *Elite* – the *Elite* is made up of the first three non-dominated fronts.
While stopping criteria not met,
    For each variable i
        Create histogram i using *Elite*.
        If $rand(0,1) \leq p_h$
            Invert histogram.
        end if
        For each bin j in histogram i
            Create $\lfloor \frac{0.5 Frequency\ of\ bin \times N}{Total\ size\ of\ Elite} \rfloor$ offspring using a truncated normal distribution with $\mu = mean\ of\ histogram\ i$ and $\sigma = UB\ of\ bin\ j - LB\ of\ bin\ j$.
        end for
    end for
    For $k = 1$ to λ_M
        Create a set of offspring: one offspring k for each parent k using (3.31).
    end for
    Evaluate offspring.
    Rank population (parents and offspring).
    For $k = 1$ to λ_M
        Calculate evolution paths $\mathbf{p}_c^{(t+1)}$ – (3.27) or (3.29)– and $\mathbf{p}_s^{(t+1)}$ (3.25), global step size $\sigma^{(t+1)}$ (3.26), covariance matrix $\mathbf{C}^{(t+1)}$ – (3.28) or (3.30) – and the mean of the μ best parents $\langle \mathbf{x} \rangle^{(t+1)}$.
        Find $\mathbf{B}_k^{(t+1)}$ and $\mathbf{D}_k^{(t+1)}$ from $\mathbf{C}_k^{(t+1)}$.
    end for
    Add best ranked offspring to *Elite*.
    If number of generations equals some predetermined value,
        Rank *Elite*.
        Keep only the first two non-dominated fronts.
    end if
end while
Rank *Elite*.
Keep only the first non-dominated front.

Chapter 4

Continuous optimisation test problems

Previous chapters covered the research rationale, an introduction to multi-objective optimisation, and the algorithms selected for comparison.

This chapter covers the unconstrained, continuous problems that were investigated in this study and some significant characteristics of these problems. The problems come from five test suites found in literature and the chapter is organised accordingly.

4.1 Test problem characteristics

Huband *et al.* (2006) provide a very thorough analysis of test problems found in literature. The analysis is done with regard to what they define as *recommendations* and *features*. Recommendations can be described as characteristics or properties that problems should ideally possess (and be designed to possess) as possessing them is always beneficial. A test problem should (Huband *et al.*, 2006):

1. **... not have extremal decision variables**. *Extremal decision variables* refer to a situation where the optimal solution is located at the edge of the decision variable domain. This should be avoided as such points can be found accidentally by algorithms truncating invalid decision variables that were generated back to their domain limits. Some algorithms reflect invalid decision variables that were generated away from the domain edge. Such algorithms will be unfairly disadvantaged by the presence of extremal decision variables.

2. **... not have medial decision variables**. Evolutionary algorithms that make use of intermediate recombination could be biased toward finding the optimal solution if *medial decision variables* are present. Such decision variables are at an optimal near the midpoint of their range.

3. **... be scalable to any number of decision variables**. Scalability in decision variables allows the algorithm to be tested at various levels of difficulty.

4. **... be scalable to any number of objectives**. Similar to the number of decision variables, the number of objectives also influences the level of difficulty of a test problem.

5. **... have decision variable domains that are dissimilar in magnitude**. This recommendation would make a test problem difficult to solve for algorithms that do not normalise decision variable domains or that do not have another effective mechanism to deal with the possibility of decision variable domains differing in size.

6. **... have trade-off ranges that are dissimilar in magnitude**. The range of a Pareto front is generally not known in advance and the trade-off ranges of the different objectives might differ in size. The aim of this recommendation is to reward algorithms that normalise objective values before executing scaling-dependent steps. Such algorithms will perform better on problems where the trade-off ranges are dissimilar.

7. **... have a known Pareto front**. Performance indicators often require information about the true Pareto front. Without such information it is difficult to independently assess the performance of a given algorithm. In cases where the Pareto optimal front is unknown, estimated Pareto fronts found by two or more algorithms have to be compared. In these instances performance indicator values are not absolute, but relative to the specific Pareto front created.

From the above, it is clear that recommendations can either be adhered to or not when designing a problem.

A feature can be seen as a difficulty presented to the optimiser (Huband *et al.*, 2006). A test problem may have one or more features. The five features identified by Huband *et al.* (2006) are:

1. **The geometry of the Pareto optimal front and set**. A Pareto optimal front can be convex, linear, concave or mixed. It can be *degenerate*, referring to a front that is of lower dimension than the number of objectives minus one. Pareto optimal fronts and sets can be connected or disconnected. An optimal front can be a combination of any of the above. Different geometries or combinations thereof present different challenges to the optimiser.

2. **Decision variable dependencies.** The decision variables of real-world problems are not always independent. Test problems could account for this by incorporating relationships between variables. Objective functions which have relationships between decision variables are referred to as *non-separable*. Conversely, *separable* is used to refer to objective functions where decision variables can be selected independently.

3. **Bias.** This refers to the mapping from the Pareto optimal set to the Pareto optimal front. If the density in the objective space (or fitness space, if applicable) varies greatly while solutions are evenly spread in the decision variable space, the problem has bias. Plotting solutions in the objective space (or fitness space, if applicable) is a good indicator of bias.

4. **Many-to-one mappings.** A *one-to-one* mapping refers to a situation where each point in the objective space (or fitness space, if applicable) corresponds to only one set of values in the decision variable space. *Many-to-one* means that more than one set of parameters can yield the same objective function value. Many-to-one mappings could be more difficult to solve as they involve a choice between two or more sets of decision variables.

5. **Modality.** *Modality* refers to the number of local optima to be found in the objective (or fitness, if applicable) landscape. A *multimodal* objective function has more than one local optimum. Conversely, an objective function is said to be *unimodal* if it has a single optimum. A problem is referred to as multimodal if it has at least one multimodal objective. Real-world problems are often multimodal problems. Such problems are more difficult to solve than their unimodal counterparts.

Other characteristics that could possibly be considered are whether or not a problem is linear, or symmetric, or canonical Van Veldhuizen (1999).

4.2 Selecting a test suite

A test suite should, in principle, include a number of test problems encompassing a wide variety of properties. The nature of multi-objective problems and the number of combinations of properties that exists, make a test suite encompassing all the possible combinations impractical. Huband *et al.* (2006) therefore suggest a set of guidelines for selecting a test suite. These suggestions are briefly outlined below:

1. A test suite should include a few unimodal test problems. This can be used to test convergence velocity under different Pareto optimal geometry and bias circumstances.

2. At least the following main categories of geometries should be tested: degenerate and disconnected Pareto optimal fronts, and disconnected Pareto optimal sets.

3. Most of the problems included in the suite should be multimodal. Some of these multimodal problems should be *deceptive*. A deceptive problem has a deceptive objective function, referring to an objective function having at least two local optima with the majority of the search space not favouring the global optimum. Optimisers can get stuck in these situations, making such problems difficult to solve.

4. Some of the problems should contain relationships between decision variables.

5. Some of the problems should contain decision variable dependencies and be multimodal as such problems offer a good representation of real-world problems.

Van Veldhuizen (1999) suggests that a test suite should contain problems with characteristics similar to those of the problem which an algorithm is meant to solve. He also suggests that part of the test suite should be made up of real-world problems and that problems should vary in difficulty.

Table 4.1: A summary of the analysis (by Huband *et al.* (2006)) of some of the selected test problems.

Name	No extremal variables	No medial variables	Scalable number of variables	Dissimilar variable domains	Dissimilar trade-off ranges	Optima known	Geometry	Separability	Bias	Many-to-one mappings	Modality
MOP 1	T	F	F	-	F	T	Convex	S	Absent	Absent	U
MOP 2	T	F	T	F	F	T	Concave	S	Absent	Absent	U
MOP 3	T	T	F	F	T	F	Disconnected	NS	Absent	Absent	M
MOP 4	T	T	T	F	T	T	Disconnected	S	Absent	Absent	M
MOP 6	F	T	F	F	T	T	Disconnected	S	Absent	Absent	M
ZDT 1	F	T	T	F	F	T	Convex	S	Absent	Absent	U
ZDT 2	F	T	T	F	F	T	Concave	S	Absent	Absent	U
ZDT 3	F	T	T	F	T	T	Disconnected	S	Absent	Absent	M
ZDT 4	T	F	T	F	F	T	Convex	S	Absent	Absent	M
ZDT 6	F	T	T	F	T	T	Concave	S	Present	Present	M
WFG 1	T	T	T	T	T	T	Convex, mixed	S	Present	Possible	U
WFG 2	T	T	T	T	T	T	Convex, disconnected	NS	Absent	Possible	M
WFG 3	T	T	T	T	T	T	Convex, linear, degenerate	NS	Absent	Possible	U
WFG 4	T	T	T	T	T	T	Concave	S	Absent	Possible	M
WFG 5	T	T	T	T	T	T	Concave	S	Absent	Possible	D
WFG 6	T	T	T	T	T	T	Concave	NS	Absent	Possible	U
WFG 7	T	T	T	T	T	T	Concave	S	Possible	Possible	U
WFG 8	T	T	T	T	T	T	Concave	NS	Possible	Possible	U
WFG 9	T	T	T	T	T	T	Concave	NS	Possible	Possible	M,D

4.3 The unconstrained, continuous test suite

Keeping in mind the recommendations for compiling a test suite as laid out above, the test suite for this study comprises five of the multi-objective problems (MOPs) as laid out by Van Veldhuizen (1999), five of the Zitzler-Deb-Thiele (ZDT) problems as laid out by Zitzler *et al.* (2000), five L_1ZDT problems described by Deb *et al.* (2006), four R problems as suggested by Iorio & Li (2006), and three variations of each of the nine walking fish group (WFG) problems (from Huband *et al.* (2005), Huband *et al.* (2006)).

The MOPs, ZDT and WFG problems were part of the analysis by Huband *et al.* (2006). Their characteristics are shown in Table 4.1. T indicates that a recommendation is adhered to (or true), whereas F indicates that it is not adhered to (or false). U indicates that all objective functions are unimodal, while M indicates that at least one objective function is multimodal. A problem is classified as non-separable if at least one objective function is deemed to be non-separable.

A similar summary of the characteristics (compiled as part of this study) of the L_1ZDT and R problems can be found in Tables 4.4 and 4.6.

There are 46 problems in total. Eight of these problems were used by Bekker (2012). The researcher had access to Matlab implementations of these. The remaining 38 problems were implemented in Matlab by the researcher.

The newly implemented problems were validated by looking at the Pareto fronts produced and by comparing the Matlab results with equivalent models the researcher had implemented in Microsoft Excel.

4.3.1 The Van Veldhuizen problems

Van Veldhuizen (1999) compiled a test suite comprising seven unconstrained numeric problems and three numeric problems with side constraints. These problems were drawn from literature and occasionally slightly revised. Five of the unconstrained problems constitute a part of the test suite for this study. These five problems were included in the test suite mainly because, although the MOO CEM was compared against the NSGA-II on these problems, this would be the first comparison of the MOO CEM, MO-CMA-ES and PDE algorithms using these problems. The MOPs are also commonly found in literature.

Table 4.2 overleaf shows the function definitions and decision variable domains for the five MOPs included in the test suite.

Table 4.2: Problem definitions for the MOPs.

Name	Function definitions
	Minimise both
MOP 1	$f_1(x) = x^2$
	$f_2(x) = (x-2)^2$
	subject to
	$-10^5 \leq x \leq 10^5$
	Minimise both
MOP 2	$f_1(\mathbf{x}) = 1 - e^{-\sum_{i=1}^{3}\left(x_i - \frac{1}{\sqrt{3}}\right)^2}$
	$f_2(\mathbf{x}) = 1 - e^{-\sum_{i+1}^{3}\left(x_i - \frac{1}{\sqrt{3}}\right)^2}$
	subject to
	$-4 \leq x_i \leq 4, \quad i = 1,2,3$
	Maximise both
MOP 3	$f_1(\mathbf{x}) = -(1 + (A-B)^2 + (C-D)^2)$, where
	$A = 0.5\sin(1) - 2\cos(1) + \sin(2) - 1.5\cos(2)$
	$B = 0.5\sin(x_1) - 2\cos(x_1) + \sin(x_2) - 1.5\cos(x_2)$
	$C = 1.5\sin(1) - \cos(1) + 2\sin(2) - 0.5\cos(2)$
	$D = 1.5\sin(x_1) - \cos(x_1) + 2\sin(x_2) - 0.5\cos(x_2)$
	$f_2(\mathbf{x}) = -((x_1+3)^2 + (x_2+1)^2)$
	subject to
	$-\pi \leq x_i \leq \pi, \quad i = 1,2$
	Minimise both
MOP 4	$f_1(\mathbf{x}) = \sum_{i=1}^{2}(-10e^{(-0.2\sqrt{x_i^2 + x_{i+1}^2})})$
	$f_2(\mathbf{x}) = \sum_{i=1}^{3}(\lvert x_i\rvert^{0}.8 + 5\sin(x_i)^3)$
	subject to
	$-5 \leq x_i \leq 5, \quad i = 1,2,3$
	Minimise both
MOP 6	$f_1(x_1) = x_1$
	$f_2(\mathbf{x}) = (1+10x_2) \times \left(1 - \left(\frac{x_1}{1+10x_2}\right)^2 - \frac{x_1}{1+10x_2}\sin(8\pi x_1)\right)$
	subject to
	$0 \leq x_i \leq 1, \quad i = 1,2$

4.3.2 The Zitzler-Deb-Thiele problems

Using the framework laid out by Deb (1999) to construct problems, Zitzler *et al.* (2000) proposed six unconstrained test problems. The Zitzler-Deb-Thiele (ZDT) suite includes problems that are concave, convex or disconnected as well as unimodal or multimodal.

ZDT 5 is a binary problem. It is therefore excluded from the continuous test suite.

Table 4.3 shows the problem definitions and decision variable domains for the selected ZDT problems. The Pareto fronts for the five problems can be found by setting $g(\mathbf{x}) = 1$.

4.3.3 The L_1ZDT problems

Building on Deb (1999) and Zitzler *et al.* (2000), Deb *et al.* (2006) demonstrate two types of relationships between variables, referred to as *linkages*, that could be used to ensure that there are relationships between variables. The first type of linkage separately introduces relationships between the variables used for each objective value. The second type of linkage introduces relationships between all decision variables.

A problem with both mechanisms is that linkages are determined randomly. Therefore problems using these linkages change all the time.

For this study, only linkages of type one will be used. This is because the true Pareto front is still predictable using these. However, for linkages of type two, both the size and the location of the Pareto front can no longer be predicted (Deb *et al.*, 2006). The fact that the location cannot reliably be predicted poses a problem when calculating the hypervolume of the front.

Linkages of type one (denoted as L_1) are introduced using two random matrices $\mathbf{P}$ and $\mathbf{Q}$ of sizes $V_1 \times V_1$ and $(V - V_1) \times (V - V_1)$ respectively, where V_1 denotes the number of decision variables used in the first objective. All members of $\mathbf{P}$ and $\mathbf{Q}$ are random values between -1 and 1. Linkages between decision variables are introduced for the first objective by altering x_1 to be x'_1 using

$$x'_1 = \mathbf{P}x_1. \tag{4.1}$$

Similarly, $\mathbf{x}_{2,\ldots,V}$ is altered to be $\mathbf{x}'_{2,\ldots,V}$ when linkages are introduced for the second objective:

Table 4.3: Problem definitions for the ZDT problems.

Name	Function definitions
ZDT 1	*Minimise both* $f_1(x_1) = x_1$ $g(\mathbf{x}) = 1 + 9\frac{\sum_{i=2}^{n} x_i}{n-1}$ $f_2(\mathbf{x}) = 1 - \sqrt{\frac{f_1}{g}}$ *subject to* $0 \leq x_i \leq 1, \quad i = 1, \ldots, 30$
ZDT 2	*Minimise both* $f_1(x_1) = x_1$ $g(\mathbf{x}) = 1 + 9\frac{\sum_{i=2}^{n} x_i}{n-1}$ $f_2(\mathbf{x}) = 1 - \sqrt{\frac{f_1}{g}}$ *subject to* $0 \leq x_i \leq 1, \quad i = 1, \ldots, 30$
ZDT 3	*Minimise both* $f_1(x_1) = x_1$ $g(\mathbf{x}) = 1 + 9\frac{\sum_{i=2}^{n} x_i}{n-1}$ $f_2(\mathbf{x}) = 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g}\right) \sin(10\pi f_1)$ *subject to* $0 \leq x_i \leq 1, \quad i = 1, \ldots, 30$
ZDT 4	*Minimise both* $f_1(x_1) = x_1$ $g(\mathbf{x}) = 1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i^2 - 10\cos(4\pi x_i)\right)$ $f_2(\mathbf{x}) = 1 - \sqrt{\frac{f_1}{g}}$ *subject to* $0 \leq x_i \leq 1, \quad i = 1, \ldots, 10$
ZDT 6	*Minimise both* $f_1(x_1) = 1 - e^{-4x_1} \sin^6(6\pi x_1)$ $g(\mathbf{x}) = 1 + 9\left(\frac{\sum_{i=2}^{n} x_i}{n-1}\right)^{0.25}$ $f_2(\mathbf{x}) = 1 - \left(\frac{f_1}{g}\right)^2$ *subject to* $0 \leq x_i \leq 1, \quad i = 1, \ldots, 10$

Table 4.4: Available characteristics of the L_1ZDT problems.

Name	Shape of front	Modality	Linkages
L_1ZDT 1	Convex	U	✓
L_1ZDT 2	Concave	U	✓
L_1ZDT 3	Disconnected	M	✓
L_1ZDT 4	Convex	M	✓
L_1ZDT 6	Concave	M	✓

$$\mathbf{x}'_{2,\ldots,V} = \mathbf{Q}\mathbf{x}_{2,\ldots,V}. \tag{4.2}$$

The available characteristics of the L_1ZDT problems are shown in Table 4.4.

Table 4.5 shows the problem definition and variable domains for the L_1ZDT problems. Note that the variable domains are applicable to the unaltered decision variables.

4.3.4 The R problems

Iorio & Li (2005) suggested a mechanism for introducing relationships between decision variables. This was done in order to compare their newly developed non-dominated sorting differential evolution (NSDE) to the non-dominated sorting genetic algorithm II (NSGA-II).

Following from this, Iorio & Li (2006) proposed four problems (the R problems) that use the mechanism.

The essential idea is that rotating decision variables away from their original axes would introduce relationships between decision variables. This is accomplished using a combination of matrix multiplication and a specially constructed rotation matrix. The construction of the rotation matrix is shown in Algorithm 9.

Similar to the L_1ZDT problems, the decision variables are adjusted:

$$\mathbf{x}' = \mathbf{R}\mathbf{x}, \tag{4.3}$$

where $\mathbf{R}$ is the rotation matrix, $\mathbf{x}$ is the original decision variable vector and $\mathbf{x}'$ is the resulting rotated decision variable vector.

Table 4.5: Problem definitions for the L_1ZDT problems.

Name	Function definitions
L_1ZDT 1	*Minimise both* $f_1(x'_1) = x'_1$ $g(\mathbf{x}') = 1 + 9\frac{\sum_{i=2}^{n} x'_i}{n-1}$ $f_2(\mathbf{x}') = 1 - \sqrt{\frac{f_1}{g}}$ *subject to* $0 \leq x_i \leq 1, \quad i = 1, \ldots, 30$
L_1ZDT 2	*Minimise both* $f_1(x'_1) = x'_1$ $g(\mathbf{x}') = 1 + 9\frac{\sum_{i=2}^{n} x'_i}{n-1}$ $f_2(\mathbf{x}') = 1 - \sqrt{\frac{f_1}{g}}$ *subject to* $0 \leq x_i \leq 1, \quad i = 1, \ldots, 30$
L_1ZDT 3	*Minimise both* $f_1(x'_1) = x'_1$ $g(\mathbf{x}') = 1 + 9\frac{\sum_{i=2}^{n} x'_i}{n-1}$ $f_2(\mathbf{x}') = 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g}\right)\sin(10\pi f_1)$ *subject to* $0 \leq x_i \leq 1, \quad i = 1, \ldots, 30$
L_1ZDT 4	*Minimise both* $f_1(x'_1) = x'_1$ $g(\mathbf{x}') = 1 + 10(n-1) + \sum_{i=2}^{n}\left(x_i^2 - 10\cos(4\pi x'_i)\right)$ $f_2(\mathbf{x}') = 1 - \sqrt{\frac{f_1}{g}}$ *subject to* $0 \leq x_i \leq 1, i = 1, \ldots, 10$
L_1ZDT 6	*Minimise both* $f_1(x'_1) = 1 - e^{-4x'_1}\sin^6(6\pi x'_1)$ $g(\mathbf{x}') = 1 + 9\left(\frac{\sum_{i=2}^{n} x'_i}{n-1}\right)^{0.25}$ $f_2(\mathbf{x}') = 1 - \left(\frac{f_1}{g}\right)^2$ *subject to* $0 \leq x_i \leq 1, \quad i = 1, \ldots, 10$

Algorithm 9 The construction of the rotation matrix **R**.

Create a matrix R of size $V \times V$ comprising of normally distributed random numbers with $\mu = 0$ and $\sigma = 1$.
For $i = 1$ to V
  Calculate the first norm of $R(i,:)$, $\|R(i,:)\|$
  Let $B = \frac{R(i,:)}{\|R(i,:)\|}$.
  $j \leftarrow 1$.
  While $j < i$
    $C \leftarrow$ the first norm of B.
    $D \leftarrow B \times$ transpose of $R(j,:)$.
    For $i = 1$ to V
      $B(k) \leftarrow \frac{B(k) - D \times R(j,k)}{C^2}$.
    end for
    $j \leftarrow j + 1$.
  end while
  $R(i,:) = \frac{B}{\|B\|}$.
end for

The adjusted decision variables are used for function evaluation as shown in Table 4.7. Similar to the L_1ZDT problems, the variable domains of the R problems are applicable to the unaltered decision variables.

The available characteristics of the R problems are shown in Table 4.6.

Table 4.6: Available characteristics of the R problems.

Name	Shape of front	Modality	Rotated
R 1	Convex	M, D	✓
R 2	Disconnected	M	✓
R 3	Concave	M	✓
R 4	Convex	M, D	✓

4.3.5 The walking fish group problems

As mentioned above, the walking fish group (WFG) toolkit is introduced in Huband *et al.* (2005). The WFG test suite comprises nine box-constrained scalable (in both the number of variables and the number of objectives) problems built using the WFG toolkit. The toolkit and the suggested test problems will now be discussed.

Table 4.7: Problem definitions of the R problems.

Name	Function definitions		
	Minimise both		
R 1	$f_1(x'_1)$	$=$	x'_1
	$g(\mathbf{x}')$	$=$	$1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i'^2 - 10\cos(4\pi x'_i)\right)$
	$h\left(f_1(x'_1), g(\mathbf{x}')\right)$	$=$	$e^{\frac{-f_1(x'_i)}{g(\mathbf{x}')}}$
	$f_2(\mathbf{x}')$	$=$	$g(\mathbf{x}')h\left(f_1(x'_1), g(\mathbf{x}')\right)$
	subject to		
			$-0.3 \leq x_i \leq 0.3, \quad i = 1, \ldots, 10,$ *and* $\lvert f_1 \rvert \leq 0.3$
	Minimise both		
R 2	$f_1(x'_1)$	$=$	x'_1
	$g(\mathbf{x}')$	$=$	$1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i'^2 - 10\cos(\pi x'_i)\right)$
	$h\left(f_1(x'_1), g(\mathbf{x}')\right)$	$=$	$1 + e^{\frac{-f_1(x'_i)}{g(\mathbf{x}')}} + \left(\frac{f_1(x'_1)+1}{g(\mathbf{x}')}\right) \sin(5\pi f_1(x'_1))$
	$f_2(\mathbf{x}')$	$=$	$g(\mathbf{x}')h\left(f_1(x'_1), g(\mathbf{x}')\right)$
	subject to		
			$-1 \leq x_i \leq 1, \quad i = 1, \ldots, 10,$ *and* $\lvert f_1 \rvert \leq 1$
	Minimise both		
R 3	$f_1(x'_1)$	$=$	$1 - \frac{e^{2x'_1} \sin^6(6\pi x'_1)}{9}$
	$g(\mathbf{x}')$	$=$	$1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i'^2 - 10\cos(\pi x'_i)\right)$
	$h\left(f_1(x'_1), g(\mathbf{x}')\right)$	$=$	$1 - \left(\frac{f_1(x'_i)}{g(\mathbf{x}')}\right)^2$
	$f_2(\mathbf{x}')$	$=$	$g(\mathbf{x}')h\left(f_1(x'_1), g(\mathbf{x}')\right)$
	subject to		
			$-1 \leq x_i \leq 1, \quad i = 1, \ldots, 10,$ *and* $0.3 \leq f_1 \leq 1$
	Minimise both		
R 4	$f_1(x'_1)$	$=$	x'_1
	$g(\mathbf{x}')$	$=$	$1 + 0.015578(n-1) + \sum_{i=2}^{n} \left(x_i'^2 - 0.25x'_i \sin(32\sqrt{\lvert x'_i \rvert})\right)$
	$h\left(f_1(x'_1), g(\mathbf{x}')\right)$	$=$	$e^{\frac{-f_1(x'_i)}{g(\mathbf{x}')}}$
	$f_2(\mathbf{x}')$	$=$	$g(\mathbf{x}')h\left(f_1(x'_1), g(\mathbf{x}')\right)$
	subject to		
			$-1 \leq x_i \leq 1, \quad i = 1, \ldots, 10,$ *and* $\lvert f_1 \rvert \leq 1$

A set of decision variables $\mathbf{x}$ (with the size of $\mathbf{x}$ being scalable), is divided into two groups: distance-related variables and position-related variables, where the number of distance-related variables is denoted by V_D and the number of position-related variables is denoted by V_P.

An objective function is made up of *transformation* and *shape functions*, as well as *degeneracy* and *scaling parameters*. Before any of these are used, all decision variables are scaled to fall between 0 and 1. The scaled decision variables are adjusted using a series of transformation functions. Before incorporating a shape function, the Pareto front can be made degenerate by appropriately adjusting the degeneracy parameter ϑ_i for each objective function i. The transformed position-related variables are used as input to a chosen shape function. Finally, an objective function is constructed using the scaled outputs of the reduction transformation and the shape function. Each objective function i is associated with a scaling parameter ψ_i which determines the range of the objective function. For example, if the scaling parameter for the first objective $\psi_1 = 2$ and the scaling parameter for the second objective $\psi_2 = 4$, and values for the first objective are plotted along the x-axis and values for the second objective are plotted along the y-axis, the Pareto front will intersect with the x-axis at $x = 2$ and with the y-axis at $y = 4$.

There are eight basic transformation functions, three of which are bias functions and three of which shift the location of the optimum. The last two are reduction functions. The researcher can select any combination of these functions and the order in which they should be carried out, keeping in mind that the order affects the resulting objective function. Before these transformation functions are discussed, please note that the original decision variables, as adjusted by the metaheuristics, are denoted by x_i throughout this chapter and this document. When referencing variables more generally, a temporary variable y_i, or temporary adjusted variables y_i' or y_i'' are preferred. In reality, all these temporary variables (y_i, y_i' and y_i'') are functions of $\mathbf{x}$. However, the exact functions vary based on the transformation and shape functions selected. As a result, it is simpler to refer to a general variable y_i (with $y_i(\mathbf{x})$) than to the specific – and unknown, varying – function of x_i. In an example at the end of this section, the relationships between x_i, y_i, y_i' and y_i'' will be illustrated clearly.

1. A **polynomial bias** (determined by τ_1, with $\tau_1 > 0$ and $\tau \neq 1$), is added to a temporary variable y_i to create a temporary adjusted variable y'_i

$$y'_i = y_i^{\tau_1}. \tag{4.4}$$

2. A **flat region bias** is introduced using three parameters: τ_2, υ_2 and ω_2, with $\tau_2, \upsilon_2, \omega_2 \in [0,1]$, $\upsilon_2 < \omega_2$, $\tau_2 = 0$ and $\omega_2 \neq 1$ if $\upsilon_2 = 0$, and $\tau_2 = 1$ and $\upsilon_2 \neq 0$ if $\omega_2 = 1$. The temporary variable y_i is adjusted:

$$y'_i = \tau_2 + \min\left(0, \lfloor y_i - \upsilon_2 \rfloor\right) \frac{\tau_2(\upsilon_2 - y_i)}{\upsilon_2} - \min\left(0, \lfloor \omega_2 - y_i \rfloor\right) \frac{(1-\tau_2)(y_i - \omega_2)}{y_i - \omega_2}. \tag{4.5}$$

   Introducing a flat region bias results in all values of y_i between υ_2 and ω_2 being assigned a value equal to τ_2.

3. A **decision variable dependent bias** can also be introduced to a temporary variable. This is done using three parameters (τ_3, υ_3 and ω_3, with $\tau_3 \in (0,1)$, and $0 < \upsilon_3 < \omega_3$) and the result of a weighted sum reduction ϖ_3 of some of the decision variables:

$$y'_1 = y_1^{\upsilon_3 + (\omega_3 - \upsilon_3)(\tau_3 - (1-\varpi_3)|\lfloor 0.5 - \varpi_3 \rfloor|)}. \tag{4.6}$$

4. The optimum of each objective function can be **shifted linearly** using a single parameter τ_4:

$$y'_i = \frac{y_i - \tau_4}{|\lfloor \tau_4 - y_1 \rfloor + \tau_4|}. \tag{4.7}$$

5. The optimum of each objective function can be shifted to be **deceptive.** This is done using three parameters: τ_5, υ_5 and ω_5, with $\tau_5 \in (0,1)$, $0 < \upsilon_5 \ll 1$, $0 < \omega_5 \ll 1$, $\tau_5 \neq \upsilon_5$, and $\tau_5 + \upsilon_5 < 0$. The position of the global minimum is controlled by τ_5 (for $y_i = \tau_5$, the solution would be optimal), ω_5 controls the position of the deceptive minimum, and υ_5 controls the depth of the valley

between the two. In order to shift the optimum, the temporary variable y_i is adjusted:

$$y_i' = 1 + (|y_i - \tau_5| - \upsilon_5)\left(\frac{\lfloor y_i - \tau_5 + \upsilon_5 \rfloor \left(1 - \omega_5 + \frac{\tau_5 - \upsilon_5}{\upsilon_5}\right)}{\tau_5 - \upsilon_5} + \frac{\lfloor \tau_5 + \upsilon_5 - y_i \rfloor \left(1 - \omega_5 + \frac{1 - \tau_5 - \upsilon_5}{\upsilon_5}\right)}{1 - \tau_5 - \upsilon_5} + \frac{1}{\upsilon_5}\right). \quad (4.8)$$

6. Each objective function can be changed from being unimodal to being **multimodal**. Similar to the previous shift function, three parameters are used: τ_6, υ_6 and ω_6, with $\tau_6 \in \mathbb{N}$, $\upsilon_6 \geq 0$, $(4\tau_6 + 2)\pi \geq 4\upsilon_6$, and $C \in (0, 1)$. The number of minima is controlled by τ_6, while υ_6 controls the size of the 'hills' and 'valleys'. The position of the global minimum is controlled by ω_6 (y_i is optimal when $y_i = \omega_6$). A unimodal objective function is transformed to be multimodal:

$$y_i' = \frac{1 + \cos\left((4\tau_6 + 2)\pi\left(0.5 - \frac{|y_i - \omega_6|}{2(\lfloor \omega_6 - y_i \rfloor + \omega_6)}\right)\right) + 4\upsilon_6\left(\frac{|y_i - \omega_6|}{2(\lfloor \omega_6 - y_i \rfloor + \omega_6)}\right)^2}{\upsilon_6 + 2}. \quad (4.9)$$

7. Multiple decision variable values can be reduced to a single value using a **weighted sum reduction** function:

$$y_i' = \frac{\sum_{j=1}^{V_w} w_j y_j}{\sum_{j=1}^{V_w} w_j}, \quad (4.10)$$

where w_j is the weight assigned to the j^{th} variable, and V_w is the number of variables to be reduced using the weighted sum reduction function. The weighted reduction function is often used as a final transformation to reduce the number of variables from V to M.

8. Multiple decision variable values can be reduced to a single value using a **non-separable reduction** function:

$$y_i' = \frac{\tau_8 \sum_{j=1}^{V_n}\left(y_j + \sum_{k=0}^{\tau_8 - 2} |y_j - y_{(1+(j+k) \bmod V_n)}|\right)}{V_n \lceil \frac{\tau_8}{2} \rceil (1 + 2\tau_8 - 2\lceil \frac{\tau_8}{2} \rceil)}. \quad (4.11)$$

The parameter τ_8 determines the degree of non-separability and V_n is the number variables to be reduced using a non-separable reduction function. The non-separable reduction function is often used as a final transformation to reduce the number of variables from V to M.

The degeneracy parameters are incorporated after the transformation functions and before the shape functions. If $\mathbf{y}'$ is the set of completely transformed variables, then the set of variables with degeneracy parameters incorporated is denoted as $\mathbf{y}^D$ and is obtained as follows:

$$\begin{aligned}\mathbf{y}^D &= [y_1^D, \ldots, y_M^D] \\ &= [\max(y'_M, \vartheta_1)(y'_1 - 0.5) + 0.5, \ldots, \max(y'_M, \vartheta_{M-1})(y'_{M-1} - 0.5) + 0.5, y'_M].\end{aligned} \tag{4.12}$$

There are five basic shape functions, influencing whether an objective function has a linear, convex, concave, mixed or disconnected shape. Each objective of a multi-objective problem can have a different shape. These shape functions are laid out in Table 4.8. The shape parameter κ determines whether an objective function is convex ($\kappa > 1$), concave ($\kappa < 1$) or linear ($\kappa = 1$), and the parameter C_C determines the number of convex or concave sections. The parameter C_D determines the number of disconnected regions of a problem, and the parameter θ determines the locations of the disconnected regions.

Once the shape functions have been applied, the objective functions are finished by incorporating the scaling parameters. The basic form of the objective functions is

$$f_{i=1:M}(\mathbf{x}) = \phi y'_M + \psi_i h_i, \tag{4.13}$$

where ϕ is a distance-scaling constant, ψ_i is a shape-scaling constant and h_i is the i^{th} shape function.

The distance-scaling constant is denoted by ϕ and is equal to one for all the WFG problems. Because of this, it will not be taken into account any further.

The transformation and shape functions used for the construction of each WFG problem is summarised in Table 4.9. The parameter values for each problem are also summarised there. The number of variables and the number of objectives are both scalable.

Table 4.8: Shape functions used when constructing the WFG problems.

Shape functions		
$linear_1(y_1, \ldots, y_{M-1})$	$=$	$\prod_{i=1}^{M-1} y_i$
$linear_{j=2:M-1}(y_1, \ldots, y_{M-1})$	$=$	$\left(\prod_{i=1}^{M-1} y_i\right)(1 - y_{(M-j+1)})$
$linear_M(y_1, \ldots, y_{M-1})$	$=$	$1 - y_1$
$convex_1(y_1, \ldots, y_{M-1})$	$=$	$\prod_{i=1}^{M-1} (1 - \cos(0.5\pi y_i))$
$convex_{j=2:M-1}(y_1, \ldots, y_{M-1})$	$=$	$\left(\prod_{i=1}^{M-1} (1 - \cos(0.5\pi y_i))\right)(1 - \sin(0.5\pi y_{(M-j+1)}))$
$convex_M(y_1, \ldots, y_{M-1})$	$=$	$1 - \sin(0.5\pi y_1)$
$concave_1(y_1, \ldots, y_{M-1})$	$=$	$\prod_{i=1}^{M-1} (\sin(0.5\pi y_i))$
$concave_{j=2:M-1}(y_1, \ldots, y_{M-1})$	$=$	$\left(\prod_{i=1}^{M-1} (\sin(0.5\pi y_i))\right)\cos(0.5\pi y_{(M-j+1)})$
$concave_M(y_1, \ldots, y_{M-1})$	$=$	$\cos(0.5\pi y_1)$
$mixed_M(y_1, \ldots, y_{M-1})$	$=$	$\left(1 - y_1 - \frac{\cos(2C_C\pi y_1 + 0.5\pi)}{2C_C\pi}\right)^{\kappa}$
$disc_M(y_1, \ldots, y_{M-1})$	$=$	$1 - y_1^{\kappa}\cos^2(C_D\pi y_1^{\theta})$

Table 4.9: Summary of the construction of the nine WFG problems.

For all problems	Scaling parameter $\psi_i = 2, \quad i = 1, \ldots, M.$ $y_i \leftarrow \frac{x_i}{\text{Upper bound of } x_i}, \quad i = 1, \ldots, V.$
	WFG 1
	Degeneracy parameter $\vartheta_i = 1, \quad i = 1, \ldots, M.$
Transformation functions	1. Distance-related variables $y_{V_P+1:V}$ shifted linearly using (4.7) with $\tau_4 = 0.35$. 2. Flat region bias introduced in distance-related variables $y_{V_P+1:V}$ using (4.5) with $\tau_2 = 0.8$, $\upsilon_2 = 0.75$ and $\omega_2 = 0.85$. 3. Polynomial bias introduced in all variables using (4.4) with $\tau_1 = 0.02$. 4. Weighted sum reduction (as in (4.10)) used to reduce the number of variables from V to M. The first $1 : M-1$ new variables $y'_{i=1:M-1}$ are created by reducing vectors made up of $[y_{(i-1)V_P/(M-1)+1}, \ldots, y_{iV_P/(M-1)}]$ and a weight vector $\mathbf{w}$ equal to $[2((i-1)V_P/(M-1)+1), \ldots, 2iV_P/(M-1)]$. The M^{th} new variable y'_M results from reducing the vector made up of $[y_{V_P+1}, \ldots, y_V]$ using a weight vector $\mathbf{w}$ equal to $[2(V_P+1), \ldots, 2V]$.
Shape functions	Objectives 1 to $M-1$: Convex Objective M: Mixed (with $\kappa = 1$ and $C_C = 5$)
	WFG 2
	Degeneracy parameter $\vartheta_i = 1, \quad i = 1, \ldots, M.$
Transformation functions	1. Distance-related variables $y_{V_P+1:V}$ shifted linearly using (4.7) with $\tau_4 = 0.35$. 2. Non-separable reduction of the distance-related variables $y_{V_P+1:V}$ using (4.11). The number of distance-related variables is reduced from V_D to $0.5V_D$. The new variables $y_{i=V_P+1:V_P+0.5V_D}$ are created by reducing vectors comprising two members $[y_{V_P+2(i-V_P)-1}, y_{V_P+2(i-V_P)}]$ with $\tau_8 = 2$. 3. Weighted sum reduction (as in (4.10)) used to reduce the number of variables from $V_P + 0.5V_D$ to M. The first $1 : M-1$ new variables $y'_{i=1:M-1}$ are created by reducing vectors made up of $[y_{(i-1)V_P/(M-1)+1}, \ldots, y_{iV_P/(M-1)}]$ and a weight vector $\mathbf{w}$ equal to $1, \ldots, 1]$. The M^{th} new variable y'_M results from reducing the vector made up of $[y_{V_P+1}, \ldots, y_{V_P+0.5V_D}]$ using a weight vector $\mathbf{w}$ equal to $[1, \ldots, 1]$.

Continued on next page

Table 4.9 – continued from previous page

Shape functions	Objectives 1 to $M-1$: Convex Objective M: Disconnected (with $\kappa = \theta = 1$ and $C_D = 5$)
WFG 3	
	Degeneracy parameter $\vartheta_1 = 1$, and $\vartheta_i = 0, \quad i = 1, \ldots, M$.
Transformation functions	1. Distance-related variables $y_{V_P+1:V}$ shifted linearly using (4.7) with $\tau_4 = 0.35$. 2. Non-separable reduction of the distance-related variables $y_{V_P+1:V}$ using (4.11). The number of distance-related variables is reduced from V_D to $0.5V_D$. The new variables $y_{i=V_P+1:V_P+0.5V_D}$ are created by reducing vectors comprising two members $[y_{V_P+2(i-V_P)-1}, y_{V_P+2(i-V_P)}]$ with $\tau_8 = 2$. 3. Weighted sum reduction (as in (4.10)) used to reduce the number of variables from $V_P + 0.5V_D$ to M. The first $1 : M-1$ new variables $y'_{i=1:M-1}$ are created by reducing vectors made up of $[y_{(i-1)V_P/(M-1)+1}, \ldots, y_{iV_P/(M-1)}]$ and a weight vector $\mathbf{w}$ equal to $1, \ldots, 1]$. The M^{th} new variable y'_M results from reducing the vector made up of $[y_{V_P+1}, \ldots, y_{V_P+0.5V_D}]$ using a weight vector $\mathbf{w}$ equal to $[1, \ldots, 1]$.
Shape functions	Objectives 1 to M: Linear
WFG 4	
	Degeneracy parameter $\vartheta_i = 1, \quad i = 1, \ldots, M$.
Transformation functions	1. Change unimodal objective function to be multimodal using (4.9) with $\tau_6 = 30$, $\upsilon_6 = 10$, and $\omega_6 = 0.35$. 2. Weighted sum reduction (as in (4.10)) used to reduce the number of variables from V to M. The first $1 : M-1$ new variables $y'_{i=1:M-1}$ are created by reducing vectors made up of $[y_{(i-1)V_P/(M-1)+1}, \ldots, y_{iV_P/(M-1)}]$ and a weight vector $\mathbf{w}$ equal to $1, \ldots, 1]$. The M^{th} new variable y'_M results from reducing the vector made up of $[y_{V_P+1}, \ldots, y_V]$ using a weight vector $\mathbf{w}$ equal to $[1, \ldots, 1]$.
Shape functions	Objectives 1 to $M-1$: Concave
WFG 5	
	Degeneracy parameter $\vartheta_i = 1, \quad i = 1, \ldots, M$.
Transformation functions	1. Change the objective functions to be deceptive using (4.8) with $\tau_5 = 0.35$, $\upsilon_5 = 0.001$, and $\omega_5 = 0.05$.

Continued on next page

Table 4.9 – continued from previous page

	2. Weighted sum reduction (as in (4.10)) used to reduce the number of variables from V to M. The first $1 : M-1$ new variables $y'_{i=1:M-1}$ are created by reducing vectors made up of $[y_{(i-1)V_P/(M-1)+1}, \ldots, y_{iV_P/(M-1)}]$ and a weight vector $\mathbf{w}$ equal to $1, \ldots, 1]$. The M^{th} new variable y'_M results from reducing the vector made up of $[y_{V_P+1}, \ldots, y_V]$ using a weight vector $\mathbf{w}$ equal to $[1, \ldots, 1]$.
Shape functions	Objectives 1 to $M-1$: Concave
	WFG 6
	Degeneracy parameter $\vartheta_i = 1, \quad i = 1, \ldots, M$.
Transformation functions	1. Distance-related variables $y_{V_P+1:V}$ shifted linearly using (4.7) with $\tau_4 = 0.35$. 2. Non-separable reduction (as in (4.11)) used to reduce the number of variables from V to M. The first $1 : M-1$ new variables $y'_{i=1:M-1}$ are created by reducing vectors made up of $[y_{(i-1)V_P/(M-1)+1}, \ldots, y_{iV_P/(M-1)}]$ and $\tau_8 = \frac{V_P}{M-1}$. The M^{th} new variable y'_M results from reducing the vector made up of $[y_{V_P+1}, \ldots, y_V]$ $\tau_8 = V_D$.
Shape functions	Objectives 1 to $M-1$: Concave
	WFG 7
	Degeneracy parameter $\vartheta_i = 1, \quad i = 1, \ldots, M$.
Transformation functions	1. Decision variable dependent bias introduced in position-related variables $y_{1:V_P}$ using (4.6) with $\tau_3 = \frac{0.98}{49.98}$, $\upsilon_3 = 0.02$, and $\omega_3 = 50$. For each $y'_{i=1:V_P}$, the value of ϖ_3 is obtained by applying a weighted sum reduction to a vector $[y_{i+1}, \ldots, y_V]$ using weights $\mathbf{w}$ equal to $[1, \ldots, 1]$. 2. Distance-related variables $y_{V_P+1:V}$ shifted linearly using (4.7) with $\tau_4 = 0.35$. 3. Weighted sum reduction (as in (4.10)) used to reduce the number of variables from V to M. The first $1 : M-1$ new variables $y'_{i=1:M-1}$ are created by reducing vectors made up of $[y_{(i-1)V_P/(M-1)+1}, \ldots, y_{iV_P/(M-1)}]$ and a weight vector $\mathbf{w}$ equal to $1, \ldots, 1]$. The M^{th} new variable y'_M results from reducing the vector made up of $[y_{V_P+1}, \ldots, y_V]$ using a weight vector $\mathbf{w}$ equal to $[1, \ldots, 1]$.
Shape functions	Objectives 1 to $M-1$: Concave

Continued on next page

Table 4.9 – continued from previous page

WFG 8	
	Degeneracy parameter $\vartheta_i = 1, \quad i = 1, \ldots, M$.
Transformation functions	1. Decision variable dependent bias introduced in distance-related variables $y_{V_P+1:V}$ using (4.6) with $\tau_3 = \frac{0.98}{49.98}$, $\upsilon_3 = 0.02$, and $\omega_3 = 50$. For each $y'_{i=V_P+1:V}$, the value of ϖ_3 is obtained by applying a weighted sum reduction to a vector $[y_1, \ldots, y_{i-1}]$ using weights $\mathbf{w}$ equal to $[1, \ldots, 1]$. 2. Distance-related variables $y_{V_P+1:V}$ shifted linearly using (4.7) with $\tau_4 = 0.35$. 3. Weighted sum reduction (as in (4.10)) used to reduce the number of variables from V to M. The first $1 : M-1$ new variables $y'_{i=1:M-1}$ are created by reducing vectors made up of $[y_{(i-1)V_P/(M-1)+1}, \ldots, y_{iV_P/(M-1)}]$ and a weight vector $\mathbf{w}$ equal to $1, \ldots, 1]$. The M^{th} new variable y'_M results from reducing the vector made up of $[y_{V_P+1}, \ldots, y_V]$ using a weight vector $\mathbf{w}$ equal to $[1, \ldots, 1]$.
Shape functions	Objectives 1 to $M-1$: Concave
WFG 9	
	Degeneracy parameter $\vartheta_i = 1, \quad i = 1, \ldots, M$.
Transformation functions	1. Decision variable dependent bias introduced in position-related variables $y_{1:V_P}$ using (4.6) with $\tau_3 = \frac{0.98}{49.98}$, $\upsilon_3 = 0.02$, and $\omega_3 = 50$. For each $y'_{i=1:V_P}$, the value of ϖ_3 is obtained by applying a weighted sum reduction to a vector $[y_{i+1}, \ldots, y_V]$ using weights $\mathbf{w}$ equal to $[1, \ldots, 1]$. 2. The optimum of each objective function is shifted to be deceptive using the position-related variables $y_{1:V_P}$ and (4.8), with $\tau_5 = 0.35$, $\upsilon_5 = 0.001$, and $\omega_5 = 0.05$. All objective function values are also adjusted to be multimodal using the distance-related variables $y_{V_P+1:V}$ and (4.9) with $\tau_6 = 30$, $\upsilon_6 = 95$, and $\omega_6 = 0.35$. 2. Non-separable reduction (as in (4.11)) used to reduce the number of variables from V to M. The first $1 : M-1$ new variables $y'_{i=1:M-1}$ are created by reducing vectors made up of $[y_{(i-1)V_P/(M-1)+1}, \ldots, y_{iV_P/(M-1)}]$ and $\tau_8 = \frac{V_P}{M-1}$. The M^{th} new variable y'_M results from reducing the vector made up of $[y_{V_P+1}, \ldots, y_V]$ $\tau_8 = V_D$.
Shape functions	Objectives 1 to $M-1$: Concave

In order to illustrate how a WFG problem is constructed, an example will now

be worked through. The example is the construction of WFG 1, with two objective functions, two position-related variables x_1 and x_2, and two distance-related variables x_3 and x_4. This is the smallest possible version of WFG 1. We will also make use of four temporary placeholders y_1 to y_4.

First, all decision variables x_i are scaled to fall between 0 and 1. Temporary variables y_i are used to store the scaled decision variables:

$$y_i = \frac{x_i}{\mathit{Upper\ bound\ of}\ x_i} \tag{4.14}$$

$$y_1 = \frac{x_1}{2}, \quad y_2 = \frac{x_2}{4}, \quad y_3 = \frac{x_3}{6}, \quad y_4 = \frac{x_4}{8}. \tag{4.15}$$

For the first transformation, the position-related variables remain unchanged:

$$y'_1 = y_1 \tag{4.16}$$

$$y'_2 = y_2, \tag{4.17}$$

while the two scaled distance-related variables are shifted linearly. For convenience, (4.7) is repeated here:

$$y'_i = \frac{y_i - \tau_4}{|\lfloor \tau_4 - y_1 \rfloor + \tau_4|}. \tag{4.18}$$

Substituting y_i in (4.18) with y_3 and $\tau_4 = 0.35$, yields:

$$y'_3 = \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|}. \tag{4.19}$$

Similarly, substituting y_i in (4.18) with y_3 and $\tau_4 = 0.35$, yields:

$$y'_4 = \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|}. \tag{4.20}$$

For the second transformation, a flat region bias is introduced in the distance-related variables, while the position-related variables remain unchanged. Recall, from (4.5), that a flat region bias is introduced as follows:

$$y'_i = \tau_2 + \min\left(0, \lfloor y_i - \upsilon_2 \rfloor\right) \frac{\tau_2(\upsilon_2 - y_i)}{\upsilon_2} - \min\left(0, \lfloor \omega_2 - y_i \rfloor\right) \frac{(1-\tau_2)(y_i - \omega_2)}{y_i - \omega_2}. \tag{4.21}$$

Substituting $\tau_2 = 0.8$, $\upsilon_2 = 0.75$ and $\omega_2 = 0.85$, and substituting y_i in (4.21) with y_3' from (4.19), yields:

$$y_3' = 0.8 + \min\left(0, \left\lfloor \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|} - 0.75 \right\rfloor\right) \frac{0.8\left(0.75 - \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|}\right)}{0.75} - \min\left(0, \left\lfloor 0.85 - \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|} \right\rfloor\right) \frac{0.2\left(\frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|} - 0.85\right)}{0.15} \tag{4.22}$$

Similarly, substituting $\tau_2 = 0.8$, $\upsilon_2 = 0.75$ and $\omega_2 = 0.85$, and substituting y_i in (4.21) with y_4' from (4.20), yields:

$$y_4' = 0.8 + \min\left(0, \left\lfloor \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|} - 0.75 \right\rfloor\right) \frac{0.8\left(0.75 - \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|}\right)}{0.75} - \min\left(0, \left\lfloor 0.85 - \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|} \right\rfloor\right) \frac{0.2\left(\frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|} - 0.85\right)}{0.15}. \tag{4.23}$$

For the third transformation, a polynomial bias is introduced in all four decision variables. Recall, from (4.4), that a polynomial bias is introduced as follows:

$$y_i' = y_i^{\tau_1}. \tag{4.24}$$

For the position-related variables, y_i in (4.24) is simply substituted with y_1 and y_2 respectively. From Table 4.9, $\tau_1 = 0.02$. This yields:

$$y_1' = y_1^{0.02} \tag{4.25}$$

$$y_2' = x_2^{0.02}. \tag{4.26}$$

For the distance-related variables, y_i in (4.24) is substituted with y_3' from (4.22) to yield:

$$y_3' = \left(0.8 + \min\left(0, \left\lfloor \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|} - 0.75 \right\rfloor\right) \frac{0.8\left(0.75 - \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|}\right)}{0.75} - \min\left(0, \left\lfloor 0.85 - \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|} \right\rfloor\right) \frac{0.2\left(\frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|} - 0.85\right)}{0.15}\right)^{0.02}, \tag{4.27}$$

and y_i in (4.24) is substituted with y_4' from (4.22) to yield:

$$y_4 = \left(0.8 + \min\left(0, \left\lfloor \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|} - 0.75 \right\rfloor\right) \frac{0.8\left(0.75 - \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|}\right)}{0.75} \right.$$
$$\left. - \min\left(0, \left\lfloor 0.85 - \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|} \right\rfloor\right) \frac{0.2\left(\frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|} - 0.85\right)}{0.15} \right)^{0.02}. \quad (4.28)$$

For the fourth transformation, a weighted sum reduction is used to reduce the number of variables from $V = 4$ to $M = 2$. The two resulting transformed variables will be denoted as y_1'' and y_2''. From (4.10), a weighted sum reduction involves:

$$y_i' = \frac{\sum_{j=1}^{V_w} w_j y_j}{\sum_{j=1}^{V_w} w_j}. \quad (4.29)$$

From (4.29), the first reduced variable y_1'' is created by reducing the vector $[y_1', y_2']$ using a weight vector $\mathbf{w}$ equal to $[2, 4]$:

$$y_1'' = \frac{2y_1 + 4y_2}{6}. \quad (4.30)$$

Substituting y_1 with y_1' from (4.25), and y_2 with y_2' from (4.26) in (4.30), yields:

$$y_1'' = \frac{2y_1^{0.02} + 4y_2^{0.02}}{6}. \quad (4.31)$$

This can be simplified to be:

$$y_1'' = \frac{y_1^{0.02} + 2y_2^{0.02}}{3}. \quad (4.32)$$

The second reduced variable y_2'' results from substituting the vector $[y_3, y_4]$ and the weight vector $\mathbf{w}$ equal to $[6, 8]$ into (4.29):

$$y_2'' = \frac{6y_3 + 8y_4}{14}, \quad (4.33)$$

which can be simplified to be

$$y_2'' = \frac{3}{7}y_3 + \frac{4}{7}y_4. \quad (4.34)$$

Subsequently substituting y_3 with y_3' from (4.27), and y_4 with y_4' from (4.28) in (4.34), yields:

$$y_2'' = \frac{3}{7}\left(0.8 + \min\left(0, \left\lfloor \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|} - 0.75 \right\rfloor\right) \frac{0.8\left(0.75 - \frac{|y_3-0.35|}{|\lfloor 0.35-y_3 \rfloor+0.35|}\right)}{0.75} \right.$$
$$\left. - \min\left(0, \left\lfloor 0.85 - \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|} \right\rfloor\right) \frac{0.2\left(\frac{|y_3-0.35|}{|\lfloor 0.35-y_3 \rfloor+0.35|} - 0.85\right)}{0.15} \right)^{0.02}$$
$$+ \frac{4}{7}\left(0.8 + \min\left(0, \left\lfloor \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|} - 0.75 \right\rfloor\right) \frac{0.8\left(0.75 - \frac{|y_4-0.35|}{|\lfloor 0.35-y_4 \rfloor+0.35|}\right)}{0.75} \right.$$
$$\left. - \min\left(0, \left\lfloor 0.85 - \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|} \right\rfloor\right) \frac{0.2\left(\frac{|y_4-0.35|}{|\lfloor 0.35-y_4 \rfloor+0.35|} - 0.85\right)}{0.15} \right)^{0.02}. \quad (4.35)$$

The Pareto optimal front for WFG 1 is not degenerate, and incorporating the degeneracy parameters has no effect. Next, the shape function for each objective function is applied to the completely transformed decision variables. Note that the shape functions are functions of only y_1'' in this case. A convex shape function is used for objective function 1. From Table 4.8, the general form of the convex shape function for a first objective function is:

$$h_1 = convex_1(y_1', \ldots, y_{M-1}') = \prod_{i=1}^{M-1} \left(1 - \cos(0.5\pi y_i')\right), \quad (4.36)$$

where h_1 denotes the shape function of the first objective function.

More specifically, the equation below shows what h_1 (the shape function used for the first objective) looks like after y_1'' from (4.32) has been substituted into it:

$$h_1 = 1 - \cos\left(\frac{\pi(t_1^{0.02} + 2t_2^{0.02})}{6}\right). \quad (4.37)$$

A mixed shape function is used for objective function 2. From Table 4.8, the general form of the mixed shape function is:

$$h_2 = mixed_M(y_1', \ldots, y_{M-1}') = \left(1 - y_1' - \frac{\cos(2C_C\pi y_1' + 0.5\pi)}{2C_C\pi}\right)^{\kappa}, \quad (4.38)$$

where h_2 denotes the shape function of the second objective function.

Substituting y_1'' from (4.32), $\kappa = 1$ and $C_C = 5$ into h_2 above, yields:

$$h_2 = 1 - \frac{y_1^{0.02} + 2y_2^{0.02}}{3} - \frac{\cos\left(10\pi\frac{y_1^{0.02}+2y_2^{0.02}}{3} + 0.5\pi\right)}{10\pi}. \quad (4.39)$$

Recall from (4.40) that the basic form of the objective functions is

$$f_{i=1:M}(\mathbf{x}) = \phi y'_M + \psi_i h_i, \tag{4.40}$$

where $\phi = 1$. From Table 4.9, the shape-scaling parameter for the first objective function is $\psi_1 = 2$, and for the second objective function $\psi_2 = 4$.

Substituting h_1, y''_2, and $\psi_1 = 2$ into (4.40) yields objective function 1:

$$\begin{aligned} f_1 = \frac{3}{7}\Bigg(0.8 + \min\left(0, \left\lfloor \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|} - 0.75 \right\rfloor\right) \frac{0.8\left(0.75 - \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|}\right)}{0.75} \\ - \min\left(0, \left\lfloor 0.85 - \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|} \right\rfloor\right) \frac{0.2\left(\frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|} - 0.85\right)}{0.15}\Bigg)^{0.02} \\ + \frac{4}{7}\Bigg(0.8 + \min\left(0, \left\lfloor \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|} - 0.75 \right\rfloor\right) \frac{0.8\left(0.75 - \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|}\right)}{0.75} \\ - \min\left(0, \left\lfloor 0.85 - \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|} \right\rfloor\right) \frac{0.2\left(\frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|} - 0.85\right)}{0.15}\Bigg)^{0.02} \\ + 2\left(1 - \cos\left(\frac{\pi(y_1^{0.02} + 2y_2^{0.02})}{6}\right)\right). \end{aligned} \tag{4.41}$$

Substituting h_2, y''_2, and $\psi_2 = 4$ into (4.40) yields objective function 2:

$$f_2 = \frac{3}{7}\left(0.8 + \min\left(0, \left\lfloor \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|} - 0.75 \right\rfloor\right) \frac{0.8\left(0.75 - \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|}\right)}{0.75} \right.$$
$$\left. - \min\left(0, \left\lfloor 0.85 - \frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|} \right\rfloor\right) \frac{0.2\left(\frac{|y_3 - 0.35|}{|\lfloor 0.35 - y_3 \rfloor + 0.35|} - 0.85\right)}{0.15} \right)^{0.02}$$
$$+ \frac{4}{7}\left(0.8 + \min\left(0, \left\lfloor \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|} - 0.75 \right\rfloor\right) \frac{0.8\left(0.75 - \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|}\right)}{0.75} \right.$$
$$\left. - \min\left(0, \left\lfloor 0.85 - \frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|} \right\rfloor\right) \frac{0.2\left(\frac{|y_4 - 0.35|}{|\lfloor 0.35 - y_4 \rfloor + 0.35|} - 0.85\right)}{0.15} \right)^{0.02}$$
$$+ 4\left(1 - \frac{y_1^{0.02} + 2y_2^{0.02}}{3} - \frac{\cos\left(10\pi \frac{y_1^{0.02} + 2y_2^{0.02}}{3} + \frac{\pi}{2}\right)}{10\pi}\right) \quad (4.42)$$

The two objective functions can be written in terms of the original decision variables x_1 to x_4, by substituting (4.15) into (4.41) and (4.44) respectively. This yields objective function 1 in terms of $\mathbf{x}$:

$$f_1(\mathbf{x}) = \frac{3}{7}\left(0.8 + \min\left(0, \left\lfloor \frac{|\frac{x_3}{6} - 0.35|}{|\lfloor 0.35 - \frac{x_3}{6} \rfloor + 0.35|} - 0.75 \right\rfloor\right) \frac{0.8\left(0.75 - \frac{|\frac{x_3}{6} - 0.35|}{|\lfloor 0.35 - \frac{x_3}{6} \rfloor + 0.35|}\right)}{0.75} \right.$$
$$\left. - \min\left(0, \left\lfloor 0.85 - \frac{|\frac{x_3}{6} - 0.35|}{|\lfloor 0.35 - \frac{x_3}{6} \rfloor + 0.35|} \right\rfloor\right) \frac{0.2\left(\frac{|\frac{x_3}{6} - 0.35|}{|\lfloor 0.35 - \frac{x_3}{6} \rfloor + 0.35|} - 0.85\right)}{0.15} \right)^{0.02}$$
$$+ \frac{4}{7}\left(0.8 + \min\left(0, \left\lfloor \frac{|\frac{x_4}{8} - 0.35|}{|\lfloor 0.35 - \frac{x_4}{8} \rfloor + 0.35|} - 0.75 \right\rfloor\right) \frac{0.8\left(0.75 - \frac{|\frac{x_4}{8} - 0.35|}{|\lfloor 0.35 - \frac{x_4}{8} \rfloor + 0.35|}\right)}{0.75} \right.$$
$$\left. - \min\left(0, \left\lfloor 0.85 - \frac{|\frac{x_4}{8} - 0.35|}{|\lfloor 0.35 - \frac{x_4}{8} \rfloor + 0.35|} \right\rfloor\right) \frac{0.2\left(\frac{|\frac{x_4}{8} - 0.35|}{|\lfloor 0.35 - \frac{x_4}{8} \rfloor + 0.35|} - 0.85\right)}{0.15} \right)^{0.02}$$
$$+ 2\left(1 - \cos\left(\frac{\pi(\frac{x_1}{2}^{0.02} + 2\frac{x_2}{4}^{0.02})}{6}\right)\right), \quad (4.43)$$

and objective function 2 in terms of $\mathbf{x}$:

$$f_2(\mathbf{x}) = \frac{3}{7}\left(0.8 + \min\left(0, \left\lfloor \frac{|\frac{x_3}{6} - 0.35|}{|\lfloor 0.35 - \frac{x_3}{6}\rfloor + 0.35|} - 0.75 \right\rfloor\right) \frac{0.8\left(0.75 - \frac{|\frac{x_3}{6} - 0.35|}{|\lfloor 0.35 - \frac{x_3}{6}\rfloor + 0.35|}\right)}{0.75} \right.$$
$$\left. - \min\left(0, \left\lfloor 0.85 - \frac{|\frac{x_3}{6} - 0.35|}{|\lfloor 0.35 - \frac{x_3}{6}\rfloor + 0.35|} \right\rfloor\right) \frac{0.2\left(\frac{|\frac{x_3}{6} - 0.35|}{|\lfloor 0.35 - \frac{x_3}{6}\rfloor + 0.35|} - 0.85\right)}{0.15} \right)^{0.02}$$
$$+ \frac{4}{7}\left(0.8 + \min\left(0, \left\lfloor \frac{|\frac{x_4}{8} - 0.35|}{|\lfloor 0.35 - \frac{x_4}{8}\rfloor + 0.35|} - 0.75 \right\rfloor\right) \frac{0.8\left(0.75 - \frac{|\frac{x_4}{8} - 0.35|}{|\lfloor 0.35 - \frac{x_4}{8}\rfloor + 0.35|}\right)}{0.75} \right.$$
$$\left. - \min\left(0, \left\lfloor 0.85 - \frac{|\frac{x_4}{8} - 0.35|}{|\lfloor 0.35 - \frac{x_4}{8}\rfloor + 0.35|} \right\rfloor\right) \frac{0.2\left(\frac{|\frac{x_4}{8} - 0.35|}{|\lfloor 0.35 - \frac{x_4}{8}\rfloor + 0.35|} - 0.85\right)}{0.15} \right)^{0.02}$$
$$+ 4\left(1 - \frac{\frac{x_1}{2}^{0.02} + 2\frac{x_2}{4}^{0.02}}{3} - \frac{\cos\left(10\pi \frac{\frac{x_1}{2}^{0.02} + 2\frac{x_2}{4}^{0.02}}{3} + \frac{\pi}{2}\right)}{10\pi}\right). \quad (4.44)$$

4.4 Conclusion: Continuous optimisation test problems

This chapter covered the test suite used for this study. It comprises problems from five test suites found in literature: the MOPs, ZDT problems, L_1ZDT problems, R problems and the WFG problems. The test suite comprises 46 problems in total, of which 38 were implemented in Matlab by the researcher. The test suite adheres to the recommendations regarding the composition of a test suite made by Huband *et al.* (2006).

The next chapter looks at the combinatorial cases investigated.

Chapter 5

Combinatorial test problem – the mission-ready resource problem

The previous chapter covered the unconstrained, continuous problems investigated in this study.

This chapter will look at the static, combinatorial test problem used to compare algorithm performance. First, the mission-ready resource (MRR) problem is introduced, followed by the general formulation of the MRR. Then, details about the specific cases used for this study are provided, followed by a short explanation of how the continuous optimisation algorithms discussed in Chapter 3 were adapted for discrete optimisation. Finally, constraint-handling strategies are discussed.

5.1 An introduction to the mission-ready resource problem

The mission-ready resource (MRR) problem stems from military decision making. Wakefield (2001) introduced the MRR problem in order to address a discrepancy between what combatant commanders need and what logisticians can provide. An MRR is defined to be a combination of resources (such as an aircraft, pilot, fuel, munitions, support equipment and personnel). Different MRRs are more or less suitable to different tasks. The degree to which an MRR is suited to a task is referred to as its task suitability. Each MRR also has a lift cost associated with it. This lift cost can be quantified as the weight and/or the volume of the MRR.

The availability of MMRs vary over time, as do the combat requirements. However, the model presented by Wakefield (2001) is not a dynamic model, but rather a static one that can be used for resource assignment at different points in time.

The goal of the MRR problem is to maximise task suitability while minimising lift cost.

5.2 Formulation of the mission-ready resource problem

The formulation of the MRR problem is based on the pilot problem described by Stephen Schwartz in an unpublished report (Wakefield, 2001).

Let $x_{i,j}$ be the number of MRRs of type j allocated to task i, with m task types and n MRR types.

The first objective is to maximise task suitability, with

$$f_1(\mathbf{x}) = \sum_{j=1}^{n} \sum_{i=1}^{m} \delta_{i,j} x_{i,j}, \tag{5.1}$$

where $\delta_{i,j}$ is the suitability of MRR type j for executing task i.

The second objective is to minimise the lift cost. Wakefield (2001) splits this up into two objectives: minimising the weight of the allocated MRRs, and minimising the volume of the assigned MRRs. The weight of the MRRs assigned is defined as

$$f_2(\mathbf{x}) = \sum_{j=1}^{n} \sum_{i=1}^{m} \varphi_j x_{i,j}, \tag{5.2}$$

where φ_j is the weight of MRR type j. Similarly, the volume of the assigned MRRs is

$$f_3(\mathbf{x}) = \sum_{j=1}^{n} \sum_{i=1}^{m} \eta_j x_{i,j}, \tag{5.3}$$

where η_j is the volume of MRR type j.

The objectives are subject to two constraints:

1. The number of MRRs assigned to tasks of type i must be equal to the task requirement R_i (the number of MRRs required) for tasks of type i.

2. The number of MRRs of type j that are assigned cannot exceed MRR availability A_j (the number of MRRs that are available) of MRRs of type j.

Constraint 1 is formulated as

$$\sum_{j=1}^{n} x_{i,j} = R_i, \quad \forall \quad i, \tag{5.4}$$

and constraint 2 as

$$\sum_{i=i}^{m} x_{i,j} \leq A_j, \quad \forall \quad j. \tag{5.5}$$

The values of $x_{i,j}$ should be non-negative integer values.

5.3 Details about the cases used for this study

The cases used for this study are based on the work done by Wakefield (2001). He suggested using a problem with three tasks and five MRR types. He assumed that an infinite number of all MRR types is available and that the number of tasks to be accomplished is the constraining factor.

The task suitability, volume and weight values selected by Wakefield (2001) are theoretical, but realistic. The task suitability matrix along with the weight and volume of the different MRR types are shown in Table 5.1.

Table 5.1: Task suitability, weights and volumes of different MRR types. Weights are measured in short stones, while volume is measured in cubic feet.

	Task 1	**Task 2**	**Task 3**	**Weight**	**Volume**
MRR 1	0.8	0.4	0.001	20.2	1650
MRR 2	0.3	0.8	0.001	28.5	2475
MRR 3	0.6	0.6	0.1	35.7	2887.5
MRR 4	0.001	0.001	0.8	19.9	1705
MRR 5	0.001	0.001	0.4	22.5	2200

Three different combinations of the number of each task that needs to be accomplished during a period will be considered. These combinations were suggested by Wakefield (2001) and are shown in Table 5.2.

Table 5.2: Number of tasks of type i requiring MRRs per period.

Index	Task 1	Task 2	Task 3	Total tasks
Combination A	10	5	1	16
Combination B	30	75	45	150
Combination C	60	90	150	300

However, for two main reasons, we will ignore the assumption by Wakefield (2001) that an infinite number of each MRR type is available. First, it is not possible for an infinite number of MRRs to be available. It is possible that there are enough MRRs available so that MRR availability is not the binding constraint. For this study, such cases are preferred to cases where infinite availability is assumed. In addition to cases where MRR availability is large enough not to be binding, cases where the MRR availability constraint could be binding are also considered.

The second reason for ignoring this assumption is that the total size of the decision space is $(\textit{The number of MRRs available+1})^{\textit{Number of variables}}$. Keeping the assumption means that the decision space is of infinite size. Ignoring the assumption by Wakefield (2001) decreases the total number of possible solutions, subsequently simplifying the task of solving the MRR cases using multi-objective optimisation algorithms.

Six cases are investigated in total. For each combination of tasks, a case where MRR availability can be a binding constraint and a case where MRR availability is large enough not to be binding, is investigated. For the sake of simplicity, it is assumed, for all cases, that the number of MRRs available are equal for all MRR types. The investigated cases are summarised in Table 5.3.

Initial experiments with the three objectives suggested by Wakefield (2001) produced Pareto fronts that were essentially only two-dimensional. This can be attributed to the fact that the weight and volume objectives are not truly conflicting. Only conflicting objectives should be taken into account, as objectives that are not conflicting have a single optimal answer (Zitzler, 1999). Even though the weight and volume objectives do not conflict with one another, the first objective (maximising task suitability) conflicts with both. Because of this, using a combination of the task suitability objective and the weight objective would result in a Pareto front roughly equivalent to

Table 5.3: Summary of MRR cases investigated.

	Number of task i to be executed			
Index	**Task 1**	**Task 2**	**Task 3**	**MRRs available**
Case A	10	5	1	5 × 5
Case B	10	5	1	5 × 20
Case C	30	75	45	5 × 50
Case D	30	75	45	5 × 200
Case E	60	90	150	5 × 100
Case F	60	90	150	5 × 400

the Pareto front produced by a combination of the task suitability objective and the volume objective. For no specific reason, the latter was preferred to the former.

5.4 From continuous to discrete optimisation

The algorithms discussed in Chapter 3 are all designed for continuous optimisation. Discrete optimisation versions of both the CEM and the MOO CEM exist. However, for the sake of simplicity, it was decided to adapt all the continuous optimisation algorithms to discrete optimisation by rounding all continuous variables to the smallest integer (this is called *flooring*).

Flooring is simple and consistent. Every time any algorithm chooses a value between a and $a+0.9999999$, the objective functions and constraints are evaluated using a. This procedure is similar to the procedure used to draw discrete random values.

5.5 Constraint handling

Unlike the continuous problems discussed in Chapter 4, the MRR is subject to constraints other than box constraints. The focus of this section falls on possible strategies for handling these constraints when solving the MRR, the constraint-handling strategies experimented with and the final method used for handling the constraints.

5.5.1 Constraint-handling strategies found in literature

Talbi (2009), Deb (2001) and Coello Coello *et al.* (2007) provide good overviews of popular constraint-handling strategies, including:

- **Rejecting infeasible solutions** – Infeasible solutions are discarded during the search. This is simple to implement, but only effective when the proportion of feasible solutions is relatively large.
- **Penalising infeasible solutions** – Infeasible solutions are considered during the search process, but objective function values are penalised when solutions are infeasible. These penalties can be linear or non-linear, and static, dynamic or adaptive. In spite of the ease of implementation of penalty functions, a significant disadvantage of using penalty functions is that choosing a penalty function and parameters suited to a problem requires extensive experimentation.
- **Repairing infeasible solutions** – A heuristic transforms infeasible solutions into feasible solutions. Such heuristics are specific to the problem at hand.
- **Treating constraints as objectives** – Constraint functions are ranked together with objective functions. For the Pareto ranking, constraints can either be summed together and treated as a single function in addition to the objective functions, or as separate functions.
- **Selecting for feasibility** – Selection rules preferring feasible solutions over infeasible solutions are used.
- **Decoding strategies** – Procedures where solutions in the search space are mapped to a space consisting of only feasible solutions. These feasible solutions are then evaluated.
- **Preserving feasibility of solutions** – Using problem-specific representations and operators, only feasible solutions are generated.

5.5.2 Unsuccessful constraint-handling strategies

Initially, the idea of comparing the abilities of the different optimisation algorithms to find near-optimal solutions for the MRR cases with minimal guidance and relatively small numbers of evaluations appealed greatly to the researcher. After extensively experimenting, without success, with different constraint-handling strategies, population sizes and increasingly large numbers of maximum evaluations, the researcher came to a conclusion similar to that of Wakefield (2001): without correcting infeasible solutions to be feasible, there are too many possible solutions to be able to find near-optimal solutions.

The researcher speculated that the fact that the decision space is no longer assumed to be infinite could possibly result in different results from that of Wakefield (2001). But even for Case B where only 16 tasks require MRR assignment and only 20 units of each MRR type are available, there are $21^{15} = 6.8 \times 10^{19}$ possibilities for a search algorithm to investigate, of which (according to Wakefield (2001)) only roughly 630 000 are feasible. That translates to only one in every 1.04×10^{14} solutions being feasible.

For the sake of completeness, the unsuccessful constraint-handling strategies are described briefly:

- **Using a sum penalty function** – a penalty function was added to objectives function values associated with infeasible solutions before ranking. Unfortunately, all the solutions are infeasible most of the time, and an algorithm cannot benefit from a difference between feasible solutions where no penalty is added and infeasible solutions where penalties are added.

- **Using a multiplicative penalty function** – objective function values of infeasible solutions are multiplied with a penalty function before ranking. Similar to the sum penalty function, the fact that a very small percentage of solutions is feasible means that an algorithm cannot learn from the difference in function values for feasible and infeasible solutions.

- **Ranking constraints with objective function values** – the constraints are treated as objective functions and ranked with the objectives. For a solution to outrank another solution, it has to be at least as good as or better than the other solution in all its objectives. As the number of objectives increases, it becomes

more difficult for one solution to clearly outrank other solutions (Di Pierro *et al.*, 2007). This is because even though the solution might be better than another solution in three out of four objectives, the other solution might outperform it on the fourth. As a result, ranking becomes weaker as the number of objectives increases. In order to overcome this, both Pareto ranking and preference order ranking schemes were investigated. Due to the small percentage of feasible solutions, this strategy still did not result in reasonable numbers of feasible solutions.

- **Ranking constraints prior to ranking objective function values** – the constraints values are ranked (using Pareto ranking) before the objectives are ranked. The problem with this strategy is that, if a feasible solution was found, that one solution would outrank all the infeasible solutions in the first round of ranking. Only that solution would go on to the second round of ranking and the algorithms that select the best few ranks of a solution would get stuck searching around that one feasible solution. This was the strategy most likely to find a feasible solution, but it would rarely find more than one feasible solution.

In the end, all these strategies were discarded in favour of fixing infeasible solutions. This strategy is described in the following subsection.

5.5.3 Final constraint-handling strategy

After experimenting with the unsuccessful constraint-handling strategies discussed above, the researcher came to the conclusion that simply penalising infeasible solutions would not be a powerful enough constraint-handling strategy to allow the algorithms to find near-optimal solutions. Instead, the algorithms should be provided with feasible solutions in some way. Once this was decided upon, a mechanism that could provide feasible solutions without being heavily biased towards a specific algorithm had to be found.

It was decided that the best way to do this, would be to have a mechanism that serves as an extension of the function evaluation. Such a mechanism would effectively only complicate the objective function, but this complication would be similar for all the algorithms.

The mechanism would be fixing infeasible solutions by scaling down a solution proposed by an algorithm to satisfy the constraints. This is done by drawing up a probability distribution function for assigning MRRs of type j to tasks of type i.

An example using Case B further illustrates how solutions were fixed. Suppose

	MRR 1	*MRR 2*	*MRR 3*	*MRR 4*	*MRR 5*	$\sum_{j=1}^{n}$
Task 1	9	12	16	10	12	59
Task 2	12	11	9	19	2	53
Task 3	13	3	19	13	3	51
$\sum_{i=1}^{m}$	34	26	44	42	17	

(5.6)

is an original floored solution for MRR Case B. The values of the matrix in (5.6) are scaled so that the number of MRRs assigned to task i is equal to the task requirement i (R_i):

	MRR 1	*MRR 2*	*MRR 3*	*MRR 4*	*MRR 5*	$\sum_{j=1}^{n}$
Task 1	1.5254	2.0339	2.7119	1.6949	2.0339	10
Task 2	1.1321	1.0377	0.8491	1.7925	0.1887	5
Task 3	0.2549	0.0588	0.3725	0.2549	0.0588	1

. (5.7)

This will be referred to as the *task scaled matrix*.

The following steps are iterative and are repeated for $k = 1$ to $\sum_{i=1}^{m} R_i$:

Step 1 Scale the task scaled matrix by dividing it by the sum of all the values in the task scaled matrix (16 at this stage). This effectively results in a probability distribution function for assigning MRRs to tasks. The i, j^{th} entry of the probability distribution matrix represents the probability that MRR j will be assigned to task i:

	MRR 1	*MRR 2*	*MRR 3*	*MRR 4*	*MRR 5*
Task 1	0.0953	0.1271	0.1695	0.1059	0.1271
Task 2	0.0708	0.0649	0.0531	0.1120	0.0118
Task 3	0.0159	0.0037	0.0233	0.0159	0.0037

. (5.8)

Figure 5.1 shows the probability distribution function.

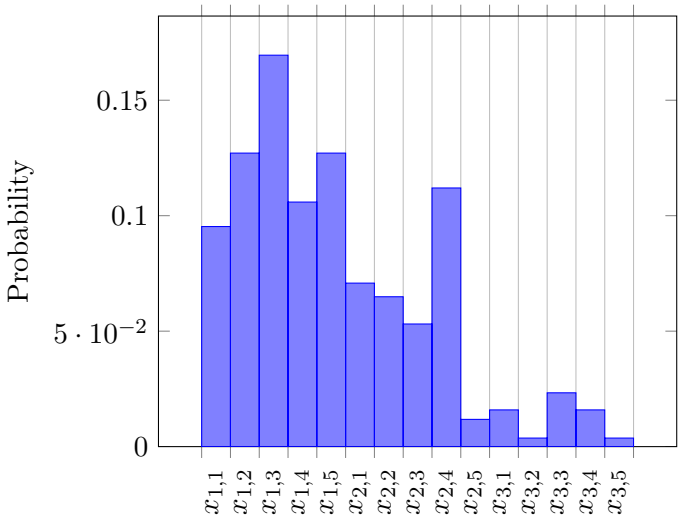


Figure 5.1: Probability distribution function for assigning an MRR of type j to task type i.

Step 2 The associated cumulative distribution function is:

	MRR 1	*MRR 2*	*MRR 3*	*MRR 4*	*MRR 5*
Task 1	0.0953	0.2225	0.3919	0.4979	0.6250
Task 2	0.6958	0.7606	0.8137	0.9257	0.9375
Task 3	0.9534	0.9571	0.9804	0.9963	1.0000

. (5.9)

The cumulative distribution function is shown in Figure 5.2.

Step 3 Select a uniformly distributed random number between zero and one, J.

Step 4 Use J and to draw an assignment from the cumulative distribution function in (5.9). For example, if $J = 0.3708$, then $x_{1,3} \leftarrow x_{1,3} + 1$:

	MRR 1	*MRR 2*	*MRR 3*	*MRR 4*	*MRR 5*
Task 1	0	0	1	0	0
Task 2	0	0	0	0	0
Task 3	0	0	0	0	0

. (5.10)

Step 5 If the number of task type i assigned in the feasible solution is equal to R_i, then row i in the task scaled matrix is set to zero.

Step 6 If the number of MRR type j assigned in the feasible solution is equal to A_j, then column j in the task scaled matrix is set to zero.

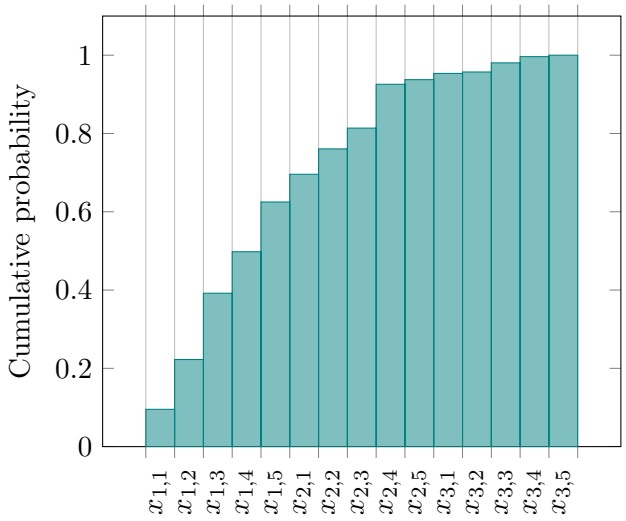


Figure 5.2: Cumulative distribution function for assigning an MRR of type j to task type i.

Even though the algorithm for fixing solutions is stochastic in nature, it constructs feasible solutions according to the ratios suggested by the optimisation algorithm. Although a ratio might not be translated to exactly the same feasible solution every time, generally the feasible solutions for a specific ratio would be similar to one another – enough so that an optimisation algorithm could learn which ratios give better and which give worse results.

However, if an optimisation algorithm suggests assigning zero MRRs, the algorithm does not fix the solution. This is because creating ratios where the algorithm did not put forth any values would seem completely arbitrary from an optimisation algorithmic point of view: for one set of zeros, very bad function values result; for another, very good function values. In such cases the solution is evaluated as is and a sum penalty function is used to penalise the resulting constraint violations.

5.6 Conclusion: Combinatorial test problem – the mission-ready resource problem

This chapter introduced the mission-ready resource (MRR) problem and the general formulation thereof. Details about the specific cases used for comparison were provided. A short explanation about how the continuous optimisation algorithms discussed in

Chapter 3 were adapted for discrete optimisation followed. Finally, constraint-handling strategies were discussed.

The next chapter will look at the dynamic, stochastic problem that was studied.

Chapter 6

Simulation case – the buffer allocation problem

The previous two chapters respectively discussed the unconstrained, continuous test suite and constrained, combinatorial cases investigated by the researcher.

This chapter will focus on the dynamic, stochastic problem that was studied. A few cases of a dynamic, stochastic buffer allocation problem (BAP) were optimised using a combination of Matlab and the simulation software package, Simio. An introduction to the BAP and details of the test cases are provided below.

6.1 An introduction to the buffer allocation problem

The buffer allocation problem (BAP) is found in manufacturing, telecommunications, material-handling systems and service provision industries such as health care.

It involves either allocating a predetermined number of buffers optimally or finding the optimal configuration of buffers without a predetermined limit on the total number of buffers available (Bekker, 2012).

Typically, the BAP is formulated as a single-objective problem in which the total cost of the buffers is to be minimised subject to a constraint on the minimum allowable throughput rate (Cruz *et al.*, 2010).

There is a definite trade-off between the throughput rate and the total buffer cost: as the buffer space increases, the throughput rate increases, but the buffer cost also increases (Cruz *et al.*, 2010). This trade-off is lost when using single-objective opti-

misation. In this light, Cruz *et al.* (2010) proposed using a multi-objective approach to the BAP. They recommend minimising the total cost as one objective and maximising throughput rate as the other.

A similar trade-off exists between the work-in-progress (WIP) of a system and the throughput rate of the system. As the number of buffers increases, the throughput rate increases, but, unfortunately, the WIP also increases. Bekker (2012) prefers using this trade-off to the one proposed by Cruz *et al.* (2010).

6.2 The BAP cases investigated

The cases investigated for this study are variations of a manufacturing process consisting of five machines. There are four buffers, as it is assumed that the buffers before the first and after the last machine are infinite. The configuration of the system is shown in Figure 6.1.

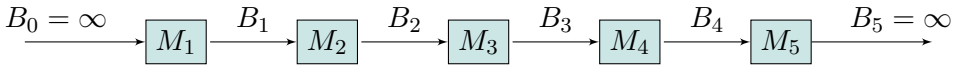


Figure 6.1: Configuration of the BAP case investigated $M_1, \ldots, M_5$ with finite buffers $B_1, \ldots, B_4$ in a queuing network.

The problem studied is stochastic and dynamic with exponentially distributed processing rates and machine repair times. Machine failures occur based on the number of jobs processed per machine and are Poisson distributed. The means applicable to each machine are shown in Table 6.1. Processing times are distributed $Expo(\beta_p)$, failure counts are distributed $Pois(\lambda_f)$, and repair times are distributed $Expo(\beta_r)$.

Table 6.1: Mean processing times, failure counts and repair times for the BAP cases.

	β_p	$\lambda_\mathbf{f}$	β_r
Machine 1	1	20	2
Machine 2	$\frac{1}{1.1}$	20	2
Machine 3	$\frac{1}{1.2}$	20	2
Machine 4	$\frac{1}{1.3}$	20	2
Machine 5	$\frac{1}{1.4}$	20	2

Because of the dynamic, stochastic nature of the problem, it was implemented in the simulation software, Simio. Figure 6.2 shows how simulation modelling can be used as an MOO decision support system.

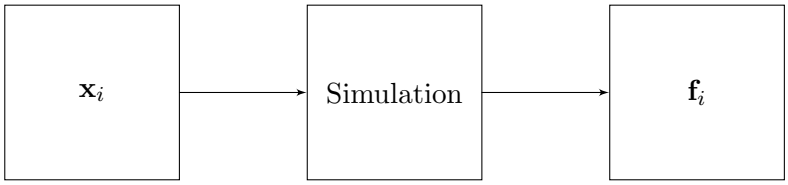


Figure 6.2: Simulation as an MOO decision support system.

In order to solve the dynamic BAP optimisation problem using the simulation model, Matlab and Simio were combined using a C♯ executable as discussed in Appendix C.

For the cases investigated, it was assumed that there were no predetermined limits on the number of buffers available. The aim was thus to find the Pareto optimal configurations of buffers.

However, in order to decrease the size of the search space, upper bounds were imposed on B_1 to B_4.

Three different sets of upper bounds were experimented with. For the first set, the upper bounds chosen resulted in a relatively small search space.

The size of the search space for the second set is the same as that of the first set. For this set, the upper bounds were determined by running the simulation model with infinite buffer sizes and determining the 95^{th} percentiles of the buffer sizes for Machines 2 to 5. These 95^{th} percentiles served as upper bounds for the multi-objective optimisation algorithms.

For the third set, the upper bounds were all equal to 10 000 in order to drastically increase the size of the search space. Results for the continuous and mission-ready resource (MRR) cases showed that the size of the decision space is an important indicator for algorithm performance. The upper bounds for the buffers for all three cases are shown in Table 6.2.

The box constraints were the only constraints, and the combination of objectives preferred by Bekker (2012) was used. In other words: WIP was minimised, while the throughput rate was maximised. The optimisation was subject only to the box constraints.

Table 6.2: Maximum buffer sizes allowed for the BAP cases investigated.

Buffer	Case A	Case B	Case C
B_1	3	16	10 000
B_2	5	9	10 000
B_3	9	5	10 000
B_4	16	3	10 000

6.3 Conclusion: Simulation case – the buffer allocation problem

This chapter covered the dynamic, stochastic problem studied. An introduction to the BAP and details of the test cases were provided.

The next chapter will discuss details about how experiments were set up.

Chapter 7

Experimental design

The previous three chapters discussed the test problems that were used for algorithm comparison. This chapter will look at how algorithm performance was measured and at details relating to the experiments – such as the parameter settings and sample sizes used.

7.1 Performance assessment

Many methods for comparing multi-objective optimisation algorithms and the Pareto fronts they achieve exist. Knowles *et al.* (2005) differentiate between two main categories of these methods: applying statistical tests directly to the found Pareto fronts, or, alternatively, reducing each Pareto front to a single value through the use of performance indicators and then applying statistical tests to these performance indicator values. For this study, we will make use of the second class of methods.

7.1.1 Performance indicators

Many performance indicators for multi-objective optimisation algorithms have been suggested. For this study, we will consider four characteristics of these indicators: whether the indicator is unary or binary, whether it requires knowledge about the Pareto front or not, the information it provides in terms of dominance relations and the information it generally provides about the Pareto fronts themselves.

Unary performance indicators assign a single value to each approximated Pareto front. This value represents an absolute quality of the front. Binary performance indi-

cators, on the other hand, compare two approximated Pareto fronts with one another and return a value that is an evaluation of the relative quality if the one front with respect to the other (Lizárraga *et al.*, 2009). Binary indicators are impractical for a study of this kind as algorithms have to be compared in pairs, which makes it difficult to draw conclusions about the overall performance of an algorithm.

Many performance indicators require knowledge of the true Pareto front. Unfortunately, the Pareto front is not always known and measures that rely on knowledge of the true front cannot be used in these cases. Even when the true Pareto front can be calculated – for test problems such as the Zitzler-Deb-Thiele (ZDT), L_1ZDT and rotated (R) problems, for example – the true front calculated depends on the number of solutions allowed during calculation and the random values used for these solutions. This means that measures that rely on knowledge of a true front will behave differently each time a true front is calculated or the size of the front is varied.

Zitzler *et al.* (2003) define five dominance relations:

- **Algorithm A strictly dominates Algorithm B** – for every member of Front B, there exists at least one member in Front A that outperforms it in all the objectives. This is the highest form of superiority.

- **Algorithm A dominates Algorithm B** – for every member of Front B, there exists at least one member in Front A that is not worse than the objective values of the member in Front B in all the objectives and outperforms the member in Front B in at least one objective.

- **Algorithm A is better than Algorithm B** – for every member of Front B, there exists at least one member in Front A that is not worse than the objective values of the member in Front B in all the objectives. For an Algorithm A to be called better than Algorithm B, the two Pareto front sets may not be equal to one another. This is the lowest form of superiority, but the one most commonly used in literature.

- **Algorithm A weakly dominates Algorithm B** – for every member of Front B, there exists at least one member in Front A that is not worse than the objective values of the member in Front B in all the objectives. In this case the two Pareto front sets are allowed to be equal.

- **Algorithm A is incomparable with Algorithm B** – this means that Algorithm A does not weakly dominate Algorithm B, nor does Algorithm B weakly dominate Algorithm A.

Zitzler *et al.* (2003) further define a difference between *compatibility* and *completeness*. In order to explain these definitions, we will use the "better" dominance relation described above. Compatibility means that whenever the indicator value of Front A is better than that of Front B, Front A will be "better" than Front B. Completeness means that whenever Front A is "better" than Front B, an indicator will have a better value for Front A than for Front B as a result. Compatibility says something about the interpretation of the results produced by an indicator: can one say that Front A is "better" than Front B if the indicator value for Front A is better than that of Front B? If an indicator is compatible with a dominance relation (such as "better"), one can. Completeness, on the other hand, says something about the ability of an indicator to identify cases where Front A outperforms Front B in terms of some dominance relation. If an indicator is complete in terms of some dominance relation, the indicator will always show a better indicator value for Front A than for Front B when Front A is "better" than Front B.

From this, it is clearly preferable to use a performance indicator that is compatible and complete with regard to some dominance relation.

Finally, it is important to consider how an indicator measures performance. Does it measure the spread of the members of the Pareto front, or the number of members or how close the known Pareto front is to the true front? In general, it is important to measure at least the spread of the front found and its proximity to the true front (Bekker, 2012). Figure 7.1 shows an approximated front that is well spread, but distant from the true front, while Figure 7.2 shows a poorly spread approximated front in close proximity to the true front. An ideal approximated front is well spread and in close proximity to the true front.

7.1.1.1 The hypervolume indicator

The hypervolume indicator was first introduced by Zitzler & Thiele (1998) and Zitzler & Thiele (1999). It is also referred to as the S-measure or Lebesque measure. It essentially measures the volume (the area for the two-objective case) of the polytope

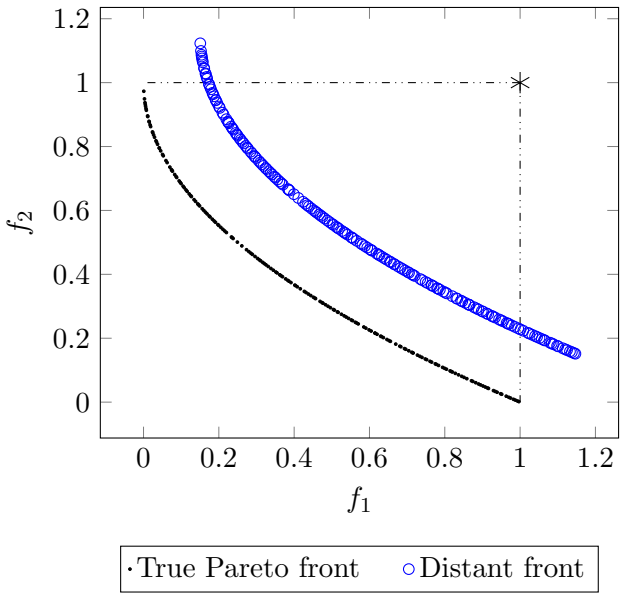


Figure 7.1: A well spread but distant front shown relative to a Pareto front.

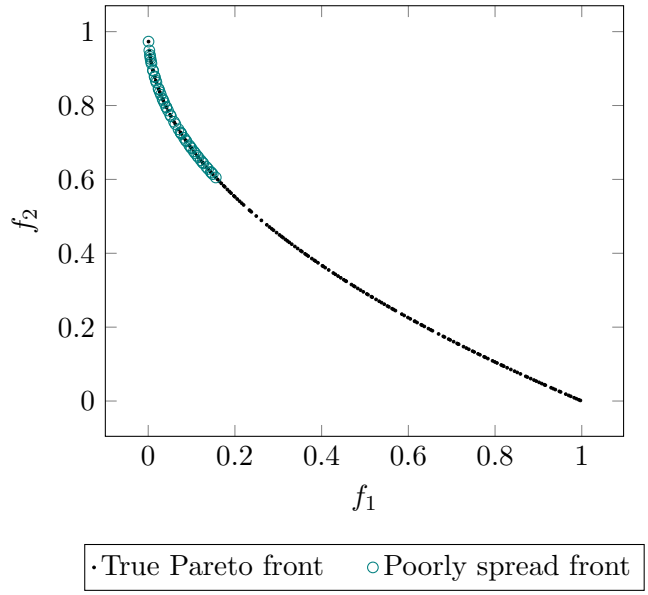


Figure 7.2: A poorly spread front in close proximity to the Pareto front shown relative to the Pareto front.

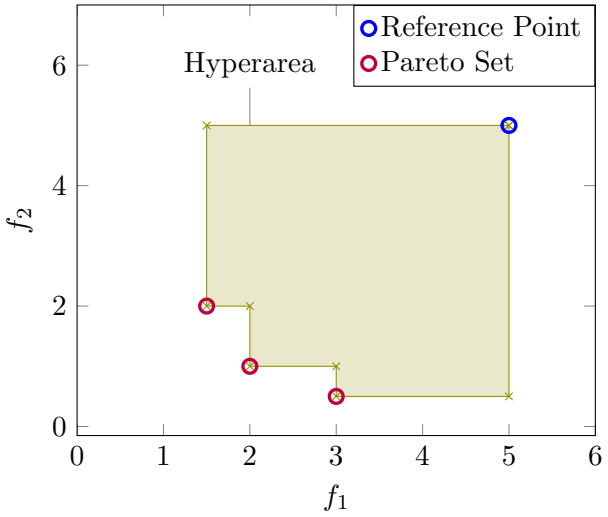


Figure 7.3: Example of a hypervolume (hyperarea) and reference point.

(the polygon for the two-objective case and the polyhedron for the three-objective case) between the Pareto front and a predetermined reference point. Figure 7.3 shows an example of such a polytope (a polygon actually, as it is calculated for two objectives), and a hypervolume (hyperarea, since there are two objectives).

Both unary and binary versions of the hypervolume indicator exist (Zitzler *et al.*, 2003). For this study, the unary hypervolume indicator is used for the reasons discussed in Section 7.1.1.

The hypervolume indicator does not require information about the true Pareto front. However, it does require that the researcher select a reference point from which the hypervolume will be calculated. Knowledge of the true Pareto front will enable better selection of this reference point.

According to Zitzler *et al.* (2007), the hypervolume indicator is compatible with the "weakly dominates" dominance relation. This means that if the hypervolume of Front A is better than the hypervolume of Front B, then Front A is not worse than Front B.

The hypervolume indicator is complete with respect to the "better" dominance relation (discussed in Subsection 7.1.1) (Knowles *et al.*, 2005). All cases where Front A is better than Front B are thus detected by the indicator.

The hypervolume indicator provides information about the spread of the Pareto front and about how well an algorithm performs in terms of finding near-optimal solu-

tions. A larger spread will result in a better hypervolume. Similarly, the closer solutions are to being optimal, the larger the hypervolume.

7.1.1.2 Relative run times

In addition to the hypervolume indicator, the run times of the algorithms are also recorded. Because the absolute run times depend on the computers used, the run times of algorithms relative to one another are shown in Appendices A and B. The largest median run time is set equal to one. All other run times are shown as fractions of this.

For this study, the performance of the algorithms in terms of run times is of secondary importance when compared to performance measured by the hypervolume indicator. The relative run times are recorded mainly out of research curiosity.

7.1.2 Significance testing

As mentioned before, the approach to performance testing adopted for this study involves applying significance tests on the hypervolume indicator values. Several significance tests were considered, including the parametric two-sample t-test, the non-parametric Kruskall-Wallis test and the non-parametric Mann-Whitney U-test.

The Mann-Whitney U-test compares the median values of two distributions, while the two-sample t-test compares mean values (Heiberger & Holland, 2004). The two-sample t-test requires data to be normally distributed.

Figure 7.4 shows box plots of some of the results obtained. The box plots provide an indication of the distributions of the results.

From Figure 7.4, it can be seen that data are not normally distributed. The Mann-Whitney U-test is thus preferred to its parametric counterpart, the two-sample t-test.

The Mann-Whitney U-test is also preferred to the Kruskal-Wallis test (the equivalent of the Mann-Whitney U-test for three or more variables (McDonald, 2008)), because the result of the Kruskal-Wallis test simply shows if any one of the variables comes from a distribution that differs significantly from the distributions of the other variables. For this study, using the Kruskall-Wallis test would require further Mann-Whitney U-tests to identify exactly which distributions differ from the rest.

Right-tailed Mann-Whitney U-tests were done on all pairs of algorithms in order to determine if an Algorithm i performed significantly better than another Algorithm j at a 5% significance level. Table 7.1 shows an example of the results for all the

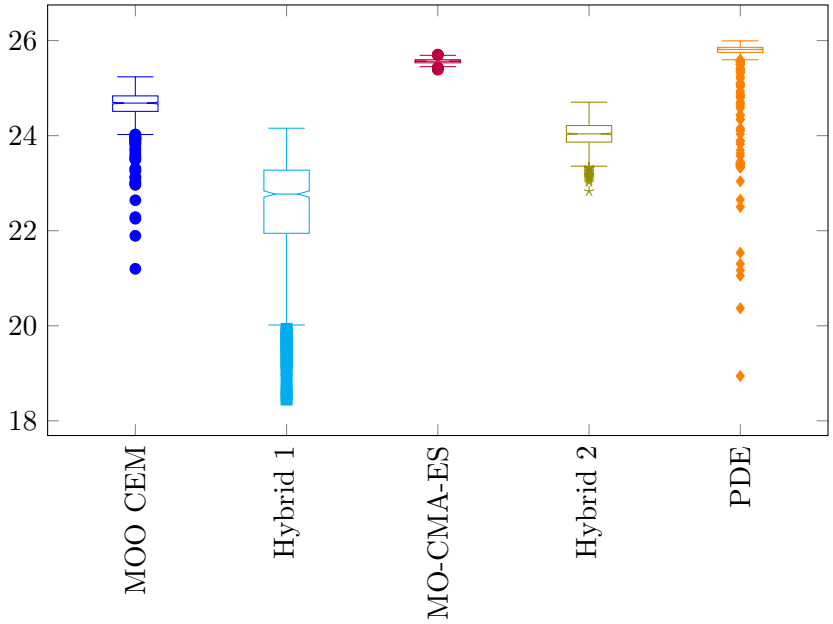


Figure 7.4: A box plot of hypervolumes achieved.

pairs. If entry $ij = 1$, it indicates that Algorithm i achieved a significantly higher hypervolume than Algorithm j at a 5% significance level. The *Outperformed* column sums the total number of algorithms that Algorithm i outperformed, whereas the *Rank* column indicates which algorithm performed best on the problem at hand (with 1 being the best and 5 being the worst).

Table 7.1: A sample of Mann-Whitney U-test results.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	0	1	0	2	**3**
Hybrid 1	0	-	0	0	0	0	**5**
MO-CMA-ES	1	1	-	1	0	3	**2**
Hybrid 2	0	1	0	-	0	1	**4**
PDE	1	1	1	1	-	4	**1**

Once the right-tailed Mann-Whitney U-tests were completed, each algorithm is assigned a rank indicating how many algorithms it significantly outperformed for the problem at hand. A rank of 1 indicates that an algorithm significantly outperformed all the other algorithms, whereas a rank of 5 means that the algorithm did not significantly outperform any algorithms.

7.2 Experimental setup

7.2.1 Population size, number of evaluations and the sample size

The population sizes for all algorithms are set to a hundred for all test problems. For the unconstrained continuous problems, the maximum number of evaluations is set to 10 000 (or a hundred generations). A thousand replications of each experiment were done; all algorithms solved all the unconstrained problems a 1 000 times.

For the mission-ready resource (MRR) cases, the maximum number of evaluations was increased to 15 000. This is due to the increased complexity of these problems. Once again each algorithm solved each case a thousand times.

For the buffer allocation problem (BAP) simulation case, the maximum number of evaluations was set to 10 000. However, instead of running each algorithm a thousand times, a hundred replications of each algorithm was run. This is because of the increased run time of the simulation case. Evaluating the population (of size hundred) takes roughly 30 seconds. This is done a hundred times each time an algorithm solves the BAP. In other words, finding a Pareto front for the BAP once takes roughly 50 minutes. To do a thousand replications with a single algorithm would take about 833 hours, or 35 days. A hundred replications still took about three and a half days to run. For this study, the increase in accuracy was not worth an extra month of run time.

As discussed in Chapter 6, the BAP is run in Simio. The warmup period is set to 25 hours, with the run time of each replication set to 500 hours. For one simulation run, 25 replications of the model were run.

7.2.2 Algorithm-specific parameters

Adjustments to algorithm-specific parameters drastically influence algorithm performance. To simplify the research problem, the effect of changes to these parameters are not investigated in this study and parameter settings were kept constant for all the

test problems. The parameter settings used were decided upon based on the literature accompanying the MOO CEM, MO-CMA-ES and PDE. Table 7.2 shows the algorithm-specific parameter setting used for all problems.

Table 7.2: Algorithm-specific parameter setting used.

Algorithm	Parameter	Value
MOO CEM	p_h	0.3
MO-CMA-ES	c_c	$\frac{2}{V+2}$
	c_{cov}	$\frac{2}{V^2+6}$
	p_g	$\frac{1}{5.5}$
	c_s	$\frac{p_g}{2+p_g}$
	d	$1+\frac{V}{2}$
	p_t	0.44
PDE	c_r	0.7
Hybrid 1	p_h	0.3
	Cluster size	15
Hybrid 2	p_h	0.3
	c_c	$\frac{2}{V+2}$
	c_{cov}	$\frac{2}{V^2+6}$
	p_g	$\frac{1}{5.5}$
	c_s	$\frac{p_g}{2+p_g}$
	d	$1+\frac{V}{2}$
	p_t	0.44

7.3 Conclusion: Experimental design

This chapter focused on the measurement of algorithm performance and the details of the experimental design.

The next chapter is a summary and analysis of the experimental results.

Chapter 8

Analysis of experimental results

The previous chapter discussed details of the experimental design.

This chapter presents an analysis and discussion of the experimental results, in the order in which the problems were presented: the continuous problem results, the results for the mission-ready resource (MRR) cases, and finally the results for the buffer allocation problem (BAP) cases.

8.1 Summary of results for the unconstrained continuous problems

Five algorithms – the multi-objective optimisation cross-entropy method (MOO CEM), the multi-objective covariance matrix adaptation evolution strategy (MO-CMA-ES), Pareto differential evolution (PDE), Hybrid 1 and Hybrid 2 – were compared on a total of 46 problem instances. Each algorithm was run on each problem for a thousand replications. Algorithm performance was measured using the hypervolume indicator and relative run times. The relative run times were recorded out of interest and do not form a big part of the discussion to follow. The focus is on the relative performance of algorithms as measured by the hypervolume indicator.

Pairwise Mann-Whitney U-tests were done to identify instances where an algorithm significantly outperformed another (at a 5% significance level) in terms of hypervolume. In order to ease comparison of algorithms, each algorithm was assigned a rank for each problem depending on how many algorithms it significantly outperformed. If an algorithm outperformed all the other algorithms, it would be assigned a rank equal

to 1. An algorithm that did not outperform any other algorithm would be assigned a rank equal to 5. If no algorithm significantly outperformed the rest of the algorithms, all the algorithms would have a rank of 5. Tables showing the Mann-Whitney U-test results, box plots of hypervolumes and relative run times for all problem instances can be found in Appendix A. Table 8.1 shows a summary of the ranks achieved by the algorithms on all the unconstrained continuous problems. A value of 1 indicates that an algorithm significantly outperformed all four the other algorithms for that problem. A value of 5 indicates that an algorithm did not significantly outperform any of the other algorithms.

Table 8.1: Summary of relative performance ranks of algorithms on continuous problems.

Problem	Variables	Shape of front	Modality	Relationships	MOO CEM	Hybrid 1	MO-CMA-ES	Hybrid 2	PDE
MOP 1	1	Convex	U	✗	2	5	3	4	1
MOP 2	3	Concave	U	✗	3	4	2	5	1
MOP 3	2	Disconnected	M	✓	5	4	3	1	3
MOP 4	3	Disconnected	M	✗	3	5	2	4	1
MOP 6	2	Disconnected	M	✗	5	3	1	4	2
ZDT 1	30	Convex	U	✗	3	4	1	5	2
ZDT 2	30	Concave	U	✗	5	5	1	3	2
ZDT 3	30	Disconnected	M	✗	2	4	1	5	3
ZDT 4	10	Convex	M	✗	5	5	5	5	5
ZDT 6	10	Concave	M	✗	5	2	1	5	5
L_1ZDT 1	30	Convex	U	✓	2	5	1	4	3
L_1ZDT 2	30	Concave	U	✓	5	5	1	5	5
L_1ZDT 3	30	Disconnected	M	✓	3	5	1	4	2
L_1ZDT 4	10	Convex	M	✓	5	5	5	5	5
L_1ZDT 6	10	Concave	M	✓	2	3	1	5	5
R 1	10	Convex	M, D	✓	4	4	5	2	1
R 2	10	Disconnected	M	✓	1	2	5	4	3
R 3	10	Concave	M	✓	3	5	5	1	2

Continued on next page

Table 8.1 – continued from previous page

Problem	Variables	Shape of front	Modality	Relationships	MOO CEM	Hybrid 1	MO-CMA-ES	Hybrid 2	PDE
R 4	10	Convex	M, D	✓	3	4	1	5	2
WFG 1	4	Convex, mixed	U	✗	5	4	1	3	5
WFG 1	20	Convex, mixed	U	✗	4	2	1	3	5
WFG 1	100	Convex, mixed	U	✗	4	1	3	2	5
WFG 2	4	Convex, disconnected	M	✓	4	3	2	1	5
WFG 2	20	Convex, disconnected	M	✓	1	3	4	2	5
WFG 2	100	Convex, disconnected	M	✓	1	3	5	2	4
WFG 3	4	Convex, linear, degenerate	U	✓	3	2	4	1	5
WFG 3	20	Convex, linear, degenerate	U	✓	2	4	1	5	3
WFG 3	100	Convex, linear, degenerate	U	✓	1	2	4	3	5
WFG 4	4	Concave	M	✗	5	4	1	3	2
WFG 4	20	Concave	M	✗	5	4	1	3	2
WFG 4	100	Concave	M	✗	5	3	2	5	1
WFG 5	4	Concave	D	✗	5	2	1	3	4
WFG 5	20	Concave	D	✗	5	2	1	4	3
WFG 5	100	Concave	D	✗	3	1	4	2	5
WFG 6	4	Concave	U	✓	5	4	1	4	2
WFG 6	20	Concave	U	✓	5	2	1	4	3
WFG 6	100	Concave	U	✓	4	1	5	2	3
WFG 7	4	Concave	U	✗	3	5	2	4	1
WFG 7	20	Concave	U	✗	3	4	1	5	2
WFG 7	100	Concave	U	✗	2	4	3	5	1
WFG 8	4	Concave	U	✓	4	5	1	3	2
WFG 8	20	Concave	U	✓	3	4	1	5	2
WFG 8	100	Concave	U	✓	2	3	4	5	1
WFG 9	4	Concave	M, D	✓	5	4	1	3	2
WFG 9	20	Concave	M, D	✓	5	2	1	3	4
WFG 9	100	Concave	M, D	✓	4	1	5	2	3

It is difficult to draw conclusions about the performance of the different algorithms by simply looking at Table 8.1. Figure 8.1 more clearly reflects trends in relative

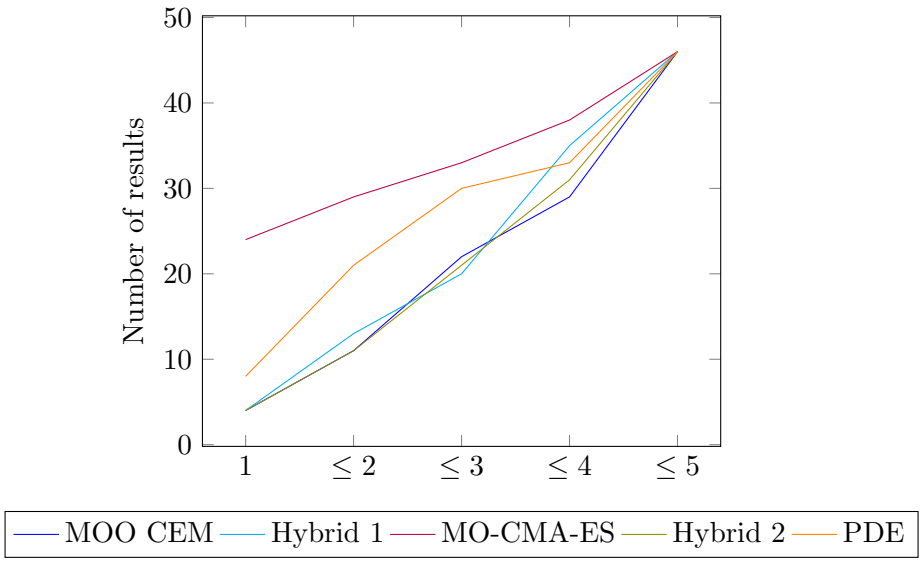


Figure 8.1: Overall performance of algorithms on continuous problems.

algorithm performance. It shows cumulative distributions of the number of times that each algorithm ranked first, second, third, fourth and fifth respectively.

Following from Figure 8.1 and Table 8.1:

- The MO-CMA-ES significantly outperformed all the other algorithms in 24 instances (roughly 52% of the time). It significantly outperformed at least three algorithms (ranking first or second) in 63% of the instances.
- The PDE significantly outperformed at least three algorithms in 46% of the test cases.

In general, the MO-CMA-ES performed best on the unconstrained continuous problems. The PDE performed second best overall, but at quite a large cost in terms of run time when compared to the other algorithms (refer to Appendix A). Very little differentiates the performances of the remaining three algorithms.

Algorithm performance will now be analysed with respect to the primary characteristics of the continuous test problems.

8.1.1 Performance relative to the number of decision variables

One of the primary characteristics of a problem is the number of variables it has. Figure 8.2 shows the relative performance of algorithms for few variables (10 of fewer variables), a medium number of variables (11 to 30 variables) and many variables (more than 30 variables).

There are visible differences in the relative performance of the algorithms depending on the number of variables. For the 22 problems with only a few variables, the MO-CMA-ES and PDE algorithms performed the best, while MOO CEM and Hybrid 1 performed the worst.

Fifteen problems are classified as having a medium number of variables. On 14 of these, the MO-CMA-ES significantly outperformed all the other variables. The PDE was the second best performer, outranking three other algorithms seven out of 15 times.

In contrast to its very good performance on a small and medium number of variables, the MO-CMA-ES is the worst performing algorithm on the nine problems that have many variables. Hybrid 1 performs the best when problems have a large number of variables, despite its relatively bad performance for fewer variables. It outranked at least three other algorithms in five of the nine test cases, and outperformed all the other algorithms in four of these. The MOO CEM also performs well on problems with many variables, outperforming at least three other algorithms in four of the nine test cases. In two of these cases the MOO CEM outperformed all the other algorithms. Hybrid 2 outperforms three other algorithms in five of the nine problems. The PDE algorithm outperformed the MO-CMA-ES but was arguably outperformed by the MOO CEM and the two Hybrid algorithms.

Although only nine problems have many variables, these nine problems were each run three times with the only difference being the number of decision variables: each problem was run for four, 20 and 100 variables. The MO-CMA-ES performed relatively well when solving the problems with four and 20 variables, but its performance dropped drastically when the number of variables was pushed up to 100.

It is believed that the difference in performance can be ascribed to the fundamental mechanisms according to which the algorithms work. The MO-CMA-ES relies on successfully updating a V by V matrix. If the matrix is successfully updated, the MO-CMA-ES is very powerful. However, for many variables, it is difficult to update

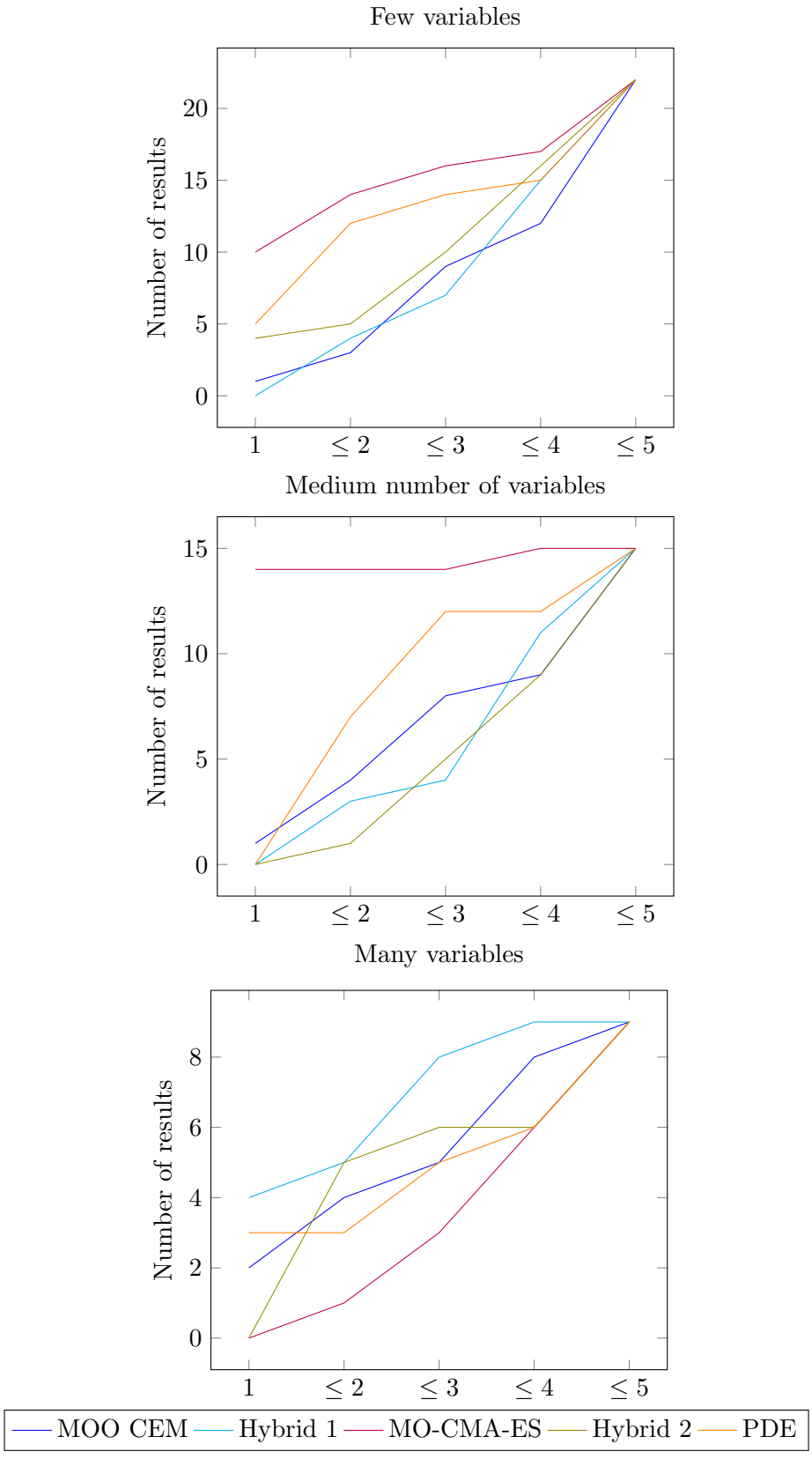


Figure 8.2: Algorithm performance relative to number of decision variables.

such a matrix successfully and the MO-CMA-ES subsequently performs worse than it would have done otherwise. The cases where the number of variables fall between 11 and 30 seem to be its sweet spot, where there are enough variables for it to be important to effectively keep track of relationships between them, but not so many that the MO-CMA-ES struggles to update the covariance matrix accurately.

The PDE has a simpler mechanism for keeping track of relationships between decision variables. While this mechanism might not be as effective as that of the MO-CMA-ES in the case for a medium number of variables, the PDE does not appear to be affected as badly by an increase in the number of decision variables.

It is believed that the improved relative performance of the MOO CEM and Hybrid 1 as the number of variables increases can be ascribed to the fact that neither of these algorithms relies on complicated mechanisms for keeping track of relationships between decision variables. For problems with small or medium numbers of variables, this lack of an intricate mechanism for keeping track of relationships between decision variables is a relative disadvantage, and the MO-CMA-ES outperforms both these algorithms in such cases. However, when the number of variables becomes too large for the MO-CMA-ES to successfully keep track of the relationships between them, the fact that the MOO CEM and Hybrid 1 do not rely on keeping track of such relationships, and subsequently cannot fail to successfully do so, becomes a relative strength.

Hybrid 1 differs from the MOO CEM only in the way clusters are created before histograms are drawn up. For the MOO CEM, histograms are drawn up for the entire elite, whereas each cluster has its own histograms when using Hybrid 1. The similar trends in performance – improved relative performance as the number of variables increases – therefore does not come as a surprise.

The improved relative performance of Hybrid 2 as the number of variables increases is ascribed to the fact that it contains elements of the MOO CEM. The fact that Hybrid 2 marginally outperforms the MOO CEM (and arguably Hybrid 1) on the smaller problems is ascribed to the fact that it also contains elements of the MO-CMA-ES.

8.1.2 Performance relative to the shape of the Pareto front

The shape of the Pareto front is often said to be an important factor in algorithm performance. So much so that that all the test suites that make up the test suite

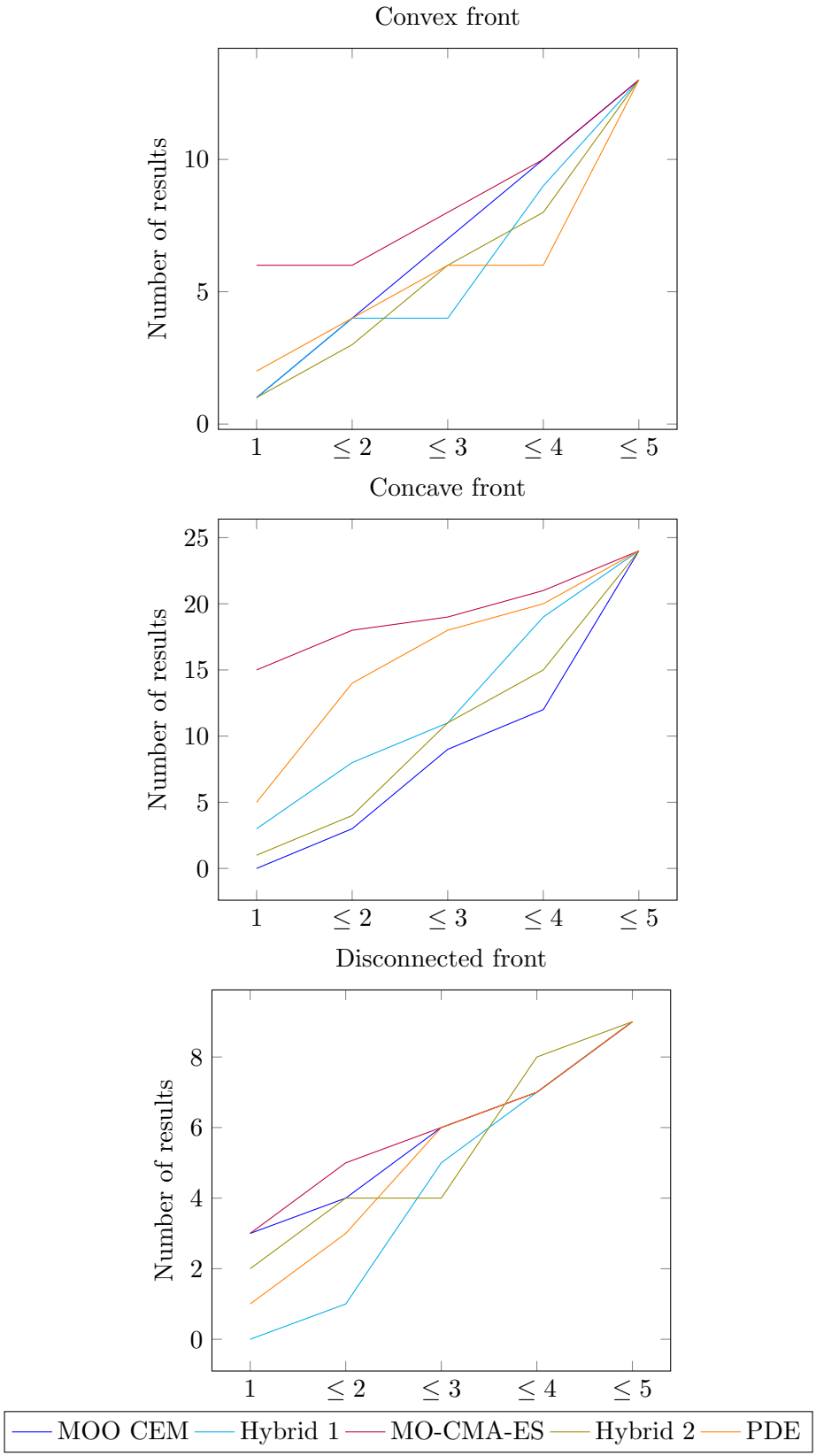


Figure 8.3: Algorithm performance for convex, concave and disconnected Pareto fronts.

for this study were assembled to contain at least one convex, one concave and one disconnected problem.

Figure 8.3 shows the results grouped according to the different shapes of the Pareto fronts. There are 13 problems with convex fronts, 24 with concave fronts and nine with disconnected fronts. Some of the problems have been classified as having both convex and disconnected fronts. These problems are only included in the disconnected front set.

Despite the focus placed on the importance of the shape of the Pareto front, the relative performance of the algorithms does not vary much for the different shapes. The MO-CMA-ES performs the best for all three shapes. The PDE performs second best for concave shapes, while the MOO CEM places second for convex and disconnected fronts.

The PDE seems to perform better when the fronts are concave than when they are convex. The MOO CEM performs worse on concave fronts than on convex or disconnected fronts.

For the algorithms investigated, the shape of the Pareto front does not affect the relative performance of the algorithms as clearly as the number of variables does. The MO-CMA-ES would be the best algorithm to use for all shapes.

8.1.3 Performance relative to the modality of the objective functions

Similar to the shape of a Pareto front, the modality of the objective functions is considered to be an important factor in algorithm performance. Figure 8.4 shows the results organised according to the modality of the objective functions.

A problem is classified to be unimodal if both its objective functions are unimodal. There are 21 such problems in the test suite. If at least one of the objective functions is multimodal, a problem is classified as being multimodal. There are 17 such problems. A problem is classified as deceptive if at least one of the objective functions is deceptive. Several problems are classified as both multimodal and deceptive. These problems were grouped in the deceptive objective function set. This set comprises eight problems.

There is no clear difference between the relative performance of algorithms as the modality of the objective functions changes. The MO-CMA-ES was the best performer for all three types of modality, followed by the PDE in all three modes.

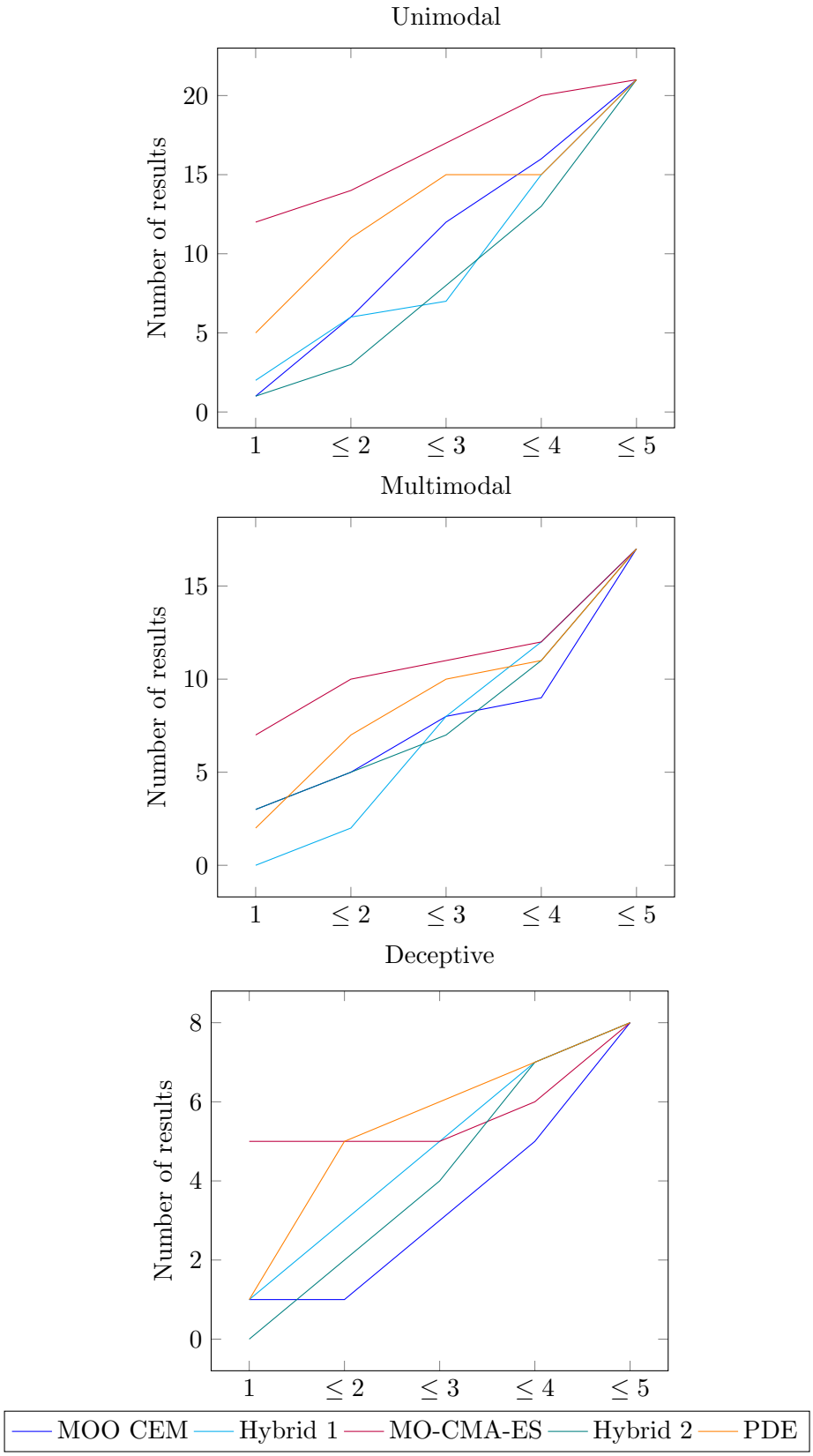


Figure 8.4: Algorithm performance for unimodal, multimodal and deceptive objective functions.

The nature of this analysis does not allow assumptions to be made about the absolute performance of algorithms. However, in terms of the relative performance of algorithms, the modality of objective functions does not have a clear effect. An algorithm that outperforms another algorithm for one type of modality would likely do so for another, and vice versa.

The MO-CMA-ES would be recommended irrespective of the modality of a problem.

8.1.4 Performance relative to the presence of reported relationships between decision variables

The focus of Chapter 4 was to find a balance between problems that were considered to have some form of relationship between decision variables and problems that were considered to have independent decision variables. There are 21 problems without noted relationships and 25 problems with reported relationships. The results are broken down accordingly in Figure 8.5.

The MOO CEM performed better on problems with reported relationships than on problems without reported relationships. Conversely, PDE performed better on problems without reported relationships than on problems with reported relationships.

The MO-CMA-ES was the best relative performer irrespective of the presence of reported relationships of the problems. PDE performed second best in cases with no reported relationships.

The effect of the presence of relationships between decision variables on algorithm performance was not as expected. The relative performance of the algorithms does not change drastically if relationships are present. Also, the noticeable changes that do exist are contrary to what was expected: the MOO CEM (which treats variables as independent) performs better on problems with reported relationships than on problems without, whereas the PDE (which is supposed to be well suited to solving problems with reported relationships) performs better on problems without reported relationships than on problems with reported relationships.

The MO-CMA-ES would be the best algorithm to select, irrespective of the presence or absence of reported relationships.

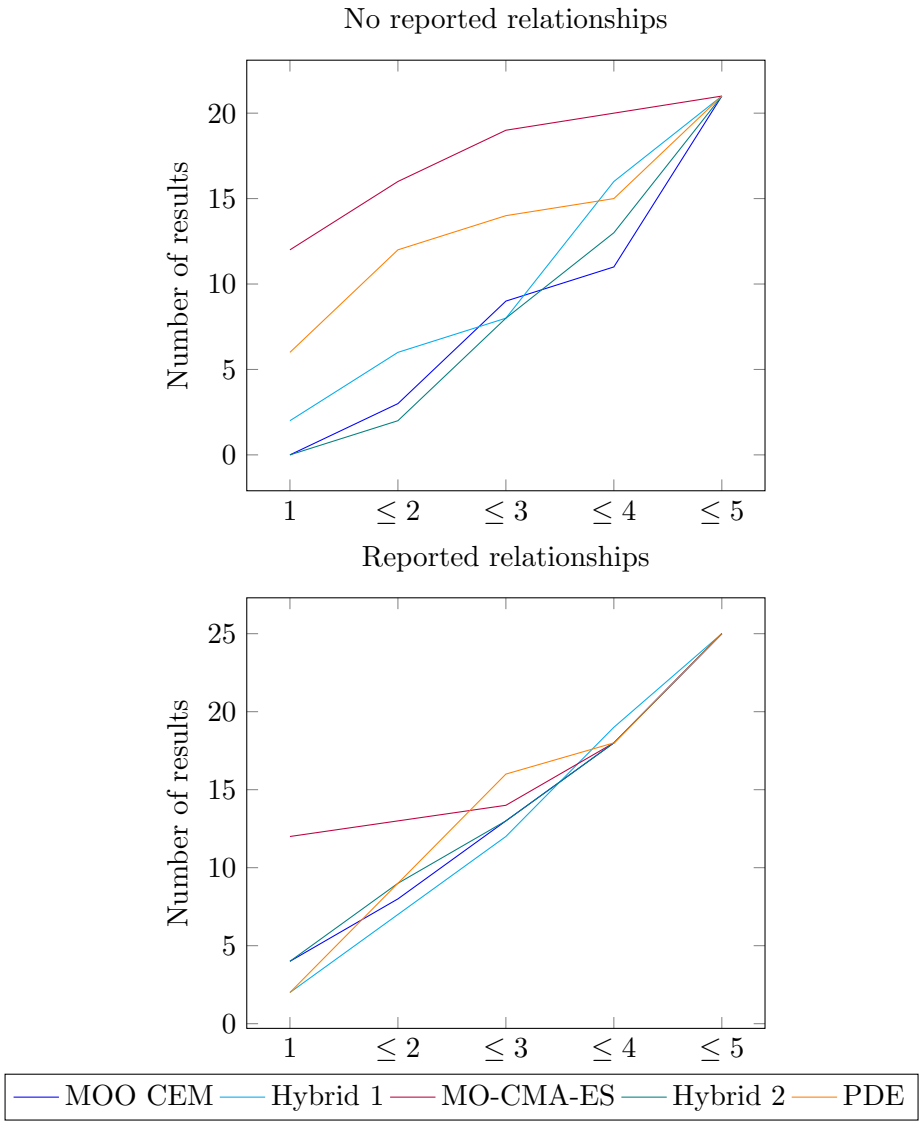


Figure 8.5: Algorithm performance for problems with and without reported relationships between decision variables.

8.1.5 General remarks on the results for unconstrained continuous problems

Of the four characteristics analysed, the number of variables a problem has, has the most marked effect on algorithm performance. For few to a medium number of variables, the MO-CMA-ES would be recommended. However, for many variable problems, the MOO CEM would be best.

The MOPs, ZDT, L_1ZDT and R problems all have too few variables to reveal the difference in performance of the algorithms as the number of variables increases. The WFG problems will only show this difference if the number of variables is large.

A very interesting result is that even when considering the presence of relationships between decision variables, the same algorithm (MO-CMA-ES) would be recommended for selection irrespective of the presence or absence of reported relationships.

The MO-CMA-ES would also be recommended irrespective of the shape of the Pareto front of a problem and irrespective of the modality of a problem.

8.2 Summary and discussion of MRR results

Six mission-ready resource (MRR) cases were set up. For all six cases, the number of variables was equal to 15. The size of the decision space was different for each case. This was achieved by adjusting the total number of tasks that required doing and the number of each MRR type that was available. Three cases were investigated where the number of each MRR type available could be a bounding constraint. For the other three cases, enough of each MRR type was available so that all tasks could be performed using a single MRR type. A summary of the results for the MRR cases is shown in Table 8.2. A value of 1 indicates that an algorithm significantly outperformed all four the other algorithms for that problem. A value of 5 indicates that an algorithm did not significantly outperform any of the other algorithms. Please refer to Appendix B for more detailed results.

The MOO CEM outperformed the other algorithms on all six test cases. Hybrid 1 generally performed well for all the cases, outperforming three algorithms on all the test cases. The relative performance of the MO-CMA-ES deteriorated as the size of the decision space increased. For the smaller test cases (Case A and Case B), the MO-CMA-ES outperformed two algorithms (PDE and Hybrid 1). However, on the larger test cases (Case C, Case D, Case E and Case F), the MO-CMA-ES was outperformed by all the other algorithms. Relatively, Hybrid 2 and the PDE performed the worst.

Similar to the continuous cases discussed above, the deteriorating performance of the MO-CMA-ES can likely be ascribed to its inability to update the covariance matrix accurately as the size of the decision space increases. Likewise, the good performance of the MOO CEM is probably due to the fact that it does not rely on successfully keeping

Table 8.2: Summary of relative performance ranks of algorithms on the MRR cases.

Problem	Total tasks	MRRs available	MOO CEM	Hybrid 1	MO-CMA-ES	Hybrid 2	PDE
Case A	16	5×5	1	2	3	5	4
Case B	16	5×20	1	2	3	5	4
Case C	150	5×50	1	2	5	3	4
Case D	150	5×200	1	2	5	4	4
Case E	300	5×100	1	2	5	4	4
Case F	300	5×400	1	2	5	4	4

track of relationships between decision variables and therefore is not hampered by an inability to do so for large decision spaces.

8.3 The quality of the Pareto fronts found for the MRR cases

Even though sensitivity analysis on the effect of changing the population size and maximum number of evaluations falls outside the scope of this study, it should be noted that the quality of the Pareto fronts achieved for the MRR cases can be improved by increasing population size and the maximum number of evaluations allowed. Figure 8.6 shows the original fronts achieved for Case D.

Figure 8.7 shows a set of fronts achieved with the population size equal to 300 and the maximum number of evaluations equal to 75 000. Even though the settings used in Figure 8.7 are by no means optimal, it is easy to see that an increased population size and number of evaluations has a positive effect on the quality of the fronts.

8.4 Summary and discussion of BAP results

The researcher experimented with three instances of the BAP problem. Two of the instances are relatively small, with the search spaces comprising 4 080 feasible solutions.

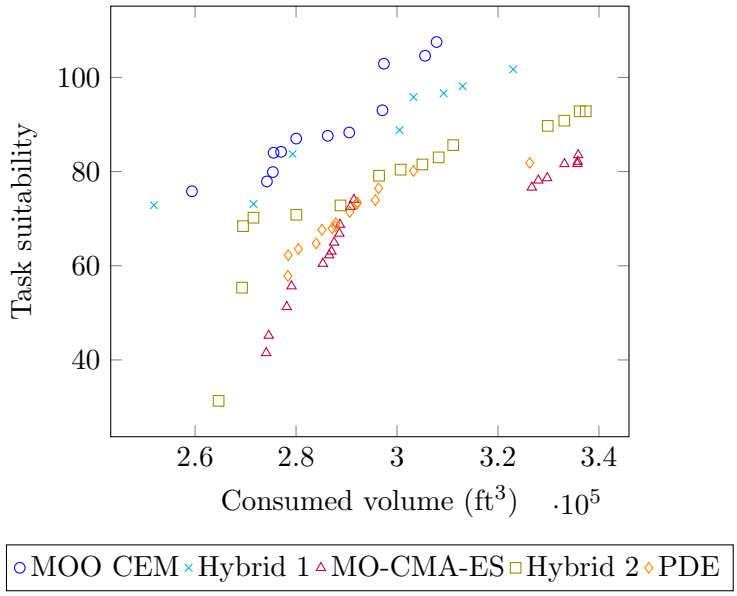


Figure 8.6: A sample of Pareto fronts achieved by the algorithms on MRR Case D

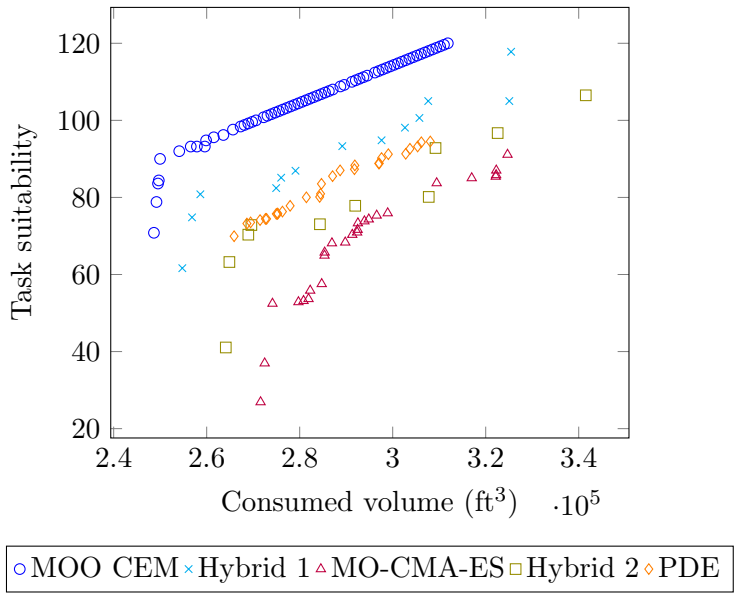


Figure 8.7: A sample of Pareto fronts achieved by the algorithms on MRR Case D with the population size equal to 300 and the maximum number of evaluations equal to 75 000.

Table 8.3: Summary of relative performance ranks of algorithms on the BAP cases.

Problem	MOO CEM	Hybrid 1	MO-CMA-ES	Hybrid 2	PDE
Case A	5	4	2	1	4
Case B	5	4	3	1	4
Case C	1	3	4	3	5

The third case is much bigger in relation, with a search space comprising 1×10^{16} feasible solutions. The relative performances of the algorithms on each of these cases are summarised in Table 8.3. A value of 1 indicates that an algorithm significantly outperformed all four the other algorithms for that problem. A value of 5 indicates that an algorithm did not significantly outperform any of the other algorithms.

Box plots, sample Pareto fronts and the Mann-Whitney U-test results for the BAP cases can be found in Appendix D.

Hybrid 2 performed the best when compared to the other algorithms on the two small cases, while the MOO CEM performed the best in comparison to the other algorithms on the large case.

The MOO CEM performed the poorest relative to the other algorithms when solving the two small cases.

Similar to the results for the continuous and MRR cases, the relative performance of the MO-CMA-ES declined as the size of the search space increased. Similar to the results of the continuous cases, the relative performance of the MOO CEM, and the closely related Hybrid 1, improved as the size of the search space increased.

In general, Hybrid 2 was the best performing algorithm on the BAP cases, while the PDE was the worst.

8.5 Conclusion: Analysis of experimental results

This chapter presented an analysis and discussion of the experimental results.

The next chapter is the final chapter and presents a summary of the research done, primary findings, and recommendations for possible future research projects.

Chapter 9

Summary and conclusions

The previous chapter presented an analysis of the experimental results.

This chapter presents a summary of the research done along with the primary findings. Recommendations for similar research are presented, along with recommended areas for future work. The chapter concludes with a summary of the skills acquired and the lessons learnt by the researcher.

9.1 Summary of research done

The purpose of this project was to investigate the effect of accounting for possible relationships between decision variables when solving multi-objective problems. This was done by comparing the performance of five multi-objective algorithms on a variety of problems.

The five optimisation algorithms included one algorithm known to work on an assumption that variables are independent, two algorithms reported to be able to handle relationships between variables, and two hybrid algorithms created by the researcher to answer some questions she had during implementation. The five algorithms are described in detail in Chapter 3. Four of the algorithms (the cross-entropy method for multi-objective optimisation (MOO CEM) being the exception) were implemented in Matlab by the researcher. An implementation of MOO CEM in Matlab was readily available to the researcher.

The test suite was made up of 46 unconstrained continuous cases, six static, combinatorial cases and three dynamic, stochastic, combinatorial cases. The unconstrained

problems are discussed in Chapter 4; details about the mission-ready resource problem (MRR) – the static, combinatorial case – can be found in Chapter 5, and the buffer allocation problem (BAP) – the dynamic problem – is described in Chapter 6. The researcher implemented 38 of the continuous problems and all the combinatorial test cases. She had access to eight already-implemented continuous problems, and to the simulation model used for the dynamic cases.

Performance was measured using the hypervolume indicator and the Mann-Whitney U-test. All performance results are relative to the performance of the other algorithms. Conclusions are not drawn about the absolute performance of algorithms, but rather about the relative performance of an algorithm when compared to the remaining algorithms.

Population size and algorithm-specific parameters were kept constant throughout. The maximum number of evaluations was also kept constant at 10 000, with the exception of the MRR cases. For these cases, the maximum number of evaluations was set to be 15 000.

9.2 Important findings

The most important findings of this project will be discussed now.

> **Accounting for relationships between decision variables is beneficial as long as relationships can be efficiently tracked.**

For small to medium-sized problems, with the population sizes and number of evaluations allowed, Pareto differential evolution (PDE) and the multi-objective covariance matrix adaptation evolution strategy (MO-CMA-ES), but especially the MO-CMA-ES, tended to outperform the other algorithms.

> **If it becomes too difficult to effectively keep track of relationships between decision variables, an algorithm that assumes independence is preferable.**

For very large problems, Hybrid 1 and the MOO CEM typically performed the best. Problems were considered to be large if they had a hundred or more variables in the

continuous cases. For the combinatorial cases, it is difficult to define a large problem. MRR Cases 2 to 6 were very large, and BAP Case 3 was the biggest of the BAP cases.

The superior performance of Hybrid 1 and the MOO CEM is attributed to the fact that the MO-CMA-ES is unable to effectively update its covariance matrix.

It has to be noted that the MO-CMA-ES might fare better if larger population sizes in conjunction with larger numbers of maximum evaluations were to be allowed. However, even if this were the case, the result is still useful: if computational resources are limited, and the problem to be solved is large, Hybrid 1 or the MOO CEM should be preferred to the otherwise superior MO-CMA-ES.

The size of a problem is the best indicator of algorithm performance.

Despite a lot of focus falling on other problem characteristics, this study shows that the size of a problem is a good indicator of relative algorithm performance. For very small problems, some algorithms might perform better than others, but the differences in performance are not that big. For small to medium-sized problems, the MO-CMA-ES performs very well. For large problems, Hybrid 1 or the MOO CEM is recommended.

Using just one test suite from literature does not tell one much.

Very often, only one test suite is used in a study, and researchers draw very optimistic conclusions about the performance of an algorithm they have developed based on the results achieved on this suite. This study used a variety of test suites. The results and conclusions of this study would have differed drastically if only one of the suites was used. For example, the MOPs, ZDT problems, L_1ZDT problems and R problems do not include problems with enough variables to have highlighted the difference in performance of the MO-CMA-ES or the MOO CEM as the number of variables increases. Had this study used only the MOPs or ZDT problems or L_1ZDT problems, the MO-CMA-ES would probably have been recommended without reservation.

9.3 Recommendations

After implementing five algorithms to work on 46 problems, the following recommendations are made:

> **Recommendation 1** – Understanding the fundamental strengths and weaknesses of an optimisation algorithm is of the utmost importance. Even though it is possible to adapt algorithms to specific problems, all algorithms have some inherent limitations, which can be avoided by choosing an appropriate optimisation algorithm.

The first example that comes to mind is the difference in performance between Hybrid 1 and the MOO CEM on the one hand, and the MO-CMA-ES on the other as the size of a problem increases. For small problems, the MO-CMA-ES typically outperforms the MOO CEM. However, when it comes to solving large problems, Hybrid 1 and the MOO CEM are superior.

> **Recommendation 2** – The more knowledge one has about the problem one is trying to solve, the better.

Problem knowledge enables one to:

1. Choose an appropriate algorithm to solve the problem.
2. Incorporate problem knowledge where necessary. The mission-ready resource (MRR) problem is a good example. Incorporating some problem-specific information regarding the constraints into the process makes it possible to find near-optimal solutions. Without this information, finding even feasible solutions is difficult.

> **Recommendation 3** – It is important to compare algorithms on a wide variety of problems.

Very often, researchers compare only a few algorithms on a small set of test problems and draw conclusions about algorithm performance based on these problems. This study shows that it becomes more and more difficult to draw definite conclusions about algorithm performance when algorithms are compared on a larger test suite. Even if an algorithm still outperforms other algorithms on average, exceptions to good or bad performance are more likely to be revealed when using a larger test suite.

9.4 Contribution to the research field

This project compared the recently developed MOO CEM with two existing algorithms to which it has not been compared before, namely the MO-CMA-ES and PDE algorithms. This comparison was done using 46 continuous problems, six cases of the combinatorial MRR problem and three cases of a dynamic, stochastic BAP.

The MOO CEM has previously been used to solve eight of these problems (the 5 MOPs and ZDT1 – ZDT3). For the other 38 problems, this is the first application of the MOO CEM to the problems.

For the extension of the covariance matrix adaptation evolution strategy (CMA-ES) to the MO-CMA-ES, Igel *et al.* (2007a) used five of the problems used here (the five ZDT problems) and some problems they developed specifically for their comparison. It seems that this study is the first official application of the MO-CMA-ES to the remaining 41 continuous problems and the combinatorial cases investigated.

Abbass *et al.* (2001) originally compared the performance of the PDE algorithm to that of the strength Pareto evolutionary algorithm (SPEA) on only two problems (two of the ZDT problems). This study is possibly the first official application of PDE to the other 44 continuous problems. It is the first application of the PDE algorithm to the two combinatorial problems investigated.

Two new hybrid algorithms (Hybrid 1 and Hybrid 2) were proposed and their performance was investigated.

In order to handle constraints for the MRR in the same fashion for all the algorithms, an algorithm for fixing infeasible solutions is suggested. This algorithm could be applied to other resource assignment problems for similar research.

The results of this study could aid future researchers and practitioners in selecting appropriate multi-objective optimisation algorithms for the problems they aim to solve.

9.5 Suggested future research

Possible future research related to this work includes:

- Despite the large test suite employed for this study, the performance of these algorithms on other problems can be investigated.
  - No constrained continuous problems were investigated.

  - One static combinatorial problem was investigated. Performance comparisons on more of these problems using these algorithms would be interesting.
  - One dynamic combinatorial problem was investigated. Performance comparisons on more of these types of problems would be interesting.

- More algorithms can be included in the comparison.
- Future work is required to define the concepts of "small", "medium", and "large" problems with respect to algorithm performance. This is true for all types of problems. For example, what is a "large" continuous problem? And a "large" combinatorial problem?
- Very little attention was paid to the effects of algorithm-specific parameters, population size and the number of generations allowed. The effects of changes to these parameters should be investigated.
- The combination of more specific definitions of problem size and changes to parameters would also be interesting.
- The method used to handle constraints could be applied to other resource assignment problems and adapted for other combinatorial problems.

9.6 Skills acquired

The bulk of the work done focused on the combined field of multi-objective optimisation and metaheuristics. The researcher has a much deeper understanding of the intricacies of both these fields than she had at the outset of this project.

In order to complete this study, the researcher had to master three existing multi-objective optimisation algorithms – the MOO CEM, MO-CMA-ES and PDE – and implement two of these – the MO-CMA-ES and PDE. In order to do this, she had to understand the workings of their single-objective counterparts as well as the complexities of extending a single-objective algorithm for multi-objective optimisation.

She had to implement 38 continuous problem cases, and the MRR test cases.

All the algorithms had to be applied to these problems. This was simple for the continuous problems, but finding a way of solving the MRR cases was far more complicated.

In order to optimise the BAP cases, the researcher had to find a way of integrating Matlab with Simio as changes to one of the two programs had rendered a previous student's guidelines for this process obsolete. Eventually, C# (a language completely new to the researcher) was used to bridge the gap.

9.7 Lessons learnt

The main lessons the researcher learnt were the following:

- There is a lot of detail to take into account when five algorithms have to be applied to 46 problems. Even small changes have different effects on each algorithm and keeping track of these effects becomes very difficult.
- At the outset of this project, the researcher saw it as a "black-and-white" study: she would be able to calculate hypervolume indicator values for each algorithm on each problem and these calculated values would then be the basis of her conclusions. She soon realised that it was not a "black-and-white" project at all. For every result she got, she had to make countless decisions to get there; each decision making the result true for a more specific case. For example, all continuous results are true for a population size equal to 100, for a maximum number of evaluations equal to 10 000, and for the algorithm parameters she decided on. Different combinations of these might very well result in different outcomes.
- In the field of finding near-optimal solutions, there are no absolutely correct answers; there might be better and worse answers, but the "correct" answers are those that satisfy the stakeholders. This is a difficult thing to come to terms with if the original appeal of the project was its apparent "black-and-white"-ness. The researcher sees an analogy with life in this lack of "correct" answers. Even though she would like life to have simple "correct" answers, she is relatively sure that most of the decisions we face do not have "correct" answers, but, at best, only near-optimal answers that we are willing to live with.

9.8 Conclusion: Summary and conclusions

This chapter presented a summary of the research done and the primary findings of the study. In addition, recommendations for similar research were made based on the primary findings, and future research work was recommended. The chapter ends on a personal note with a summary of the skills acquired and lessons learnt by the researcher.

References

Abbass, H.A., Sarker, R. & Newton, C. (2001). PDE: A Pareto-frontier differential evolution approach for multi-objective optimization problems. In *Proceedings of the 2001 Congress on Evolutionary Computation*. 34, 36, 37, 38, 118

Babu, B.V. & Jehan, M.M.L. (2003). Differential evolution for multi-objective optimisation. In *The 2003 Congress on Evolutionary Computation*. 34, 36

Bekker, J. (2012). *Applying the cross-entropy method in multi-objective optimisation of dynamic, stochastic systems.*. Ph.D. thesis, Department of Industrial Engineering, University of Stellenbosch. iv, v, 1, 3, 4, 17, 21, 48, 84, 85, 86, 90

Bekker, J. & Aldrich, C. (2010). The cross-entropy method in multi-objective optimisation: An assessment. *European Journal of Operational Research*, **211**, 112–121. iv, v, 3, 17, 21, 23

Boussaïd, I., Lepagnot, J. & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, **237**, 82–117. 10, 11, 25, 34

Coello Coello, C.A. (2006). Evolutionary multi-objective optimization: A historical view of the field. *IEEE Computational Intelligence Magazine*, **February**, 28–36. 6, 7, 8

Coello Coello, C.A., Lamont, G.B. & Van Veldhuizen, D.A. (2007). *Evolutionary Algorithms for Solving Multi-objective Problems*. Genetic and Evolutionary Computation Series, Springer. 77

Cover, T.M. & Thomas, J.A. (2006). *Elements of information theory*. Wiley-Interscience. 18

Cruz, F., Van Woensel, T. & Smith, J.M. (2010). Buffer and throughput trade-offs in M/G/1/K queueing networks: A bi-criteria approach. *International Journal of Production Economics*, **125**, 224–234. 84, 85

Das, I. (1999). A preference ordering among various Pareto optimal alternatives. *Structural Optimization*, **18**, 30–35. 10

De Jong, K.A. (1975). *Analysis of the behavior of a class of genetic adaptive systems*. Ph.D. thesis, The University of Michigan. 13

De Weck, O.L. (2004). Multiobjective optimization: History and promise. In *The Third China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems*, vol. 2, Kanazawa, Japan, invited keynote paper. 6, 7

Deb, K. (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, **7**, 205. 50

Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons. 77

Deb, K., Sinha, A. & Kukkonen, S. (2006). Multi-objective test problems, linkages, and evolutionary methodologies. In *GECCO '06 Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 1141–1148. 2, 48, 50

Di Pierro, F., Khu, S.T. & Savić, D.A. (2007). An investigation on preference order ranking scheme for multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, **11**, 17–45. 10, 79

Edgeworth, F.Y. (1881). *Mathematical psychics: an essay on the application of mathematics to the moral sciences*. C. Kegan Paul & Co. 7

Hansen, N. & Ostermeier, A. (1996). Adapting arbitary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation, 1996*. 25, 26, 27, 28, 29

Hansen, N. & Ostermeier, A. (1997). Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The $(\mu/\mu_I, \lambda)$-CMA-ES. In

EUFIT '97, 5th European Congress on Intelligent Techniques and Soft Computing. 27

Hansen, N. & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, **9**, 159–195. 27, 28, 30

Hansen, N., Ostermeier, A. & Gawelczyk, A. (1995). On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In L.J. Eshelman, ed., *Proceedings of the Sixth International Conference on Genetic Algorithms*, 57–64. 26, 27

Hansen, N., Müller, S.D. & Koumoutsakos, P. (2003). Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, **11**, 1–18. 27

Heiberger, R.M. & Holland, B. (2004). *Statistical analysis data display – An intermediate course with examples S-Plus, R, and SAS*. Springer. 93

Huband, S., Barone, L., While, L. & Hingston, P. (2005). A scalable multi-objective test problem toolkit. In *3rd International Conference on Evolutionary Multi-Criterion Optimization*, vol. 3410 of *Lecture Notes in Computer Science*, 280–294, Springer-Verlag. 48, 54

Huband, S., Hingston, P., Barone, L. & While, L. (2006). A review of multi-objective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, **10**, 477–506. 2, 3, 43, 44, 46, 47, 48, 71

Igel, C., Hansen, N. & Roth, S. (2007a). Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, **15**, 1–28. 25, 28, 31, 32, 33, 34, 118

Igel, C., Suttorp, T. & Hansen, N. (2007b). Steady-state selection and efficient covariance matrix update in the multi-objective CMA-ES. In *Evolutionary Multi-Criterion Optimization*. 34

Iorio, A.W. & Li, X. (2005). Solving rotated multi-objective optimization problems using differential evolution. *AI 2004: Advances in Artificial Intelligence*, **3339**, 861–872. 34, 52

Iorio, A.W. & Li, X. (2006). Rotated test problems for assessing the perfomance of multi-objective optimization algorithms. In *GECCO '06 Proceedings of the 8th annual conference on Genetic and evolutionary computation*, 683–690. 2, 48, 52

Knowles, J., Thiele, L. & Zitzler, E. (2005). A tutorial on the performance assessment of stochastic multiobjective optimizers. 88, 92

Lizárraga, G., Gomez, M.J., Castañon, M.G., Acevedo-Davila, J. & Rionda, S.B. (2009). Why unary quality indicators are not inferior to binary quality indicators. In *Proceedings of the 8th Mexican International Conference on Artificial Intelligence*. 89

Loshchilov, I., Schoenauer, M. & Sebag, M. (2011). Not all parents are equal for MO-CMA-ES. In *Evolutionary Multi-Criterion Optimization*. 34

Madavan, N.K. (2002). Multiobjective optimization using a Pareto differential evolution approach. In *Proceedings of the 2002 Congress on Evolutionary Computation*. 34, 36

McDonald, J.H. (2008). *Handbook of biological statistics*. Sparky House Publishing. 93

Nelder, J.A. & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, **7**, 308–313. 35

Ostermeier, A., Gawelczyk, A. & Hansen, N. (1994a). A derandomized approach to self adaptation of evolution strategies. *Evolutionary Computation*, **2**, 369–380. 25, 26

Ostermeier, A., Gawelczyk, A. & Hansen, N. (1994b). Step-size adaptation based on non-local use selection information. In Y. Davidor, H. Schwefel & R. Männer, eds., *Parallel Problem Solving from Nature – PPSN III, Proceedings*, 189–198. 26, 27, 28, 29

Rubinstein, R.Y. (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, **1**, 127–190. 17, 18, 19, 21

Rubinstein, R.Y. & Kroese, D.P. (2004). *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Springer. 18, 19, 21

Rubinstein, R.Y. & Kroese, D.P. (2008). *Simulation and the Monte Carlo Method*. Wiley. 18, 19, 20, 21

Rubinstein, R.Y. & Shapiro, A. (1990). Optimization of static simulation models by the score function method. *Mathematics and Computers in Simulation*, **32**, 373–392. 19

Stork, R. & Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimisation over continuous spaces. *Journal of Global Optimization*, **11**, 341–359. 35, 36, 37

Talbi, E.G. (2009). *Metaheuristics from design to implementation*. Wiley. 11, 12, 77

Van Veldhuizen, D.A. (1999). *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. Ph.D. thesis, Air Force Institute of Technology. 46, 48

Voss, T., Hansen, N. & Igel, C. (2010). Improved step size adaptation for the mo-cma-es. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. 34

Wakefield, D.J. (2001). *Indentification of preferred operational plan force mixes using a multiobjective methodology to optimize resouce suitability and lift cost*. Master's thesis, Air Force Institute of Technology. 72, 73, 74, 75, 78

Wolpert, D.H. & Macready, W.G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, **1**, 67–82. 2

Wood, J. (2012). Reading and writing CSV files in C#. 239

Xue, F., Sanderson, A.C. & Graves, R.J. (2003). Pareto-based multi-objective differential evolution. In *The 2003 Congress on Evolutionary Computation*. 34, 36

Zitzler, E. (1999). *Evolutionary algorithms for multiobjective optimization: Methods and applications*. Ph.D. thesis, Swiss Federal Institute of Technology Zurich. 5, 8, 9, 12, 13, 14, 75

Zitzler, E. & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms – a comparative case study. In *Parallel problem solving from nature – PPSN V*. 90

Zitzler, E. & Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, **3**, 257–271. 90

Zitzler, E., Deb, K. & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, **8**, 173–195. 48, 50

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. & da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, **7**, 117–132. 89, 90, 92

Zitzler, E., Brockhoff, D. & Thiele, L. (2007). The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In *Evolutionary Multi-criterion Optimization*. 92

Appendix A

Results for the continuous test problems

This appendix presents the results for the continuous test problems. For each problem, a summary of some important problem characteristics, along with a sample of the Pareto fronts achieved are presented. Each set of sample Pareto fronts is plotted with the true Pareto front for the problem and the reference point used to calculate the hypervolume. For improved visibility, the maximum value of the y-axis is sometimes limited, subsequently excluding points very far away from the true Pareto front from the plot.

For all the MOPs, ZDT problems, L_1ZDT problems, and R problems, two sets of box plots are also shown: a summary of the hypervolumes achieved and a summary of the relative run times of the algorithms.

For each of the WFG problems, three experiments were performed: one with the number of variables equal to four, the second with the number of variables equal to 20 and a third with 100 variables. The box plots for the each of the WFG problems show summaries of these three experiments.

Mann-Whitney U-tests were performed on the hypervolumes achieved. These results are presented for all the experiments. For the Mann-Whitney U-test results, if a matrix entry $ij = 1$, it indicates that Algorithm i achieved a significantly higher hypervolume than Algorithm j at a 5% significance level. The *Outperformed* column sums the total number of algorithms that Algorithm i outperformed, whereas the *Rank*

column indicates which algorithm performed best on the problem at hand (with 1 being the best and 5 being the worst).

A thousand replications of all the continuous test problem experiments were performed using a population size of 100, and a maximum number of evaluations equal to 10 000. All the continuous problems have two objectives.

A.1 MOP 1

Table A.1: Problem details for MOP 1.

Number of variables	1
Box constraints	$-10^5 \leq x \leq 10^5$
Geometry	Convex
Relationship	No reported relationships
Modality	Unimodal
Function definitions	Minimise both
	$f_1(x) = x^2$
	$f_2(x) = (x-2)^2$

Table A.2: Mann-Whitney U-test results for MOP 1.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	1	1	0	3	**2**
Hybrid 1	0	-	0	0	0	0	**5**
MO-CMA-ES	0	1	-	1	0	2	**3**
Hybrid 2	0	1	0	-	0	1	**4**
PDE	1	1	1	1	-	4	**1**

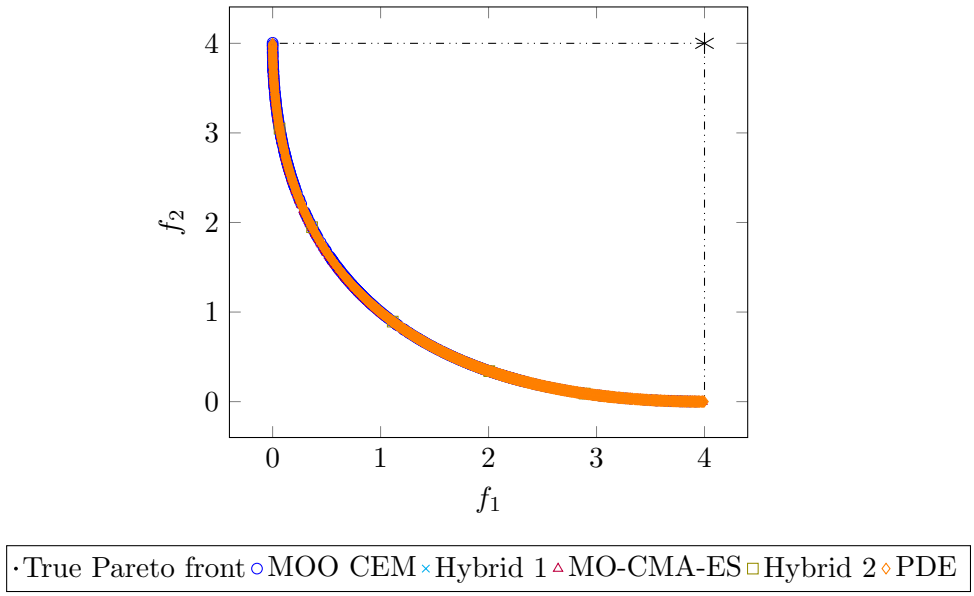


Figure A.1: A sample of Pareto fronts achieved by the algorithms on MOP 1.

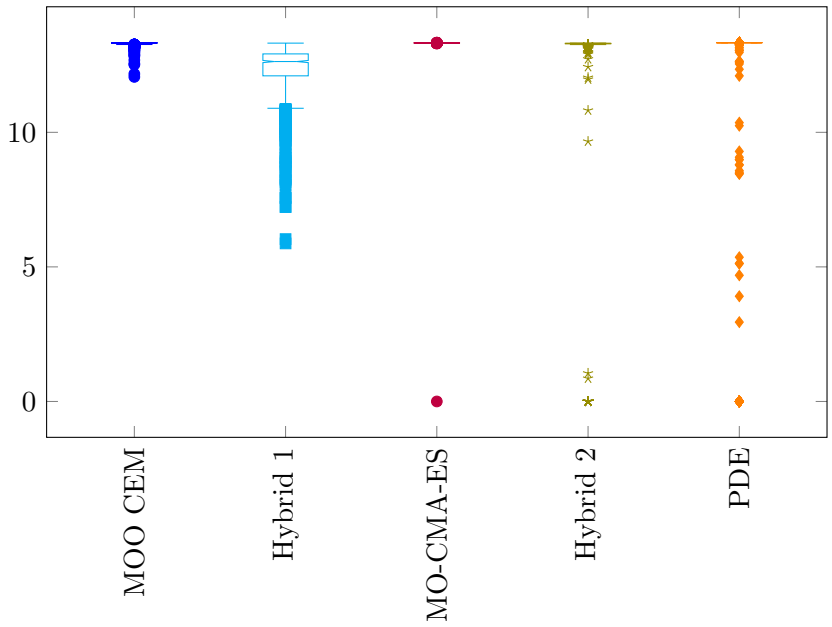


Figure A.2: Box plot of hypervolumes achieved when solving MOP 1.

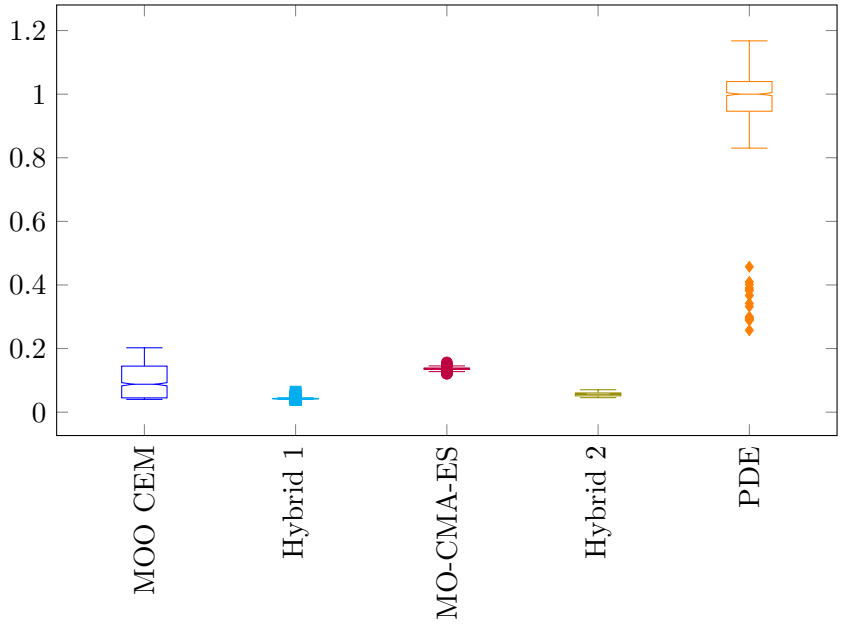


Figure A.3: Box plot of relative run times when solving MOP 1.

A.2 MOP 2

Table A.3: Problem details for MOP 2.

Number of variables	3
Box constraints	$-4 \leq x_i \leq 4, \quad i = 1, 2, 3$
Geometry	Concave
Relationship	No reported relationships
Modality	Unimodal
Function definitions	Minimise both
	$f_1(\mathbf{x}) = 1 - e^{-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}})^2}$
	$f_2(\mathbf{x}) = 1 - e^{-\sum_{i+1}^{3}(x_i - \frac{1}{\sqrt{3}})^2}$

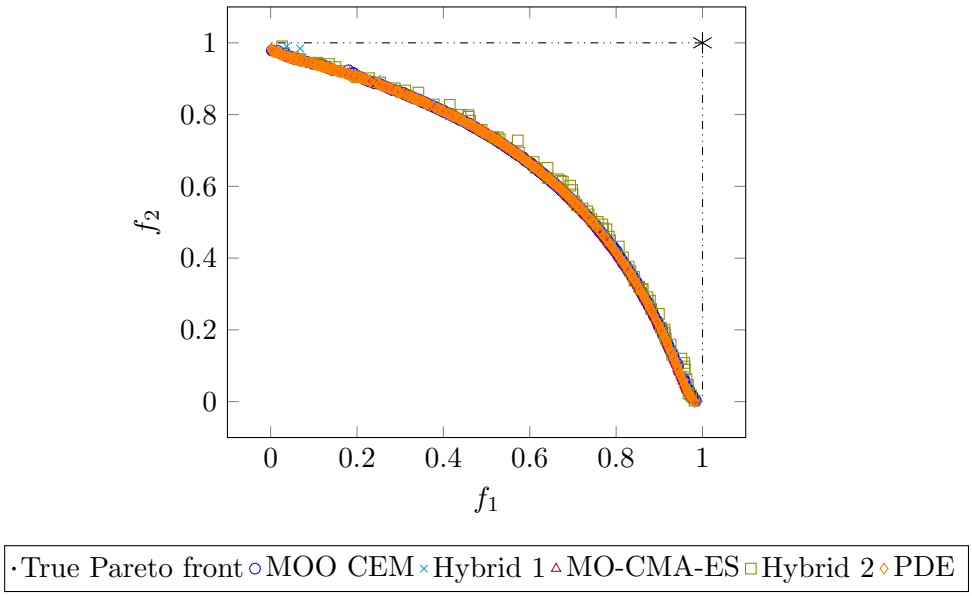


Figure A.4: A sample of Pareto fronts achieved by the algorithms on MOP 2.

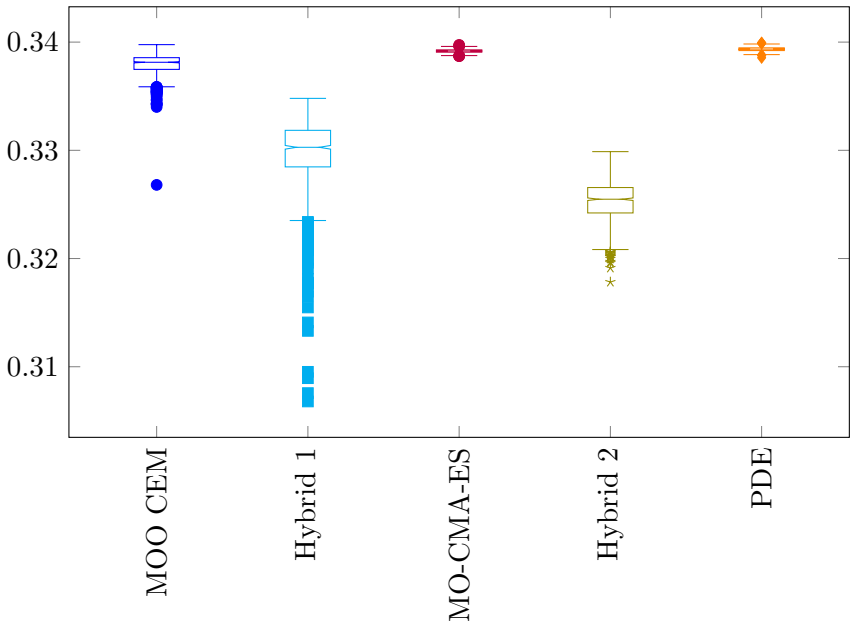


Figure A.5: Box plot of hypervolumes achieved when solving MOP 2.

Table A.4: Mann-Whitney U-test results for MOP 2.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	0	1	0	2	**3**
Hybrid 1	0	-	0	1	0	1	**4**
MO-CMA-ES	1	1	-	1	0	3	**2**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	1	1	1	1	-	4	**1**

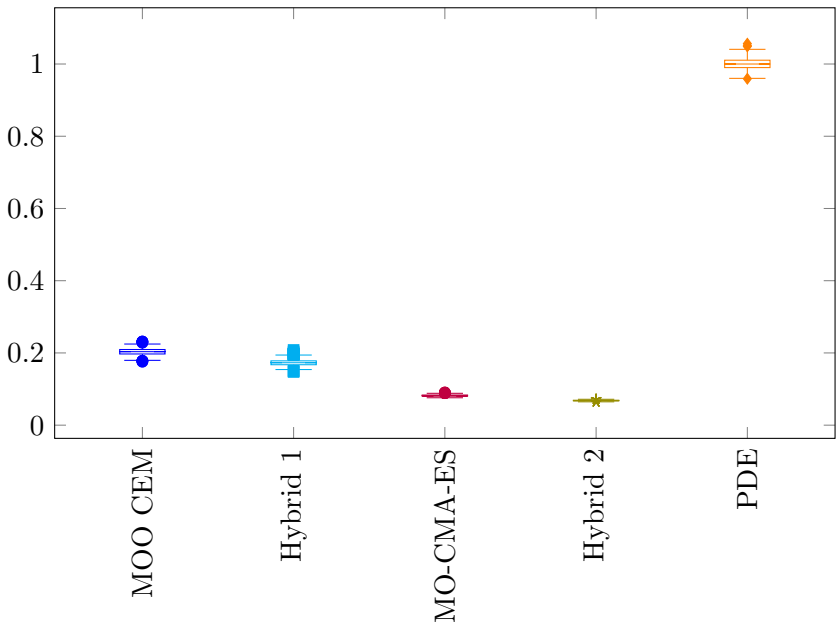


Figure A.6: Box plot of relative run times when solving MOP 2.

A.3 MOP 3

Table A.5: Problem details for MOP3.

Number of variables	2		
Box constraints	$-\pi \leq x_i \leq \pi, \quad i = 1, 2$		
Geometry	Disconnected		
Relationship	Reported relationships		
Modality	Multimodal		
Function definitions	Maximise both		
	$f_1(\mathbf{x})$	$=$	$-(1 + (A - B)^2 + (C - D)^2)$, where
	A	$=$	$0.5\sin(1) - 2\cos(1) + \sin(2) - 1.5\cos(2)$
	B	$=$	$0.5\sin(x_1) - 2\cos(x_1) + \sin(x_2) - 1.5\cos(x_2)$
	C	$=$	$1.5\sin(1) - \cos(1) + 2\sin(2) - 0.5\cos(2)$
	D	$=$	$1.5\sin(x_1) - \cos(x_1) + 2\sin(x_2) - 0.5\cos(x_2)$
	$f_2(\mathbf{x})$	$=$	$-((x_1 + 3)^2 + (x_2 + 1)^2)$

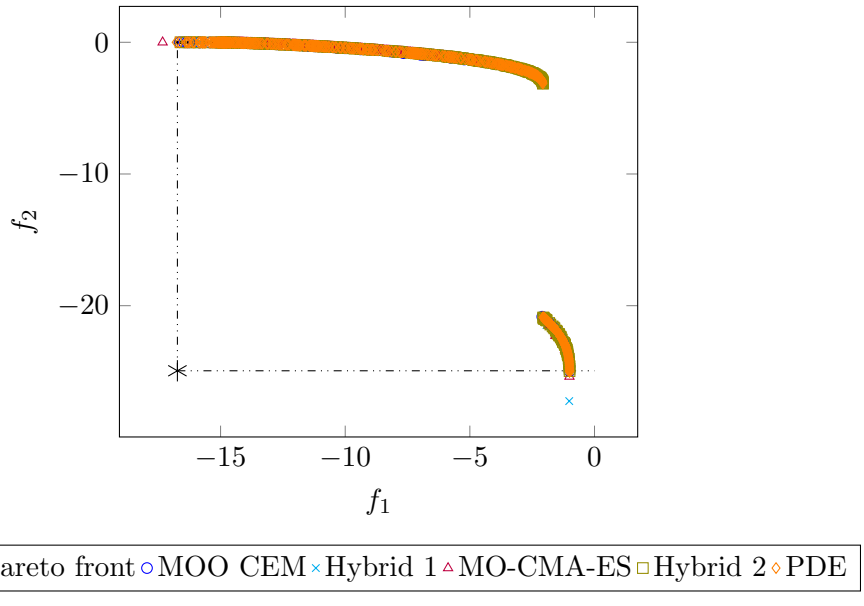


Figure A.7: A sample of Pareto fronts achieved by the algorithms on MOP 3.

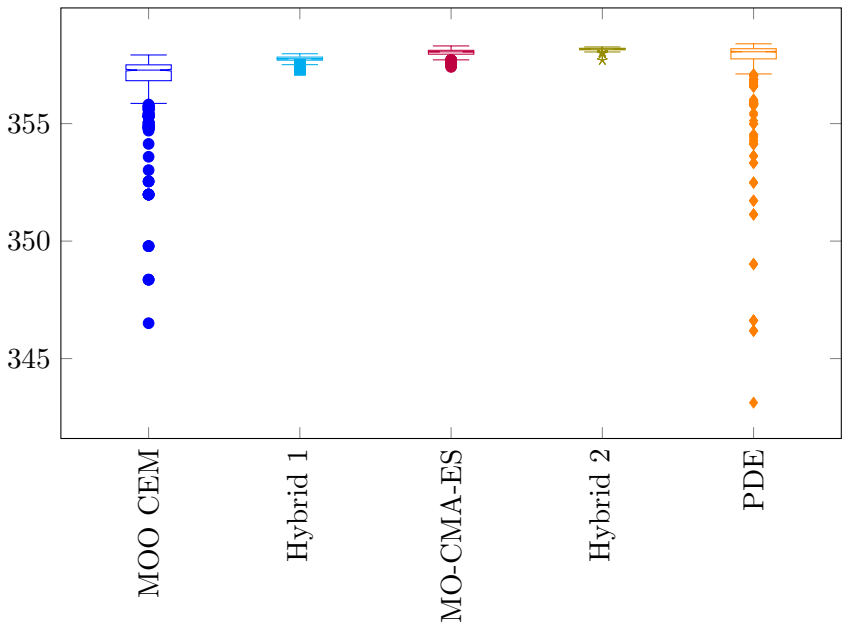


Figure A.8: Box plot of hypervolumes achieved when solving MOP 3.

Table A.6: Mann-Whitney U-test results for MOP 3.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	0	0	1	**4**
MO-CMA-ES	1	1	-	0	0	2	**3**
Hybrid 2	1	1	1	-	1	4	**1**
PDE	1	1	0	0	-	2	**3**

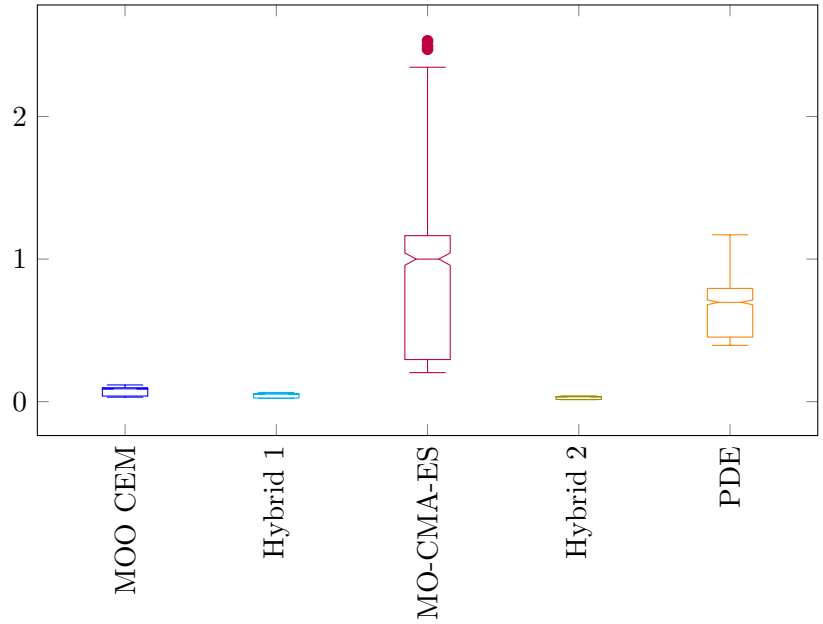


Figure A.9: Box plot of relative run times when solving MOP 3.

A.4 MOP 4

Table A.7: Problem details for MOP4.

Number of variables	3
Box constraints	$-5 \leq x_i \leq 5, \quad i = 1, 2, 3$
Geometry	Disconnected
Relationship	No reported relationships
Modality	Multimodal
Function definitions	Minimise both
	$f_1(\mathbf{x}) = \sum_{i=1}^{2}(-10e^{(-0.2\sqrt{x_i^2+x_{i+1}^2})})$
	$f_2(\mathbf{x}) = \sum_{i=1}^{3}(\lvert x_i\rvert^{0}.8 + 5\sin(x_i)^3)$

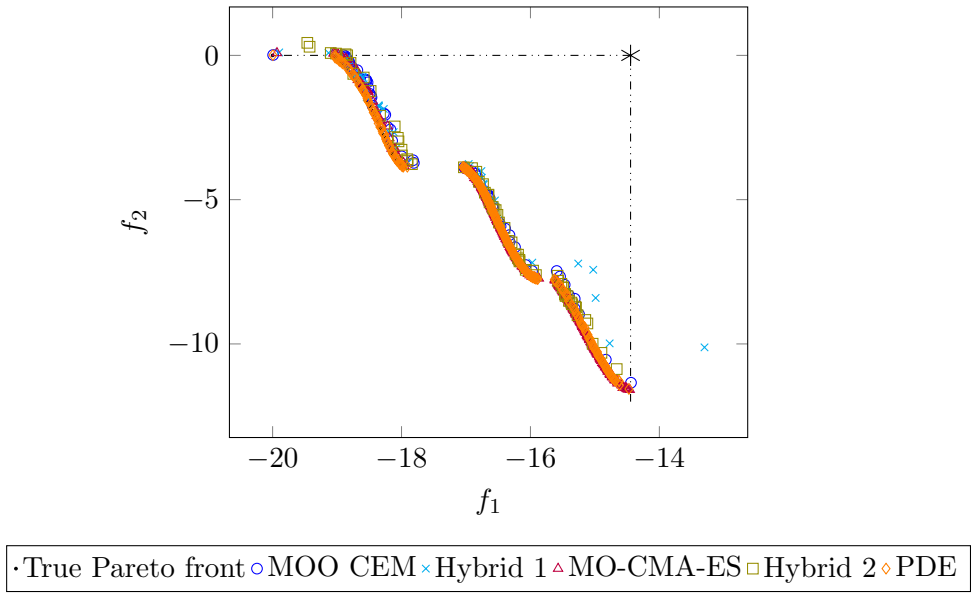


Figure A.10: A sample of Pareto fronts achieved by the algorithms on MOP 4.

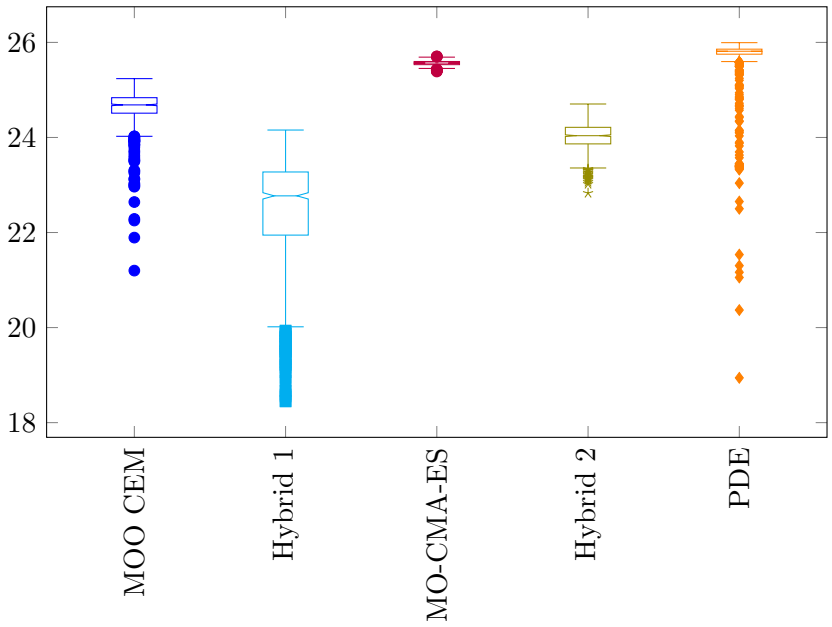


Figure A.11: Box plot of hypervolumes achieved when solving MOP 4.

Table A.8: Mann-Whitney U-test results for MOP 4.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	0	1	0	2	**3**
Hybrid 1	0	-	0	0	0	0	**5**
MO-CMA-ES	1	1	-	1	0	3	**2**
Hybrid 2	0	1	0	-	0	1	**4**
PDE	1	1	1	1	-	4	**1**

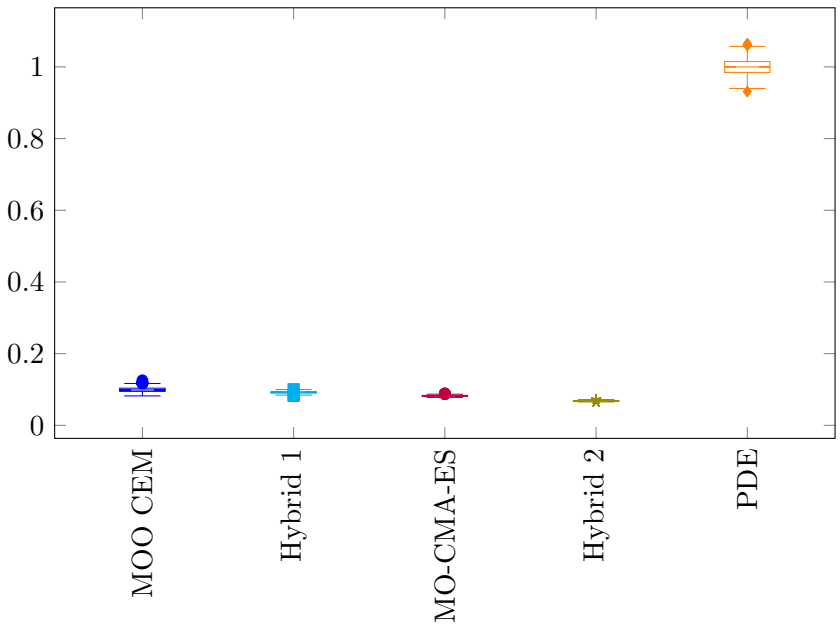


Figure A.12: Box plot of relative run times when solving MOP 4.

A.5 MOP 6

Table A.9: Problem details for MOP 6.

Number of variables		2
Box constraints		$0 \leq x_i \leq 1, \quad i = 1, 2$
Geometry		Disconnected
Relationship		No reported relationships
Modality		Multimodal
Function definitions	Minimise both	
	$f_1(x_1) = x_1$	
	$f_2(\mathbf{x}) = (1 + 10x_2) \times (1 - (\frac{x_1}{1+10x_2})^2 - \frac{x_1}{1+10x_2} \sin(12\pi x_1))$	

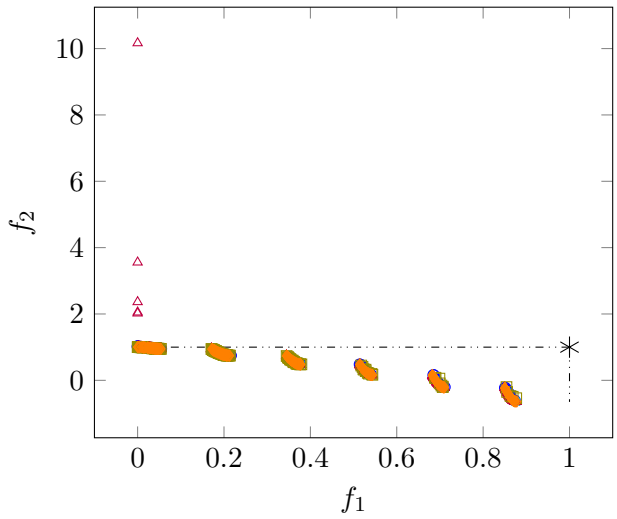


Figure A.13: A sample of Pareto fronts achieved by the algorithms on MOP 6.

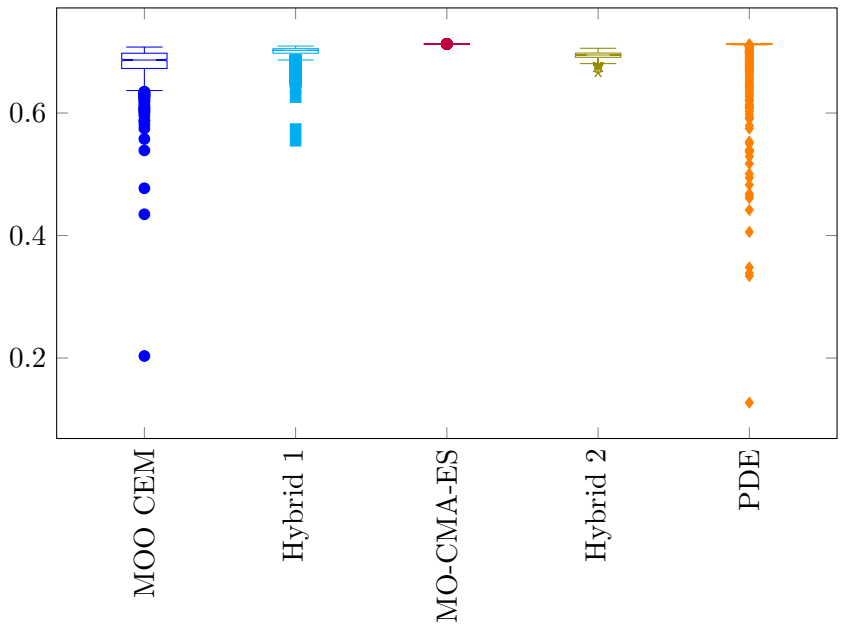


Figure A.14: Box plot of hypervolumes achieved when solving MOP 6.

Table A.10: Mann-Whitney U-test results for MOP 6.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	1	0	2	**3**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	1	0	0	-	0	1	**4**
PDE	1	1	0	1	-	3	**2**

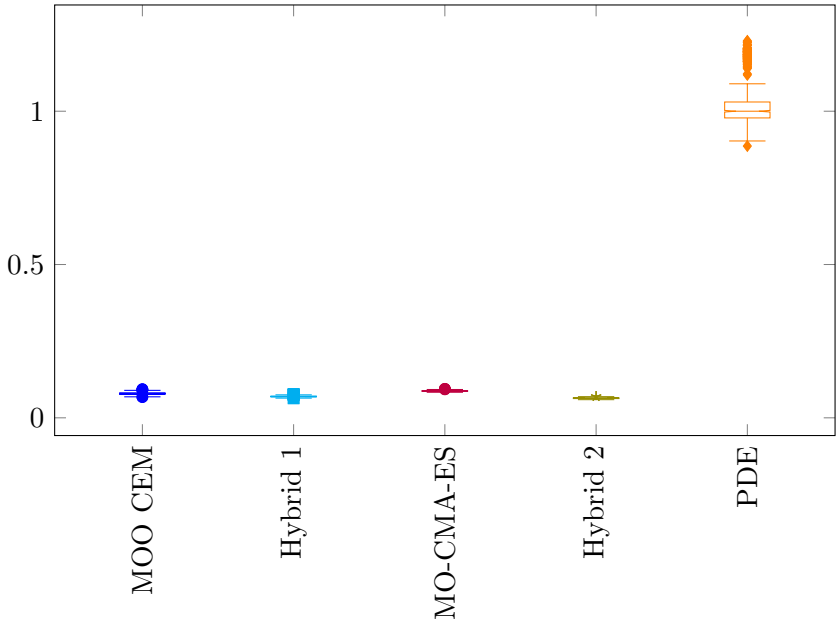


Figure A.15: Box plot of relative run times when solving MOP 6.

A.6 ZDT 1

Table A.11: Problem details for ZDT 1.

Number of variables	30
Box constraints	$0 \leq x_i \leq 1, \quad i = 1, \ldots, 30$
Geometry	Convex
Relationship	No reported relationships
Modality	Unimodal
Function definitions	Minimise both
	$f_1(x_1) = x_1$
	$g(\mathbf{x}) = 1 + 9\frac{\sum_{i=2}^{n} x_i}{n-1}$
	$f_2(\mathbf{x}) = 1 - \sqrt{\frac{f_1}{g}}$

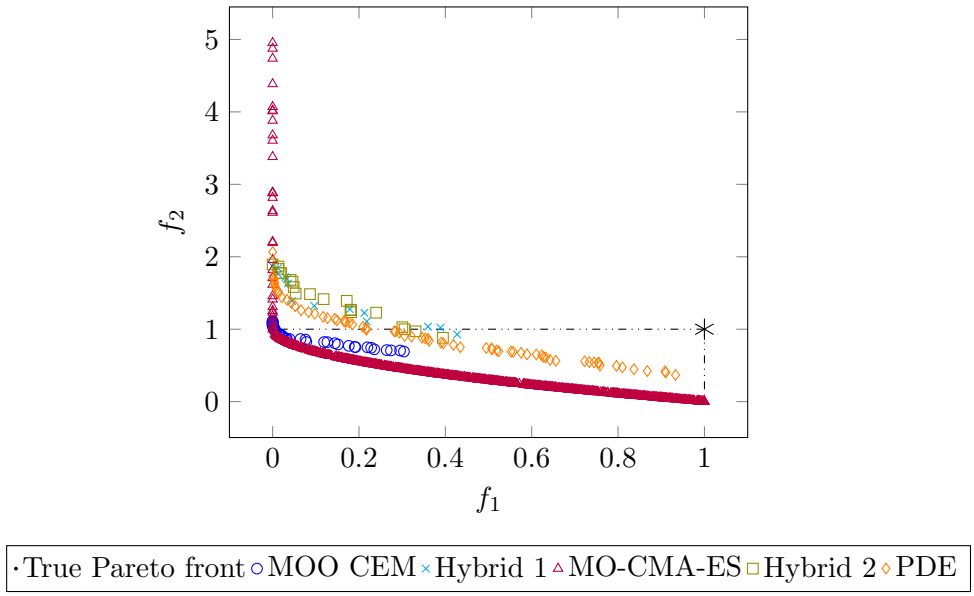


Figure A.16: A sample of Pareto fronts achieved by the algorithms on ZDT 1.

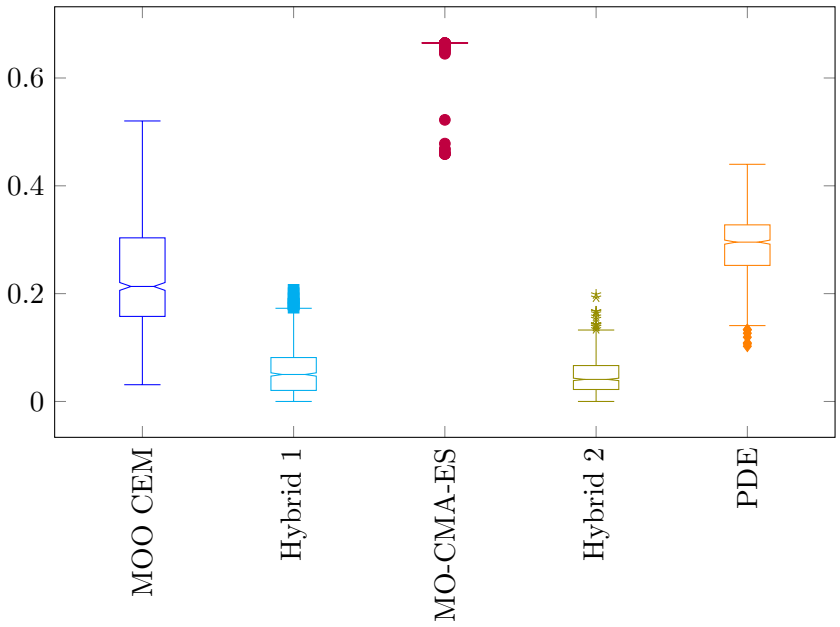


Figure A.17: Box plot of hypervolumes achieved when solving ZDT 1.

Table A.12: Mann-Whitney U-test results for ZDT 1.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	0	1	0	2	**3**
Hybrid 1	0	-	0	1	0	1	**4**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	1	1	0	1	-	3	**2**

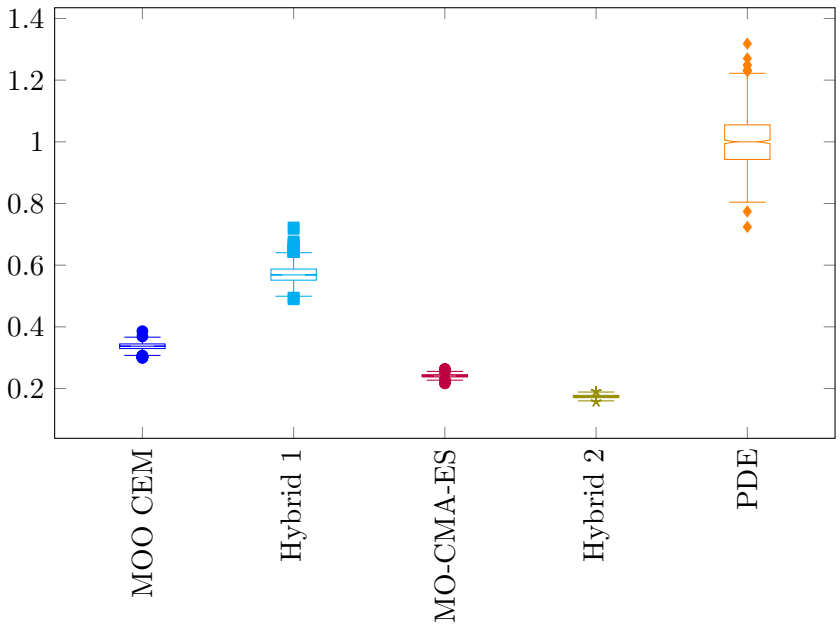


Figure A.18: Box plot of relative run times when solving ZDT 1.

A.7 ZDT 2

Table A.13: Problem details for ZDT 2.

Number of variables	30
Box constraints	$0 \leq x_i \leq 1, \quad i = 1, \ldots, 30$
Geometry	Concave
Relationship	No reported relationships
Modality	Unimodal
Function definitions	Minimise both
	$f_1(x_1) = x_1$
	$g(\mathbf{x}) = 1 + 9\frac{\sum_{i=2}^{n} x_i}{n-1}$
	$f_2(\mathbf{x}) = 1 - \sqrt{\frac{f_1}{g}}$

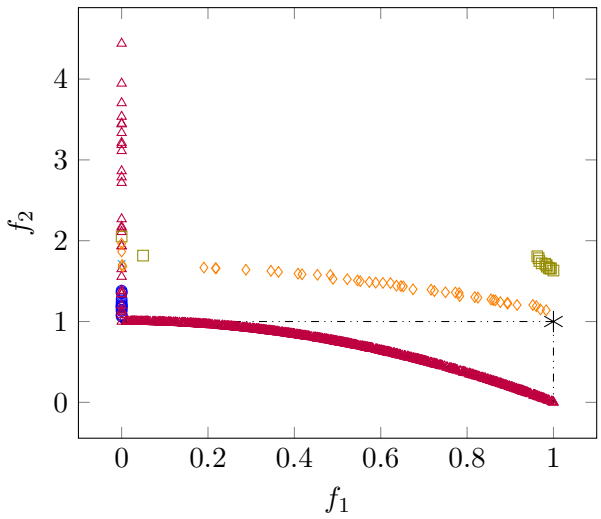


Figure A.19: A sample of Pareto fronts achieved by the algorithms on ZDT 2.

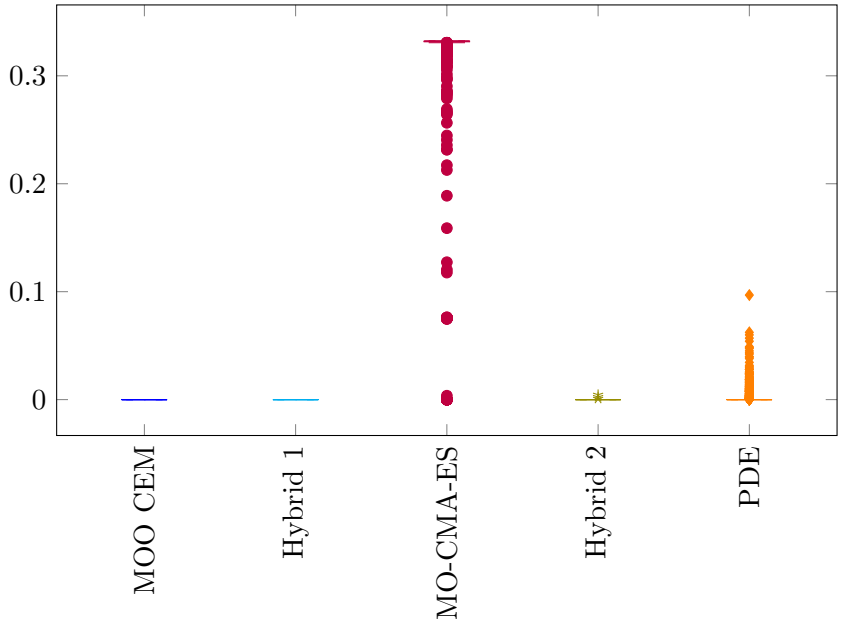


Figure A.20: Box plot of hypervolumes achieved when solving ZDT 2.

Table A.14: Mann-Whitney U-test results for ZDT 2.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	0	-	0	0	0	0	**5**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	1	1	0	-	0	2	**3**
PDE	1	1	0	1	-	3	**2**

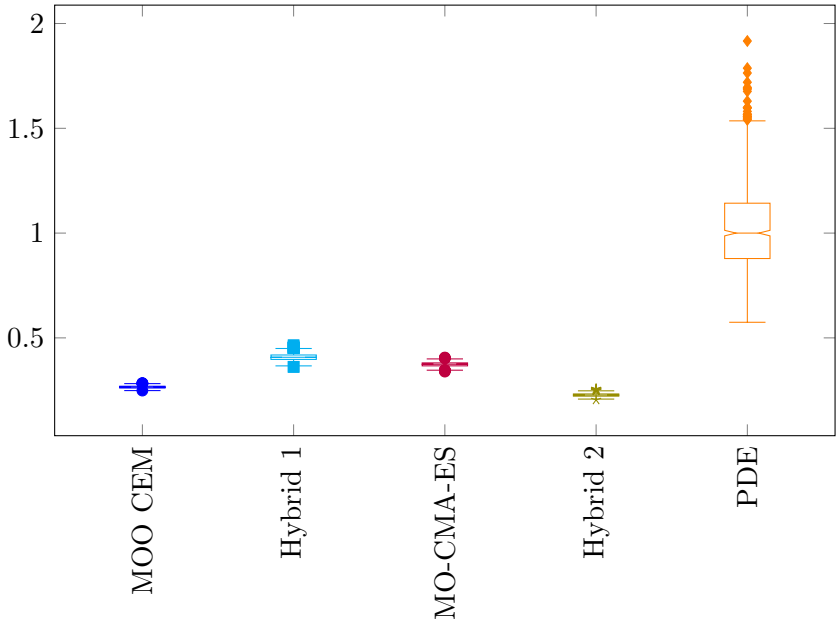


Figure A.21: Box plot of relative run times when solving ZDT 2.

A.8 ZDT 3

Table A.15: Problem details for ZDT 3.

Number of variables	30
Box constraints	$0 \leq x_i \leq 1, \quad i = 1, \ldots, 30$
Geometry	Disconnected
Relationship	No reported relationships
Modality	Multimodal
Function definitions	Minimise both
	$f_1(x_1) = x_1$
	$g(\mathbf{x}) = 1 + 9\frac{\sum_{i=2}^{n} x_i}{n-1}$
	$f_2(\mathbf{x}) = 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g}\right) \sin(10\pi f_1)$

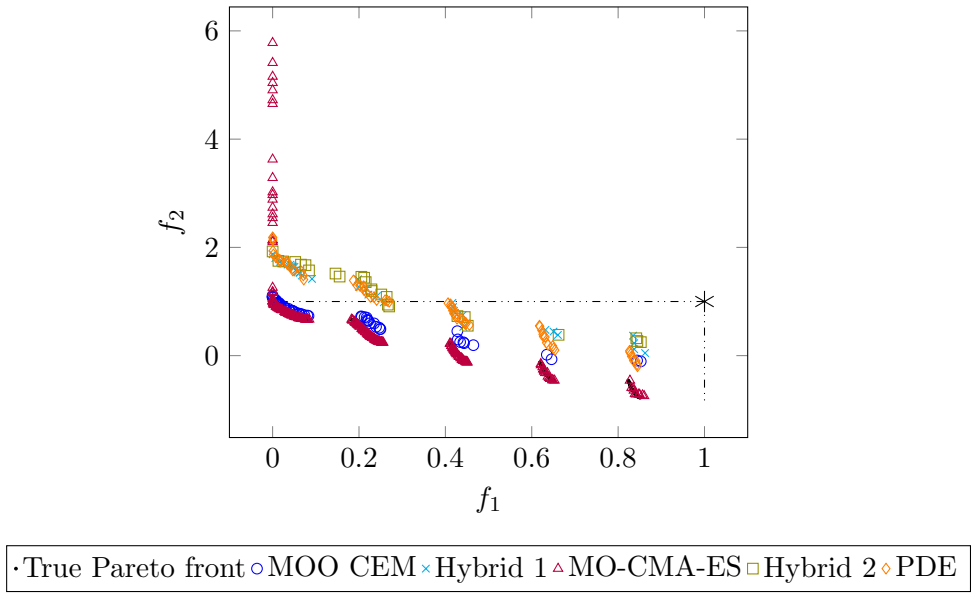


Figure A.22: A sample of Pareto fronts achieved by the algorithms on ZDT 3.

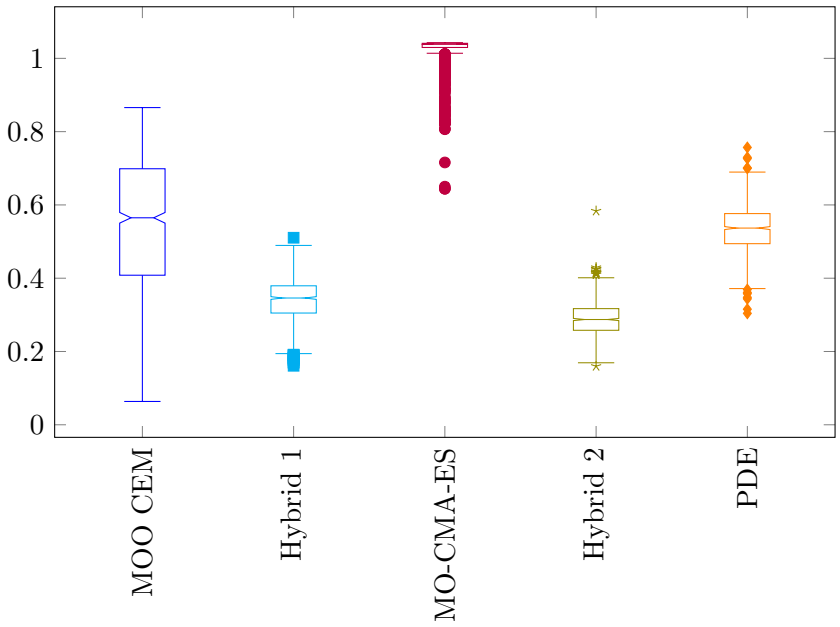


Figure A.23: Box plot of hypervolumes achieved when solving ZDT 3.

Table A.16: Mann-Whitney U-test results for ZDT 3.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	0	1	1	3	**2**
Hybrid 1	0	-	0	1	0	1	**4**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	0	1	0	1	-	2	**3**

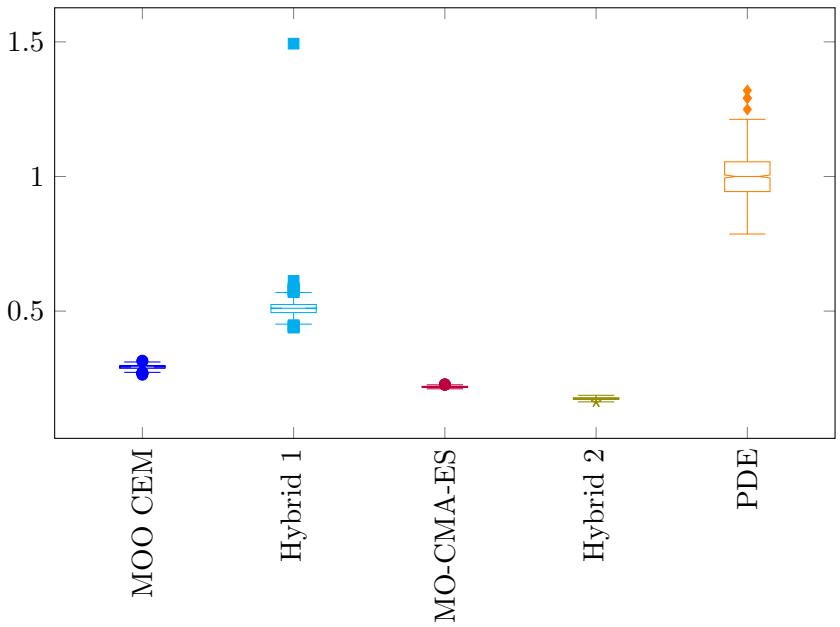


Figure A.24: Box plot of relative run times when solving ZDT 3.

A.9 ZDT 4

Table A.17: Problem details for ZDT 4.

Number of variables	10
Box constraints	$0 \leq x_i \leq 1, \quad i = 1, \ldots, 10$
Geometry	Convex
Relationship	No reported relationships
Modality	Multimodal
Function definitions	Minimise both
	$f_1(x_1) = x_1$
	$g(\mathbf{x}) = 1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i^2 - 10\cos(4\pi x_i)\right)$
	$f_2(\mathbf{x}) = 1 - \sqrt{\frac{f_1}{g}}$

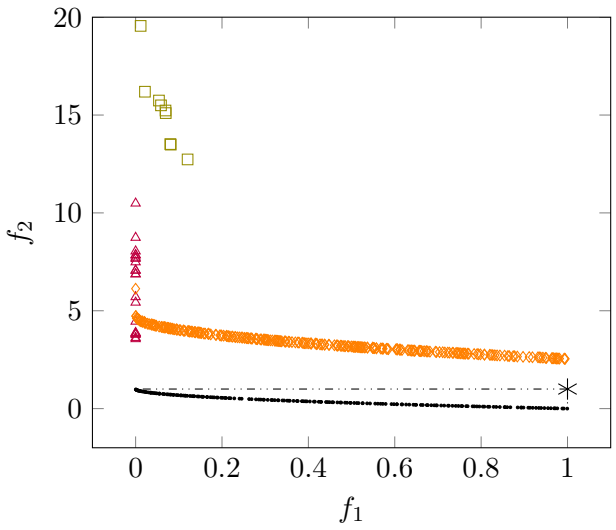


Figure A.25: A sample of Pareto fronts achieved by the algorithms on ZDT 4.

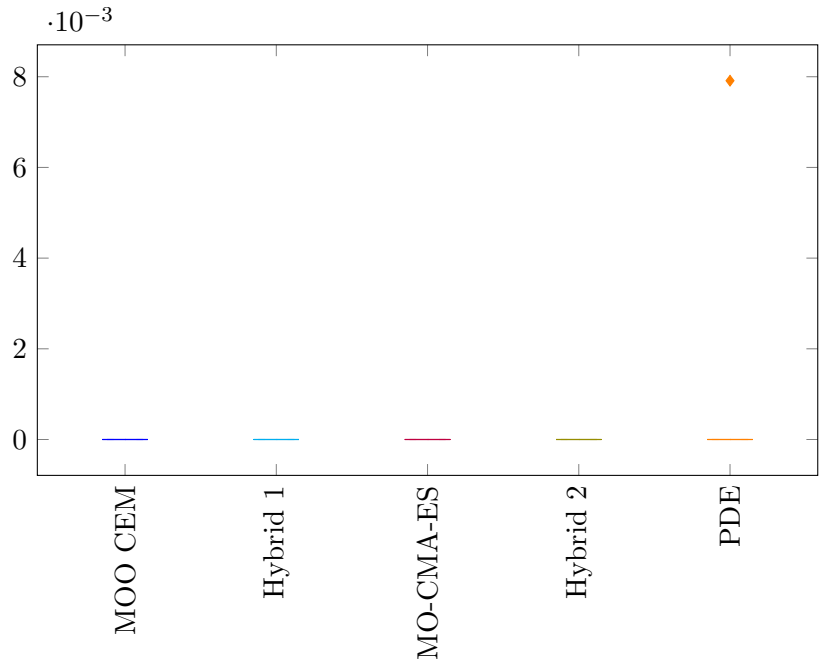


Figure A.26: Box plot of hypervolumes achieved when solving ZDT 4.

Table A.18: Mann-Whitney U-test results for ZDT 4.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	0	-	0	0	0	0	**5**
MO-CMA-ES	0	0	-	0	0	0	**5**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	0	0	0	0	-	0	**5**

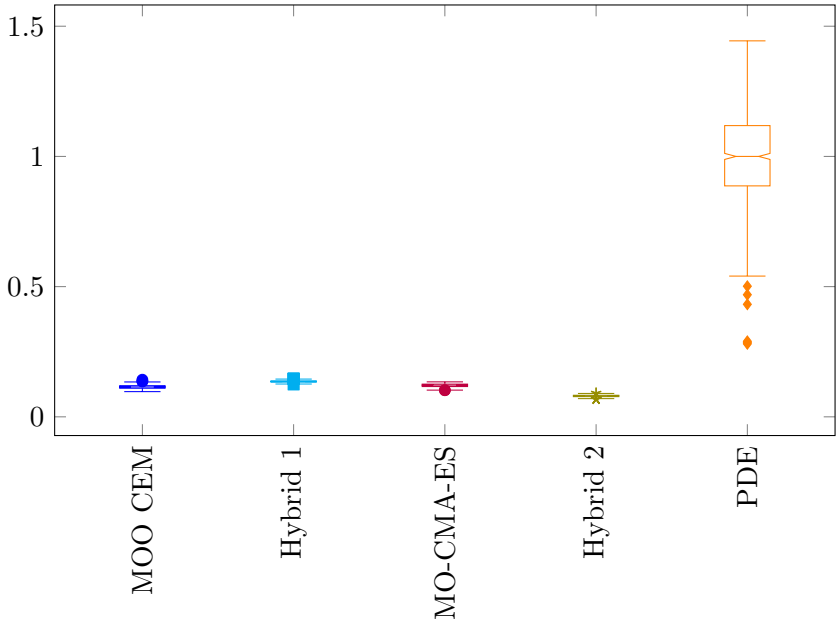


Figure A.27: Box plot of relative run times when solving ZDT 4.

A.10 ZDT 6

Table A.19: Problem details for ZDT 6.

Number of variables	10
Box constraints	$0 \leq x_i \leq 1, \quad i = 1, \ldots, 10$
Geometry	Concave
Relationship	No reported relationships
Modality	Multimodal
Function definitions	Minimise both
	$f_1(x_1) = 1 - e^{-4x_1} \sin^6(6\pi x_1)$
	$g(\mathbf{x}) = 1 + 9\left(\frac{\sum_{i=2}^{n} x_i}{n-1}\right)^{0.25}$
	$f_2(\mathbf{x}) = 1 - \left(\frac{f_1}{g}\right)^2$

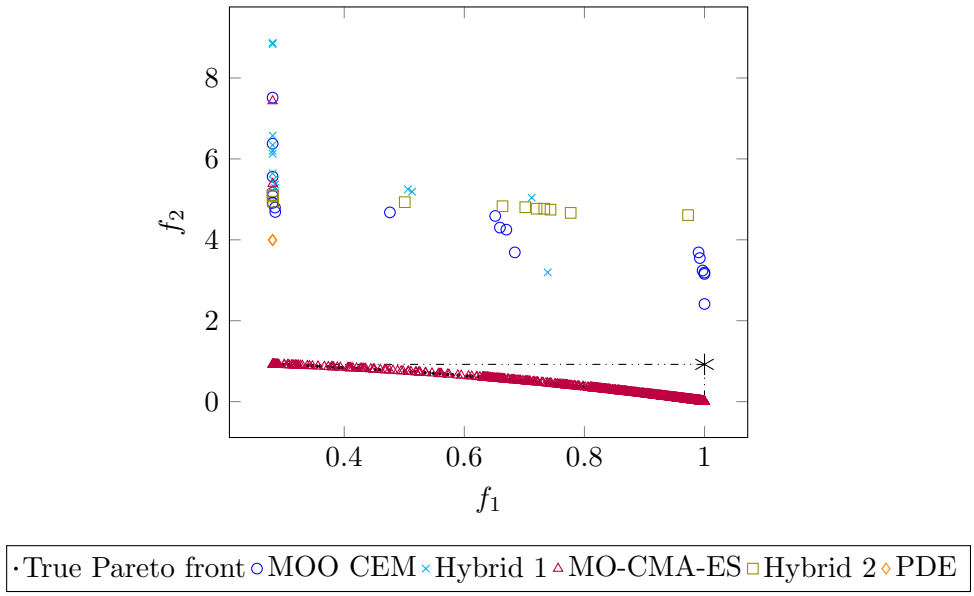


Figure A.28: A sample of Pareto fronts achieved by the algorithms on ZDT 6.

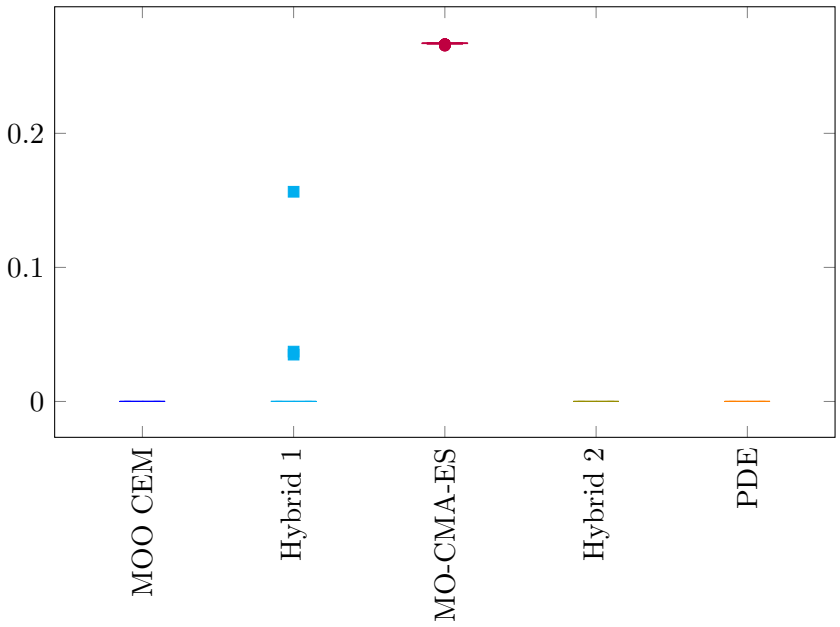


Figure A.29: Box plot of hypervolumes achieved when solving ZDT 6.

Table A.20: Mann-Whitney U-test results for ZDT 6.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	1	1	3	**2**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	0	0	0	0	-	0	**5**

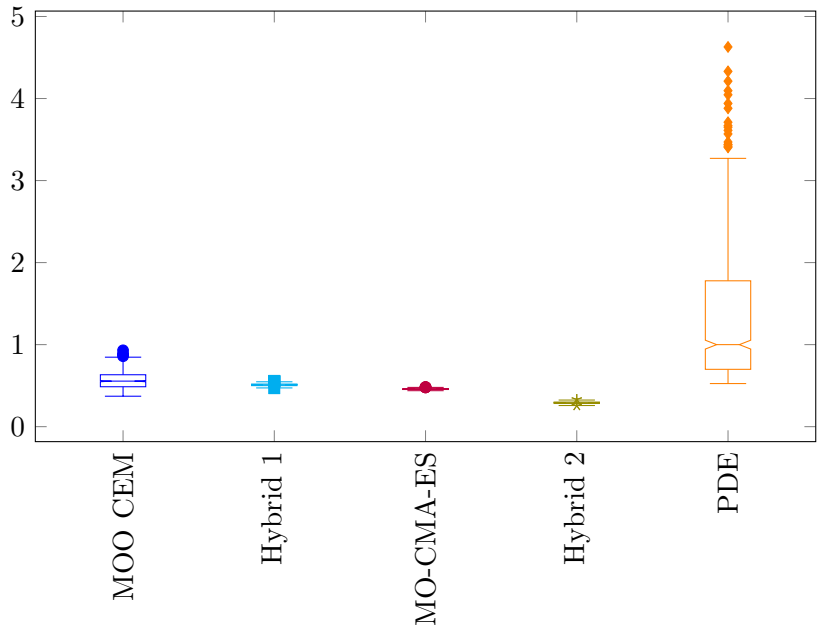


Figure A.30: Box plot of relative run times when solving ZDT 6.

A.11 L_1ZDT 1

Table A.21: Problem details for L_1ZDT 1.

Number of variables	30
Box constraints	$0 \leq x_i \leq 1, \quad i = 1, \ldots, 30$
Geometry	Convex
Relationship	Reported relationships
Modality	Unimodal
Function definitions	Minimise both
	$f_1(x'_1) = x'_1$
	$g(\mathbf{x}') = 1 + 9\frac{\sum_{i=2}^{n} x'_i}{n-1}$
	$f_2(\mathbf{x}') = 1 - \sqrt{\frac{f_1}{g}}$

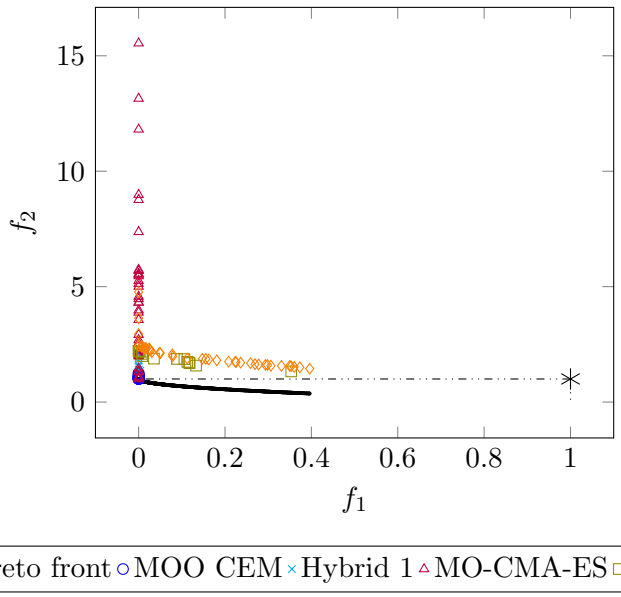


Figure A.31: A sample of Pareto fronts achieved by the algorithms on L_1ZDT 1.

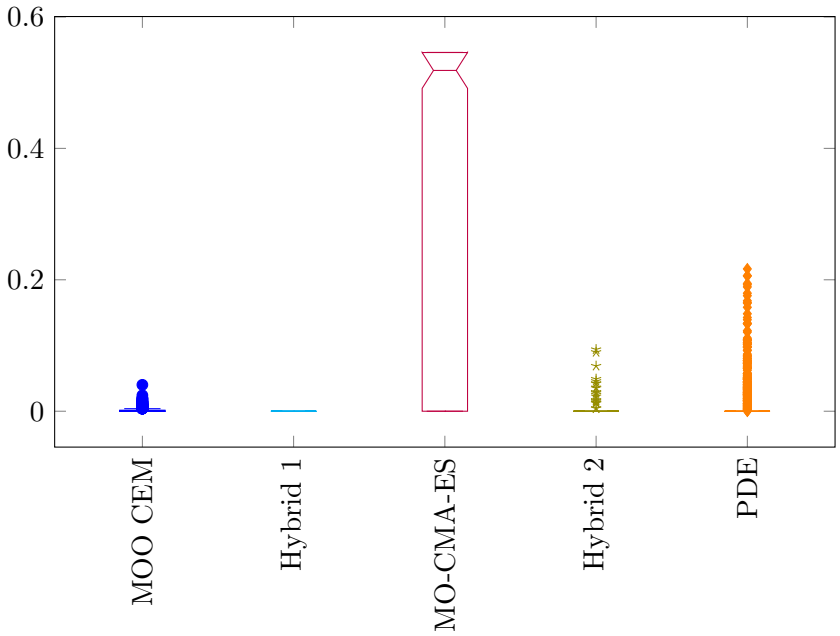


Figure A.32: Box plot of hypervolumes achieved when solving L_1ZDT 1.

Table A.22: Mann-Whitney U-test results for L_1ZDT 1.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	0	1	1	3	**2**
Hybrid 1	0	-	0	0	0	0	**5**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	0	1	0	-	0	1	**4**
PDE	0	1	0	1	-	2	**3**

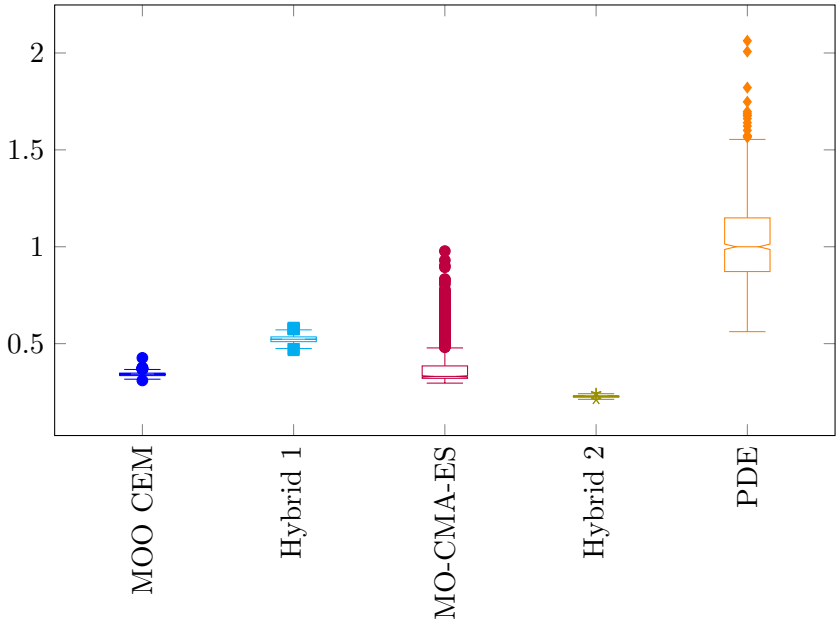


Figure A.33: Box plot of relative run times when solving L_1ZDT 1.

A.12 L_1ZDT 2

Table A.23: Problem details for L_1ZDT 6.

Number of variables	30
Box constraints	$0 \leq x_i \leq 1, \quad i = 1, \ldots, 30$
Geometry	Convex
Relationship	Reported relationships
Modality	Unimodal
Function definitions	Minimise both
	$f_1(x'_1) = x'_1$
	$g(\mathbf{x}') = 1 + 9\frac{\sum_{i=2}^{n} x'_i}{n-1}$
	$f_2(\mathbf{x}') = 1 - \sqrt{\frac{f_1}{g}}$

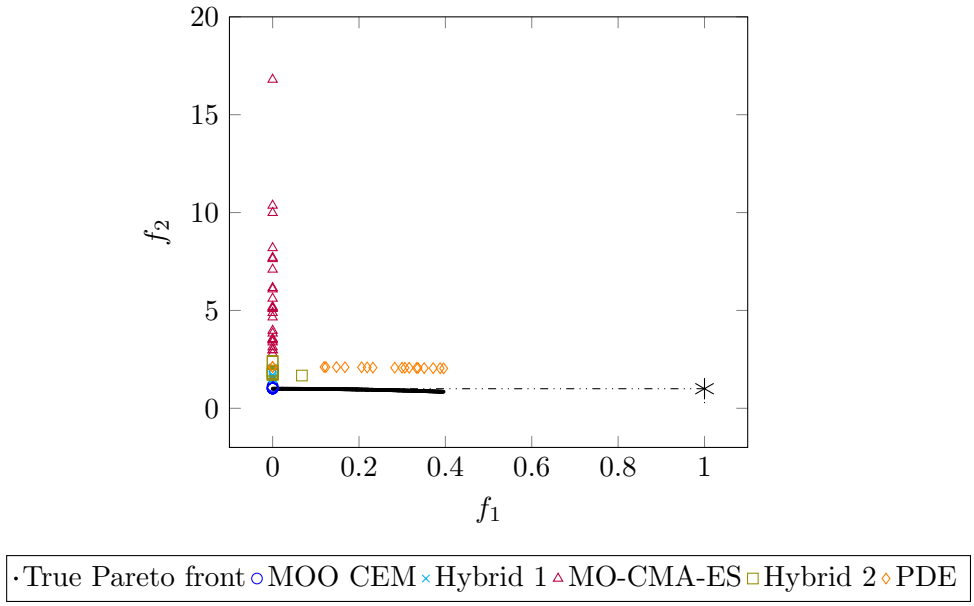


Figure A.34: A sample of Pareto fronts achieved by the algorithms on L_1ZDT 2.

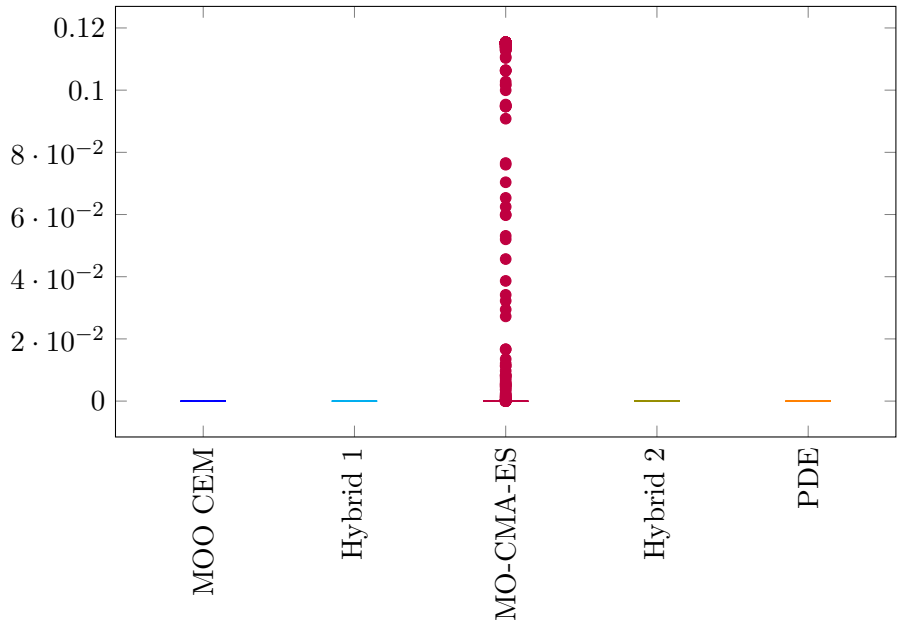


Figure A.35: Box plot of hypervolumes achieved when solving L_1ZDT 2.

Table A.24: Mann-Whitney U-test results for L_1ZDT 2.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	0	-	0	0	0	0	**5**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	0	0	0	0	-	0	**5**

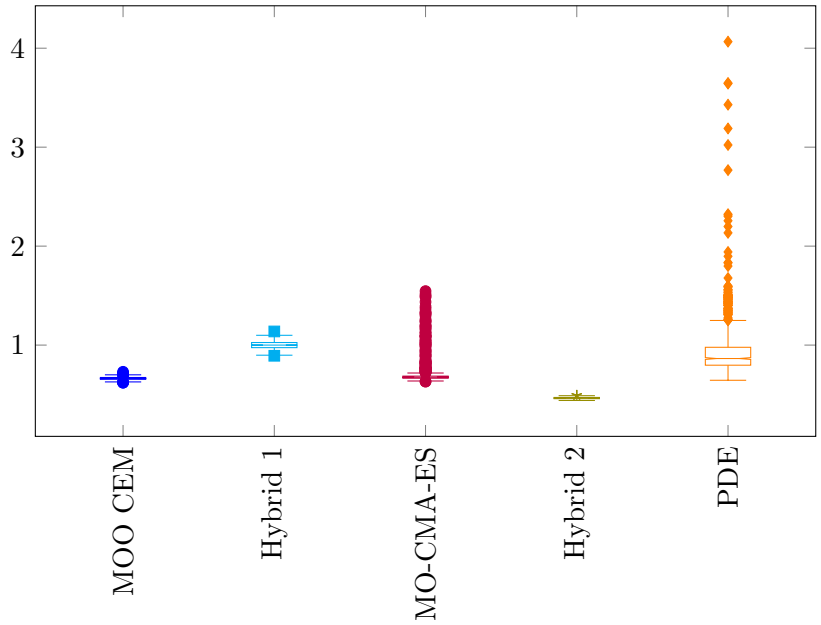


Figure A.36: Box plot of relative run times when solving L_1ZDT 2.

A.13 L_1ZDT 3

Table A.25: Problem details for L_1ZDT 3.

Number of variables	30
Box constraints	$0 \leq x_i \leq 1, \quad i = 1, \ldots, 30$
Geometry	Disconnected
Relationship	Reported relationships
Modality	Multimodal
Function definitions	Minimise both
	$f_1(x'_1) = x'_1$
	$g(\mathbf{x}') = 1 + 9\frac{\sum_{i=2}^{n} x'_i}{n-1}$
	$f_2(\mathbf{x}') = 1 - \sqrt{\frac{f_1}{g}} - \left(\frac{f_1}{g}\right) \sin(10\pi f_1)$

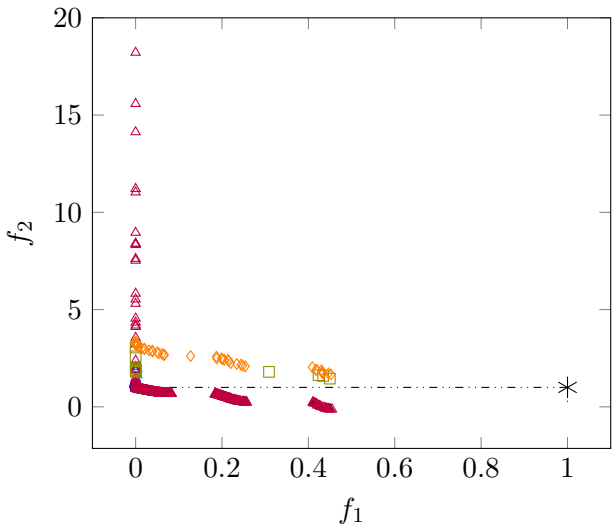


Figure A.37: A sample of Pareto fronts achieved by the algorithms on L_1ZDT 3.

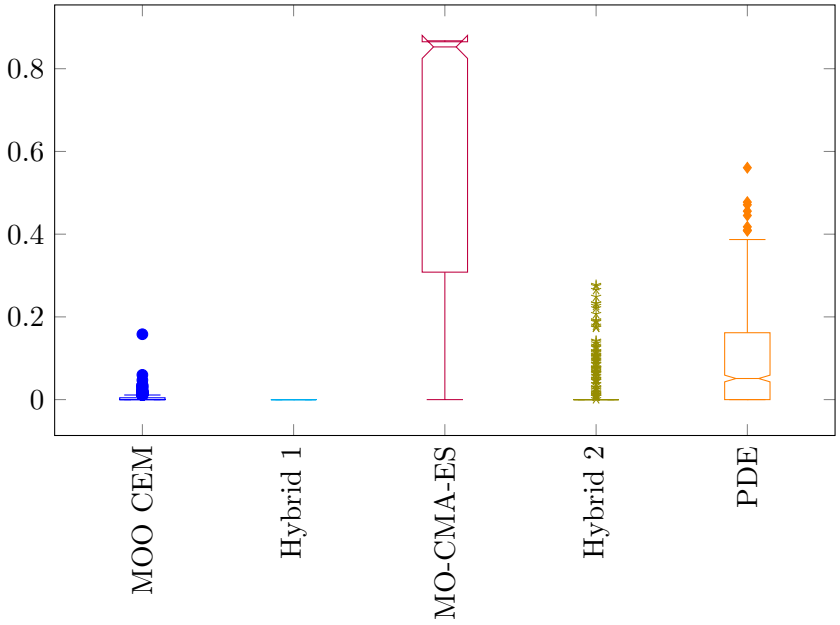


Figure A.38: Box plot of hypervolumes achieved when solving L_1ZDT 3.

Table A.26: Mann-Whitney U-test results for L_1ZDT 3.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	0	1	0	2	**3**
Hybrid 1	0	-	0	0	0	0	**5**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	0	1	0	-	0	1	**4**
PDE	1	1	0	1	-	3	**2**

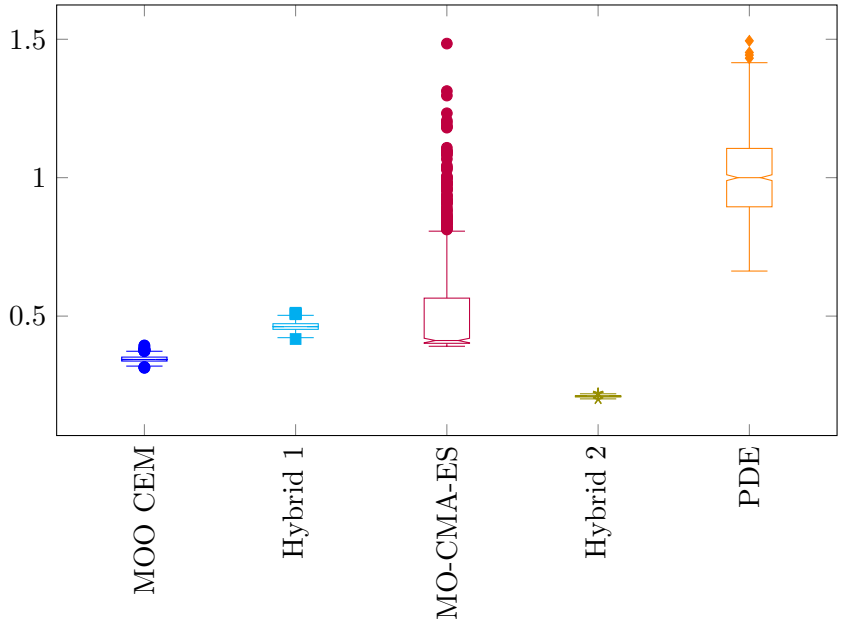


Figure A.39: Box plot of relative run times when solving L_1ZDT 3.

A.14 L_1ZDT 4

Table A.27: Problem details for L_1ZDT 4.

Number of variables	10	
Box constraints	$0 \leq x_i \leq 1, i = 1, \ldots, 10$	
Geometry	Convex	
Relationship	Reported relationships	
Modality	Multimodal	
Function definitions	Minimise both	
	$f_1(x'_1)$	$= \quad x'_1$
	$g(\mathbf{x}')$	$= \quad 1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i^2 - 10\cos(4\pi x'_i)\right)$
	$f_2(\mathbf{x}')$	$= \quad 1 - \sqrt{\frac{f_1}{g}}$

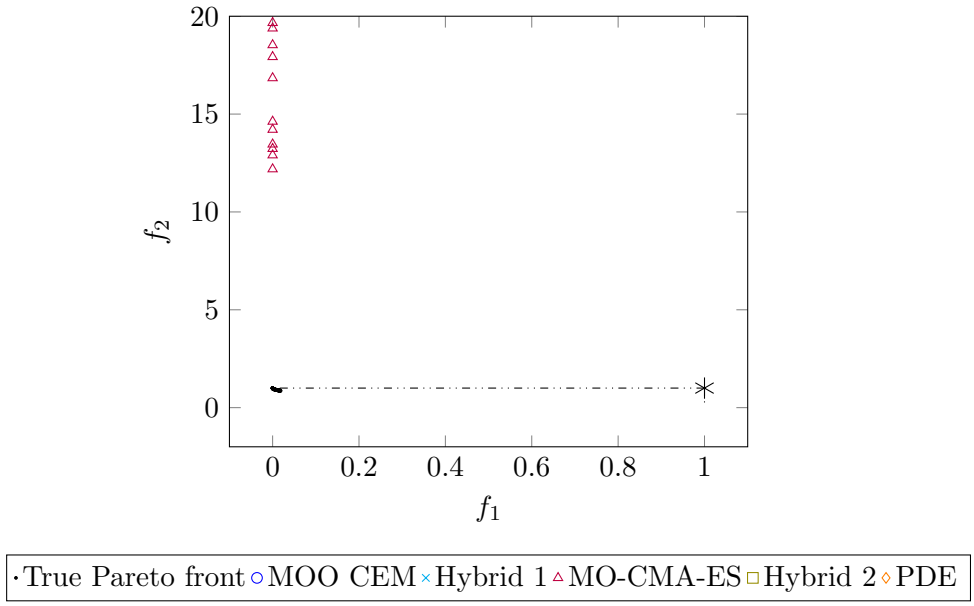


Figure A.40: A sample of Pareto fronts achieved by the algorithms on L_1ZDT 4.

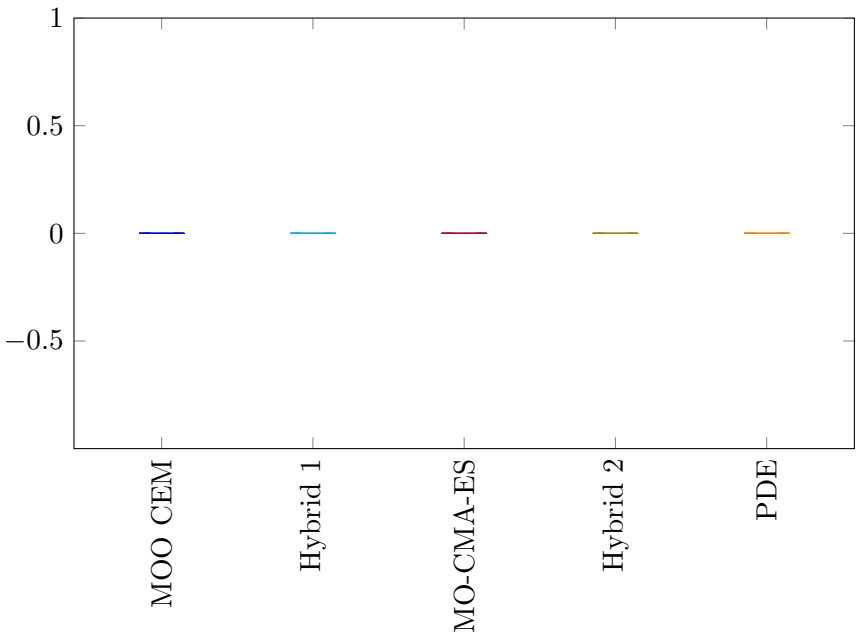


Figure A.41: Box plot of hypervolumes achieved when solving L_1ZDT 4.

Table A.28: Mann-Whitney U-test results for L_1ZDT 4.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	0	-	0	0	0	0	**5**
MO-CMA-ES	0	0	-	0	0	0	**5**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	0	0	0	0	-	0	**5**

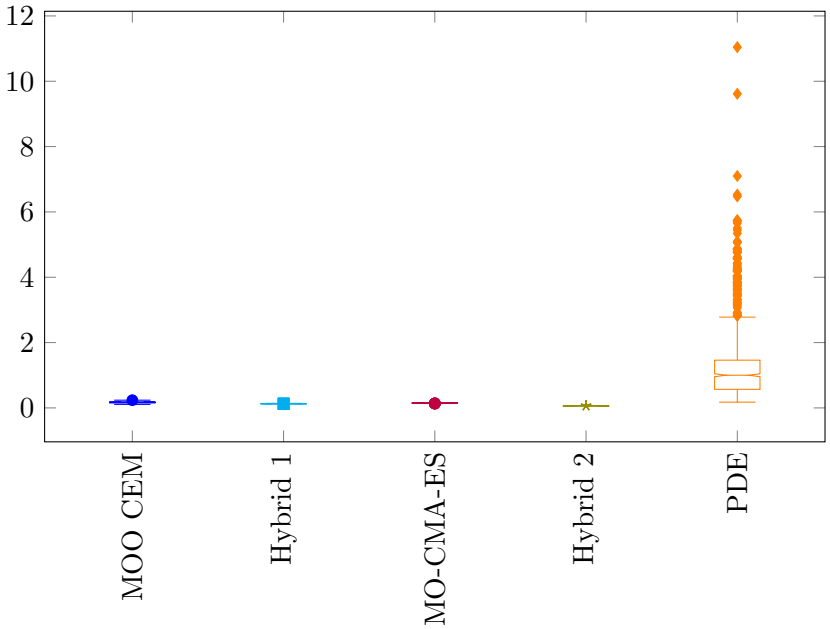


Figure A.42: Box plot of relative run times when solving L_1ZDT 4.

A.15 L_1ZDT 6

Table A.29: Problem details for L_1ZDT 6.

Number of variables	10
Box constraints	$0 \leq x_i \leq 1, i = 1, \ldots, 10$
Geometry	Concave
Relationship	Reported relationships
Modality	Multimodal
Function definitions	Minimise both
	$f_1(x'_1) = 1 - e^{-4x'_1} \sin^6(6\pi x'_1)$
	$g(\mathbf{x}') = 1 + 9\left(\frac{\sum_{i=2}^{n} x'_i}{n-1}\right)^{0.25}$
	$f_2(\mathbf{x}') = 1 - \left(\frac{f_1}{g}\right)^2$

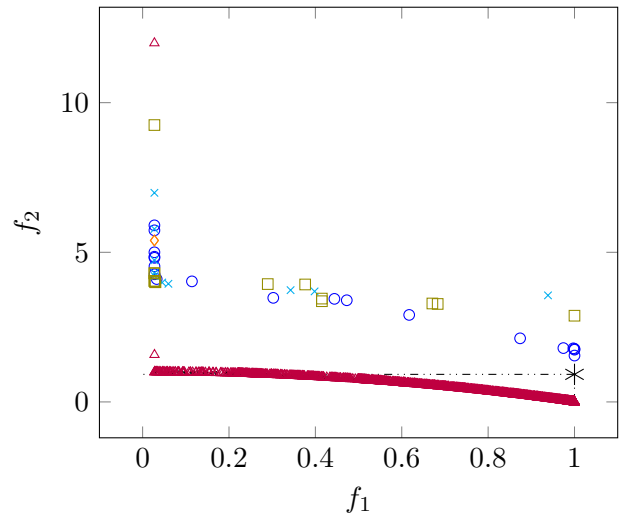


Figure A.43: A sample of Pareto fronts achieved by the algorithms on L_1ZDT 6.

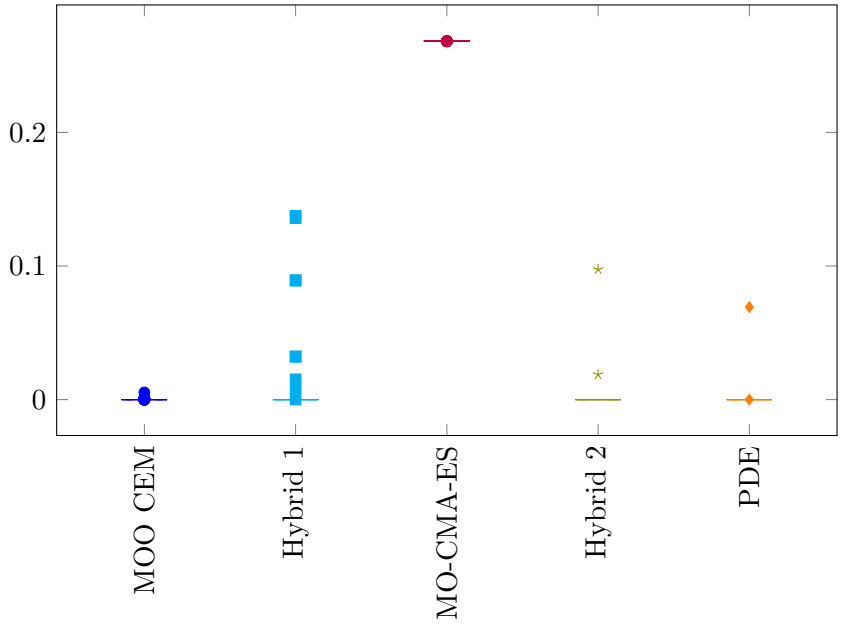


Figure A.44: Box plot of hypervolumes achieved when solving L_1ZDT 6.

Table A.30: Mann-Whitney U-test results for L_1ZDT 6.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	0	1	1	3	**2**
Hybrid 1	0	-	0	1	1	2	**3**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	0	0	0	0	-	0	**5**

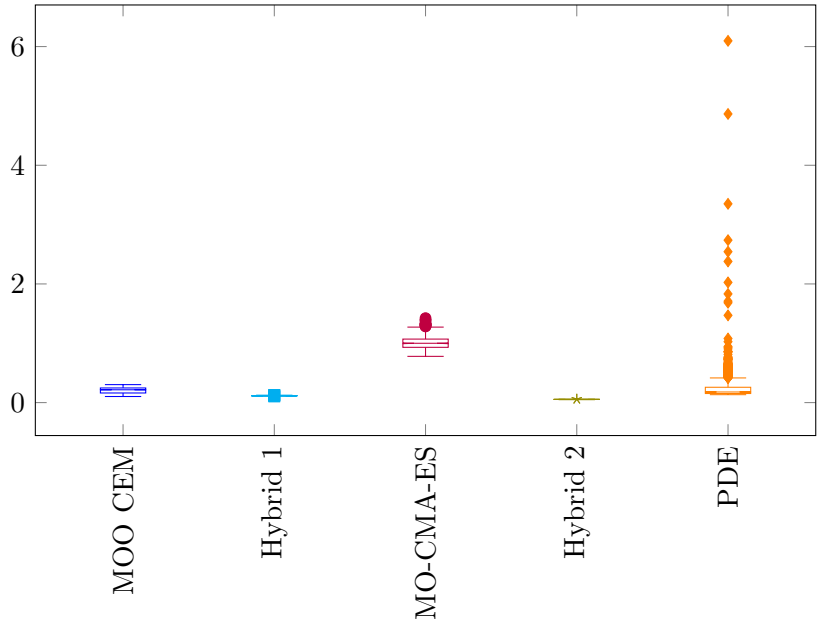


Figure A.45: Box plot of relative run times when solving L_1ZDT 6.

A.16 R 1

Table A.31: Problem details for R 1.

Number of variables	10	
Box constraints	$-0.3 \leq x_i \leq 0.3, \quad i = 1, \ldots, 10$	
Geometry	Convex	
Relationship	Reported relationships	
Modality	Multimodal, deceptive	
Function definitions	Minimise both	
	$f_1(x_1')$	$= \quad x_1'$
	$g(\mathbf{x}')$	$= \quad 1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i'^2 - 10\cos(4\pi x_i')\right)$
	$h\left(f_1(x_1'), g(\mathbf{x}')\right)$	$= \quad e^{\frac{-f_1(x_i')}{g(\mathbf{x}')}}$
	$f_2(\mathbf{x}')$	$= \quad g(\mathbf{x}')h\left(f_1(x_1'), g(\mathbf{x}')\right)$
	subject to	$\lvert f_1 \rvert \leq 0.3$

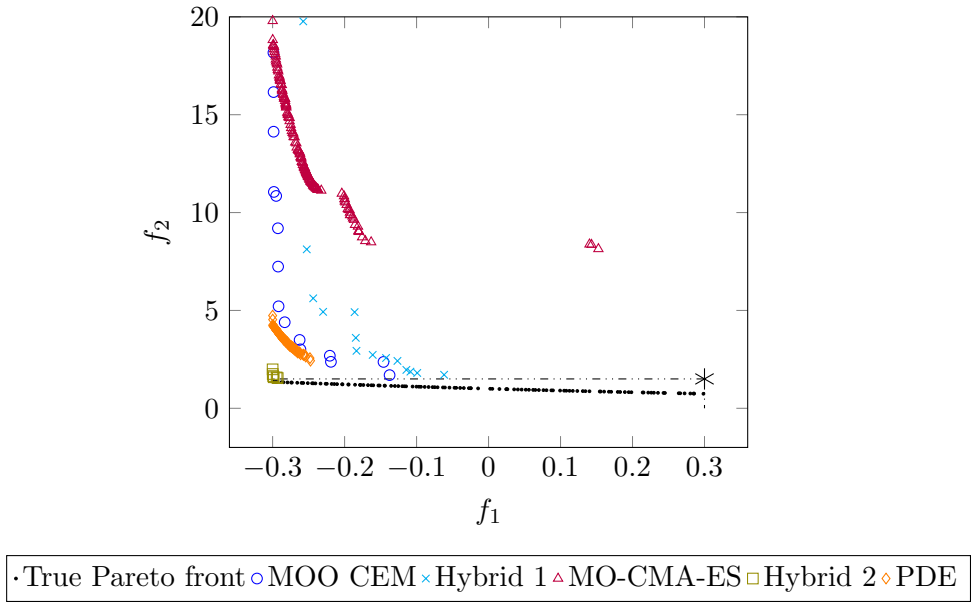


Figure A.46: A sample of Pareto fronts achieved by the algorithms on R 1.

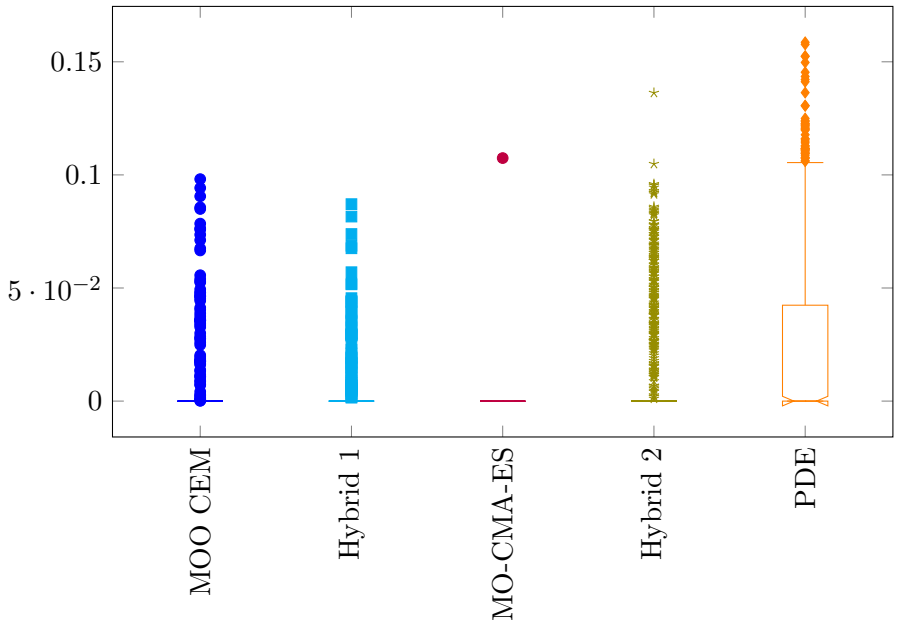


Figure A.47: Box plot of hypervolumes achieved when solving R 1.

Table A.32: Mann-Whitney U-test results for R 1.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	1	0	0	1	**4**
Hybrid 1	0	-	1	0	0	1	**4**
MO-CMA-ES	0	0	-	0	0	0	**5**
Hybrid 2	1	1	1	-	0	3	**2**
PDE	1	1	1	1	-	4	**1**

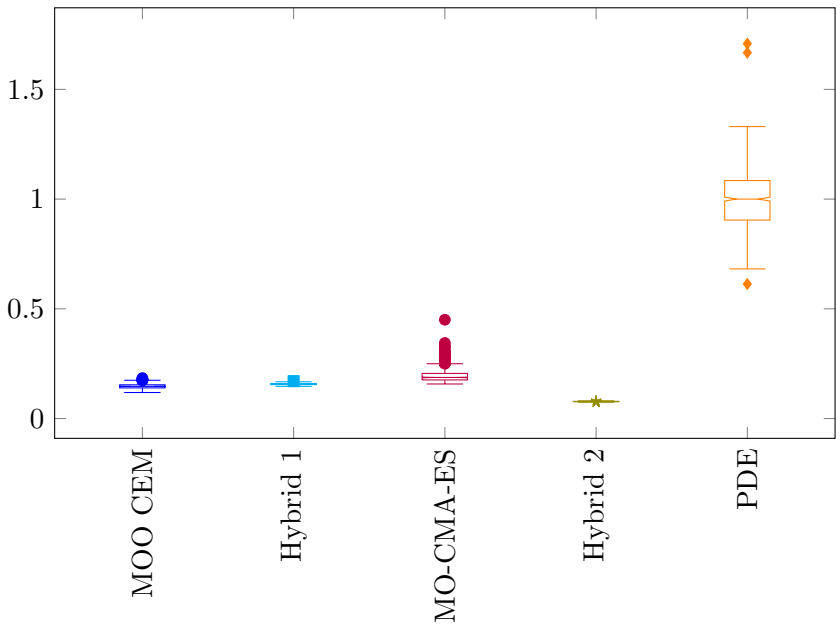


Figure A.48: Box plot of relative run times when solving R 1.

A.17 R 2

Table A.33: Problem details for R 2.

Number of variables	10	
Box constraints	$-1 \leq x_i \leq 1, \quad i = 1, \ldots, 10$	
Geometry	Disconnected	
Relationship	Reported relationships	
Modality	Multimodal	
Function definitions	Minimise both	
	$f_1(x_1')$	$= \quad x_1'$
	$g(\mathbf{x}')$	$= \quad 1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i'^2 - 10\cos(\pi x_i')\right)$
	$h\left(f_1(x_1'), g(\mathbf{x}')\right)$	$= \quad 1 + e^{\frac{-f_1(x_i')}{g(\mathbf{x}')}} + \left(\frac{f_1(x_1')+1}{g(\mathbf{x}')}\right) \sin(5\pi f_1(x_1'))$
	$f_2(\mathbf{x}')$	$= \quad g(\mathbf{x}')h\left(f_1(x_1'), g(\mathbf{x}')\right)$
	subject to	$\lvert f_1 \rvert \leq 1$

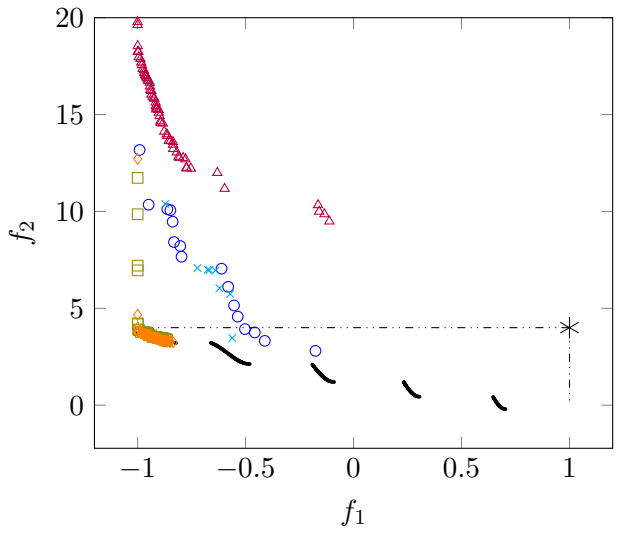


Figure A.49: A sample of Pareto fronts achieved by the algorithms on R 2.

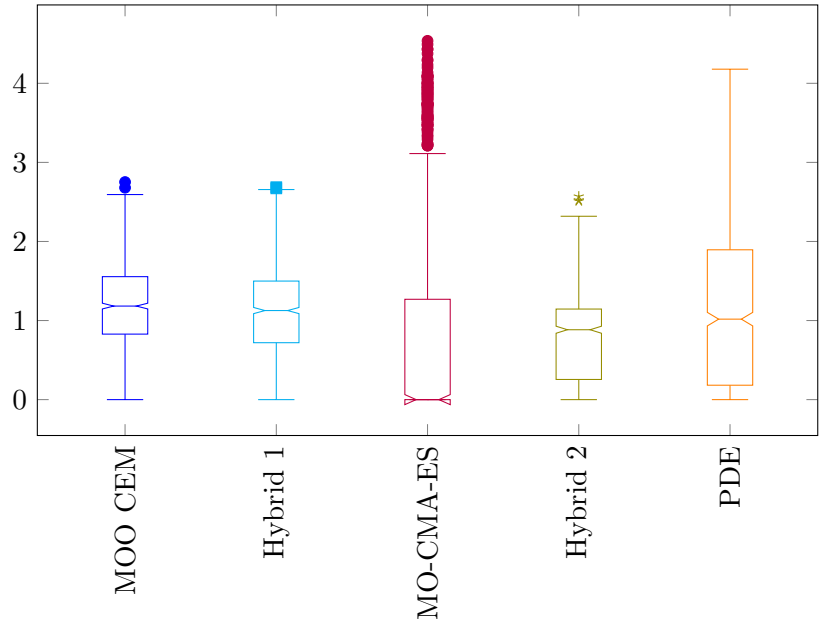


Figure A.50: Box plot of hypervolumes achieved when solving R 2.

Table A.34: Mann-Whitney U-test results for R 2.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	1	1	1	4	**1**
Hybrid 1	0	-	1	1	1	3	**2**
MO-CMA-ES	0	0	-	0	0	0	**5**
Hybrid 2	0	0	1	-	0	1	**4**
PDE	0	0	1	1	-	2	**3**

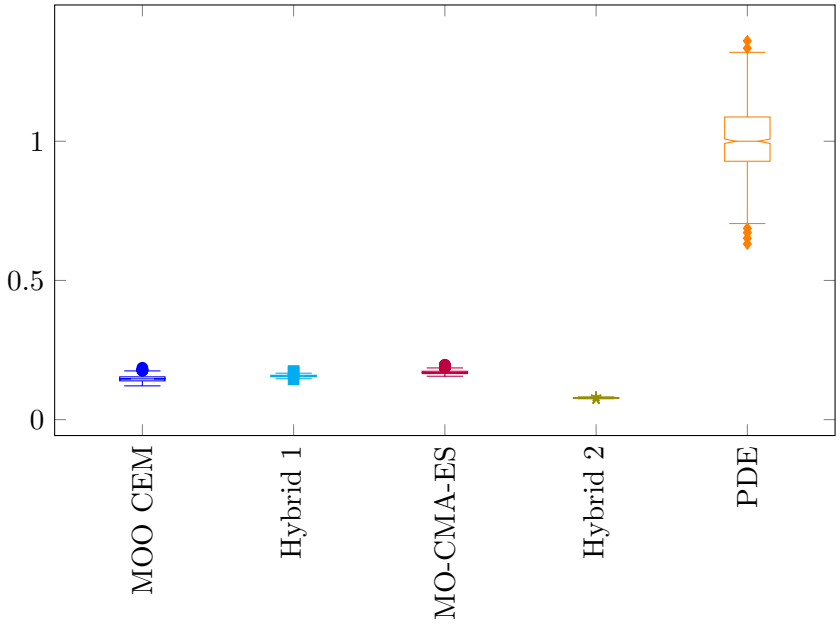


Figure A.51: Box plot of relative run times when solving R 2.

A.18 R 3

Table A.35: Problem details for R 3.

Number of variables		10
Box constraints		$-1 \leq x_i \leq 1, \quad i = 1, \ldots, 10$
Geometry		Concave
Relationship		Reported relationships
Modality		Multimodal
Function definitions	Minimise both	
	$f_1(x_1')$	$= 1 - \frac{e^{2x_1'} \sin^6(6\pi x_1')}{9}$
	$g(\mathbf{x}')$	$= 1 + 10(n-1) + \sum_{i=2}^{n} \left(x_i'^2 - 10\cos(\pi x_i')\right)$
	$h\left(f_1(x_1'), g(\mathbf{x}')\right)$	$= 1 - \left(\frac{f_1(x_i')}{g(\mathbf{x}')}\right)^2$
	$f_2(\mathbf{x}')$	$= g(\mathbf{x}')h\left(f_1(x_1'), g(\mathbf{x}')\right)$
	subject to	$0.3 \leq f_1 \leq 1$

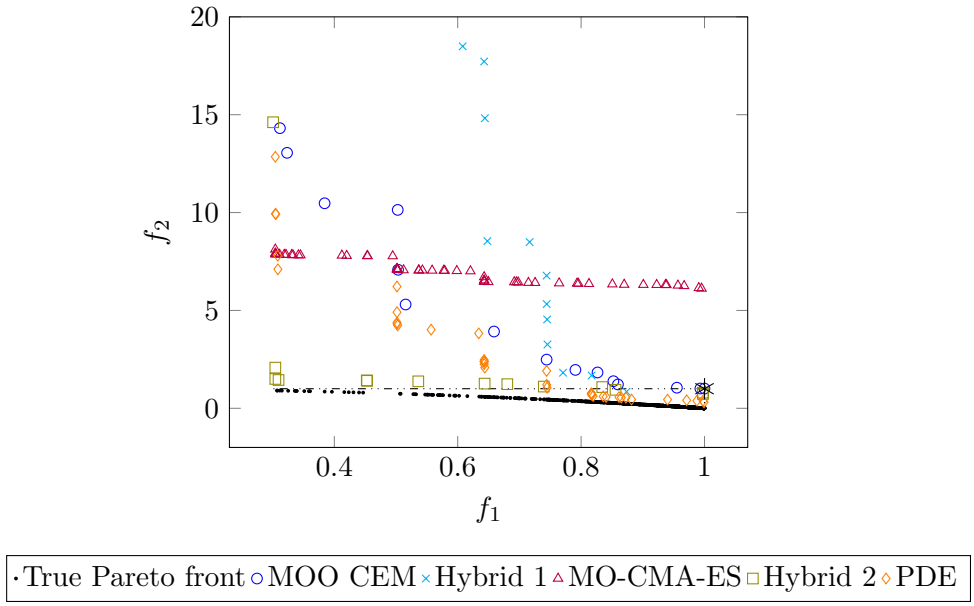


Figure A.52: A sample of Pareto fronts achieved by the algorithms on R 3.

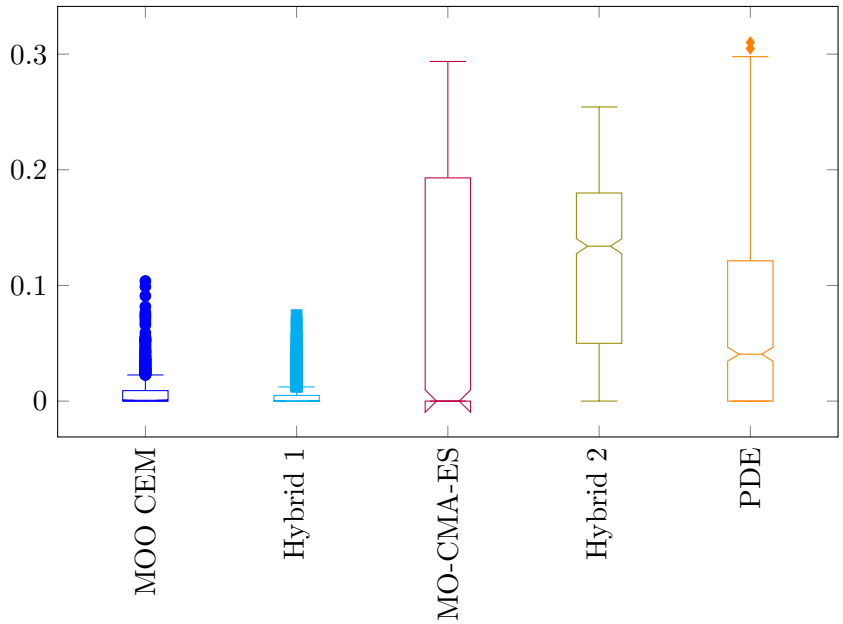


Figure A.53: Box plot of hypervolumes achieved when solving R 3.

Table A.36: Mann-Whitney U-test results for R 3.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	1	0	0	2	**3**
Hybrid 1	0	-	0	0	0	0	**5**
MO-CMA-ES	0	0	-	0	0	0	**5**
Hybrid 2	1	1	1	-	1	4	**1**
PDE	1	1	1	0	-	3	**2**

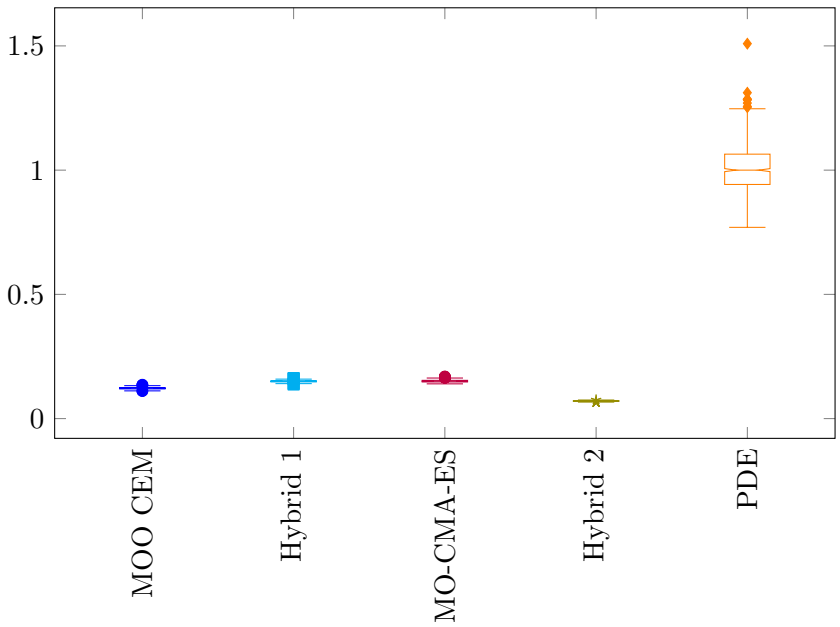


Figure A.54: Box plot of relative run times when solving R 3.

A.19 R 4

Table A.37: Problem details for R 4.

Number of variables	10	
Box constraints	$-1 \leq x_i \leq 1, \quad i = 1, \ldots, 10$	
Geometry	Convex	
Relationship	Reported relationships	
Modality	Multimodal, deceptive	
Function definitions	Minimise both	
	$f_1(x'_1)$	$= \quad x'_1$
	$g(\mathbf{x}')$	$= \quad 1 + 0.015578(n-1)$
		$+ \sum_{i=2}^{n} \left(x_i'^2 - 0.25 x'_i \sin(32\sqrt{\lvert x'_i \rvert}) \right)$
	$h\left(f_1(x'_1), g(\mathbf{x}')\right)$	$= \quad e^{\frac{-f_1(x'_i)}{g(\mathbf{x}')}}$
	$f_2(\mathbf{x}')$	$= \quad g(\mathbf{x}') h\left(f_1(x'_1), g(\mathbf{x}')\right)$
	subject to	$\lvert f_1 \rvert \leq 1$

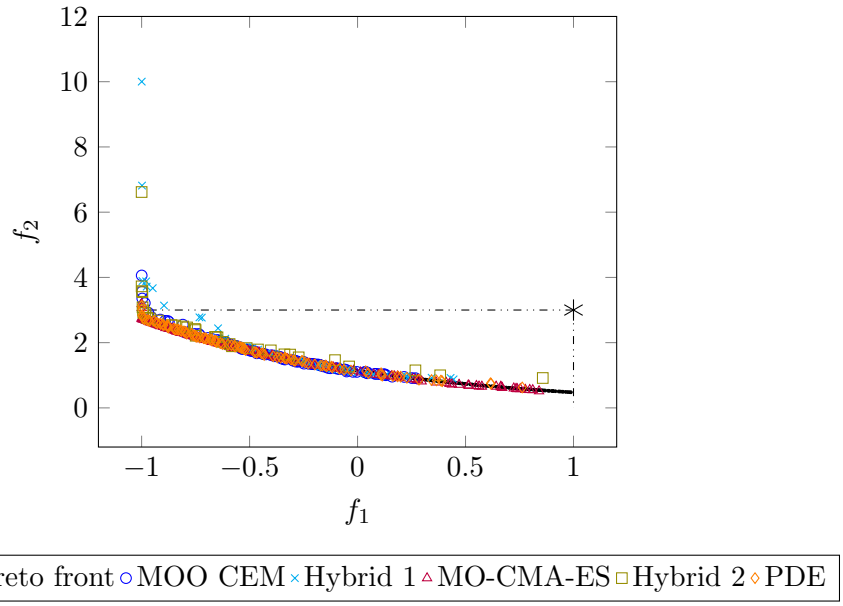


Figure A.55: A sample of Pareto fronts achieved by the algorithms on R 4.

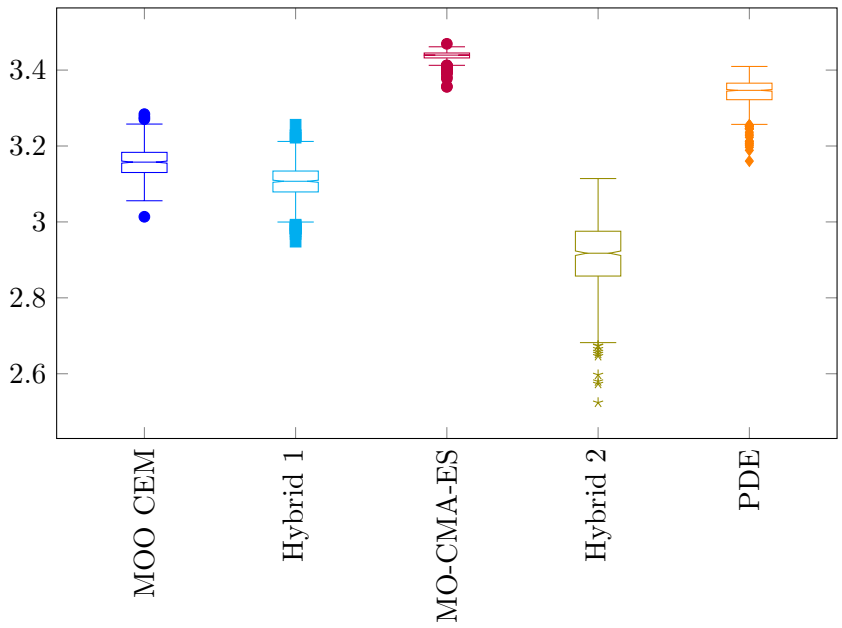


Figure A.56: Box plot of hypervolumes achieved when solving R 4.

Table A.38: Mann-Whitney U-test results for R 4.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	0	1	0	2	**3**
Hybrid 1	0	-	0	1	0	1	**4**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	1	1	0	1	-	3	**2**

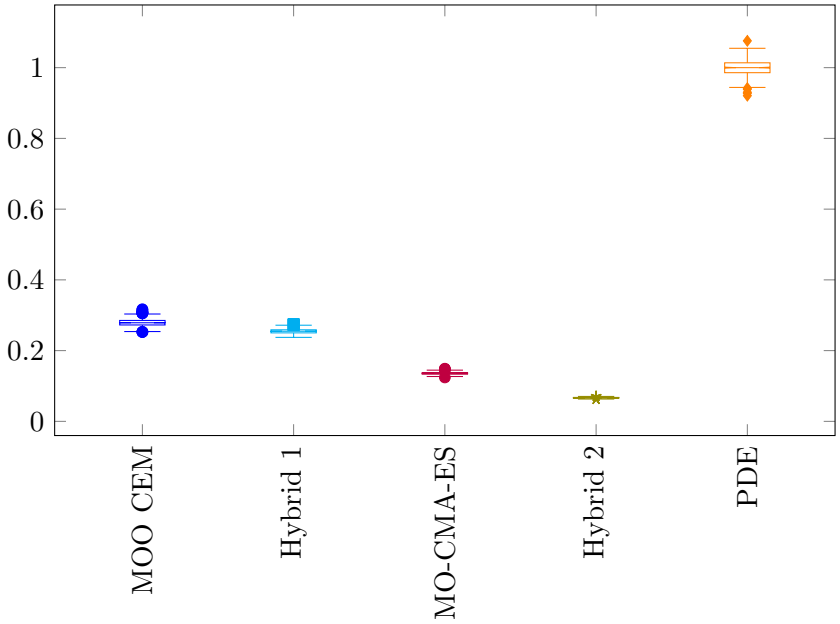


Figure A.57: Box plot of relative run times when solving R 4.

A.20 WFG 1

Table A.39: Problem details for WFG 1.

Number of variables	$V = 4$, 20 or 100
Box constraints	$0 \leq x_i \leq 2i, \quad i = 1, \ldots, V$
Geometry	Convex, mixed
Relationship	No reported relationships
Modality	Unimodal
Function definitions	See Table 4.9

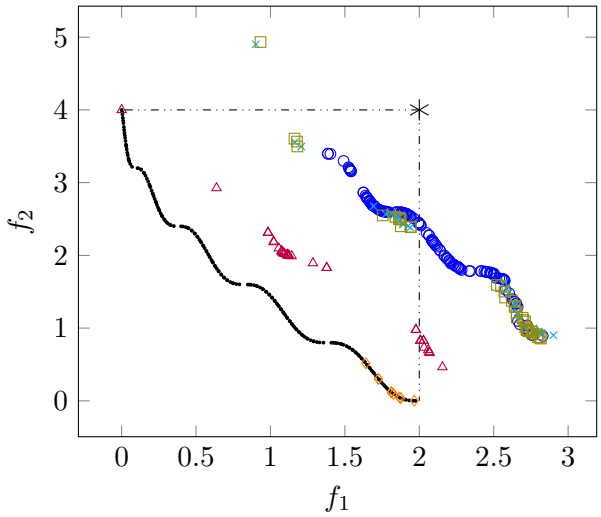


Figure A.58: A sample of Pareto fronts achieved by the algorithms on WFG 1 with four variables.

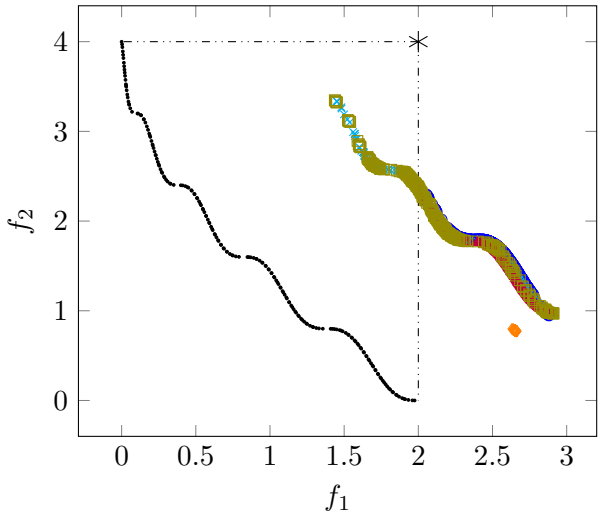


Figure A.59: A sample of Pareto fronts achieved by the algorithms on WFG 1 with 20 variables.

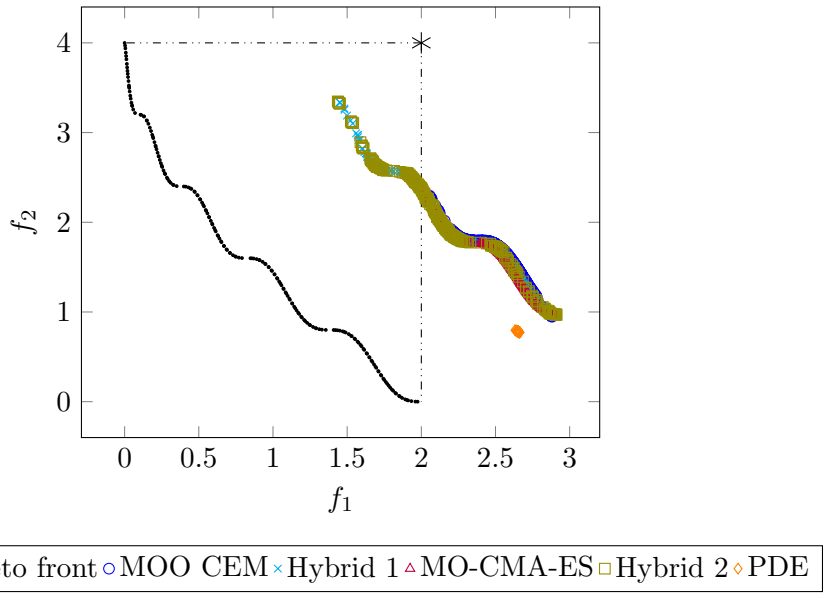


Figure A.60: A sample of Pareto fronts achieved by the algorithms on WFG 1 with a hundred variables.

Table A.40: Mann-Whitney U-test results for WFG 1 with four variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	0	0	1	**4**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	1	1	0	-	0	2	**3**
PDE	0	0	0	0	-	0	**5**

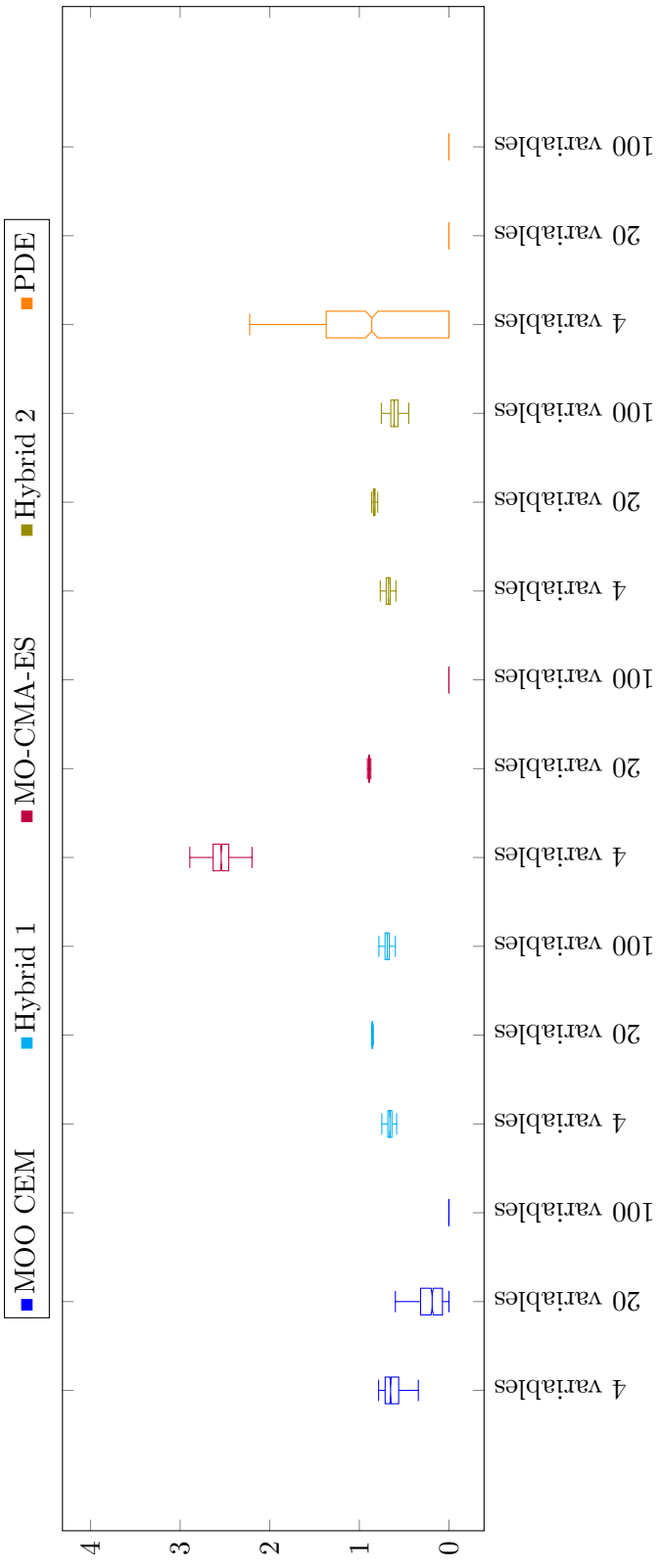


Figure A.61: Box plot of hypervolumes achieved when solving WFG 1 with two objectives.

Table A.41: Mann-Whitney U-test results for WFG 1 with 20 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	1	1	**4**
Hybrid 1	1	-	0	1	1	3	**2**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	1	0	0	-	1	2	**3**
PDE	0	0	0	0	-	0	**5**

Table A.42: Mann-Whitney U-test results for WFG 1 with 100 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	1	1	**4**
Hybrid 1	1	-	1	1	1	4	**1**
MO-CMA-ES	1	0	-	0	1	2	**3**
Hybrid 2	1	0	1	-	1	3	**2**
PDE	0	0	0	0	-	0	**5**

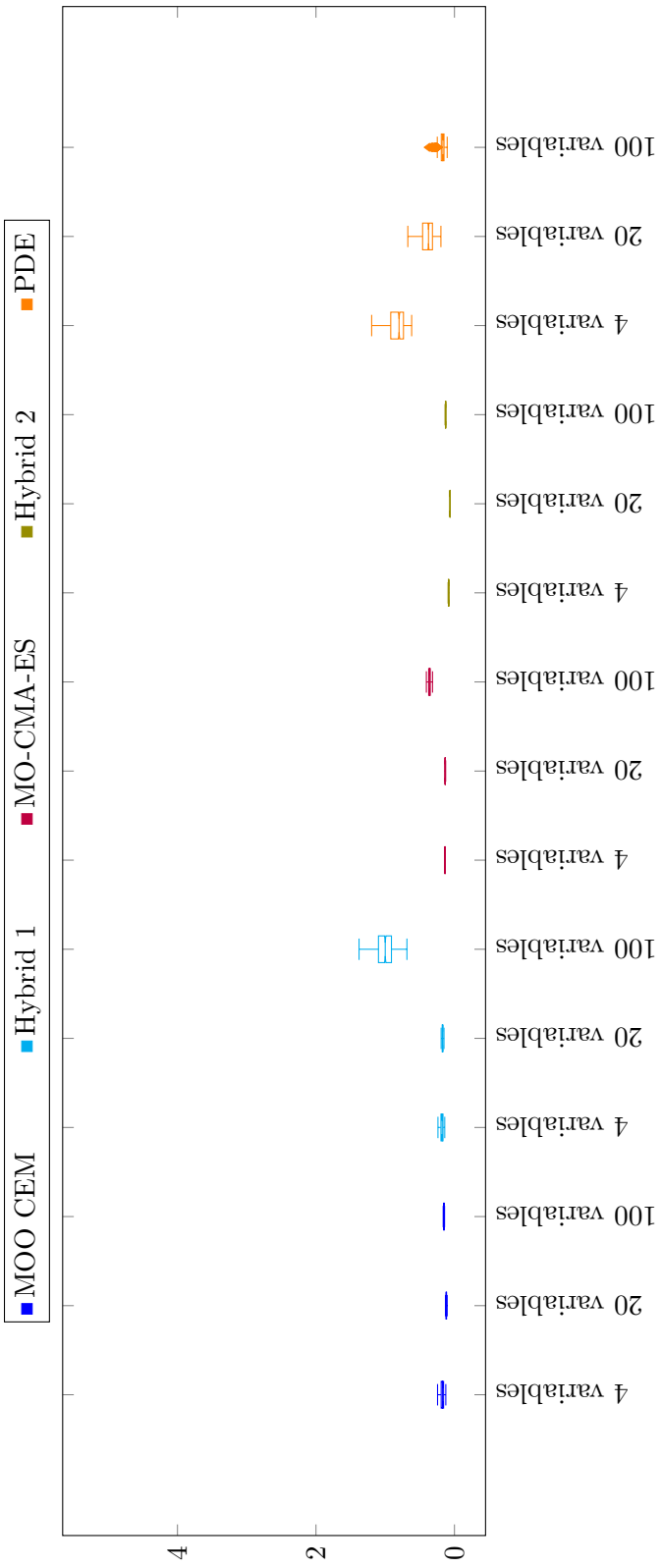


Figure A.62: Box plot of runtimes when solving WFG 1 with two objectives.

A.21 WFG 2

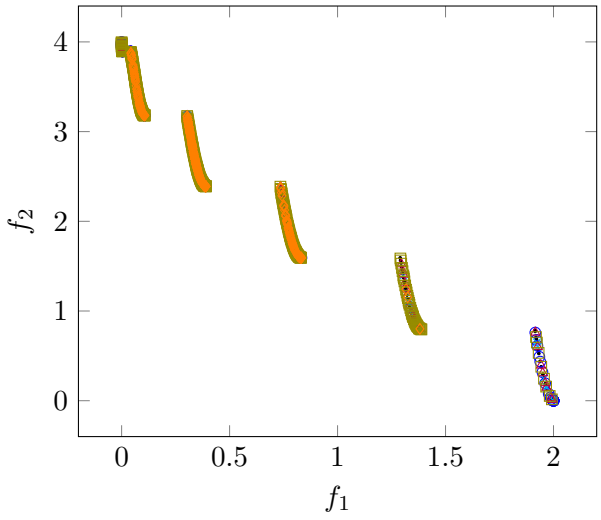


Figure A.63: A sample of Pareto fronts achieved by the algorithms on WFG 2 with four variables.

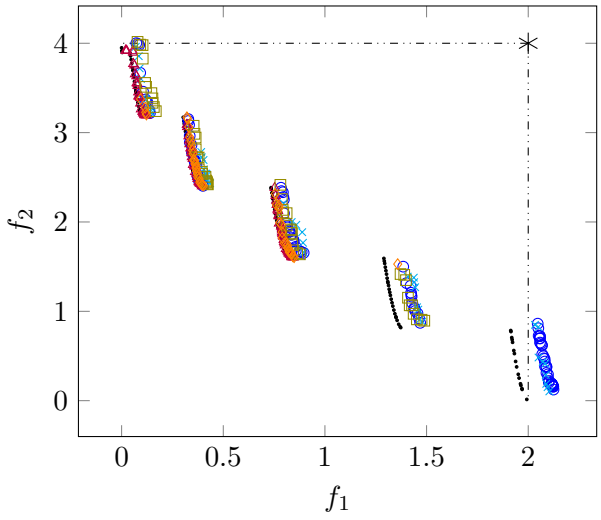


Figure A.64: A sample of Pareto fronts achieved by the algorithms on WFG 2 with 20 variables.

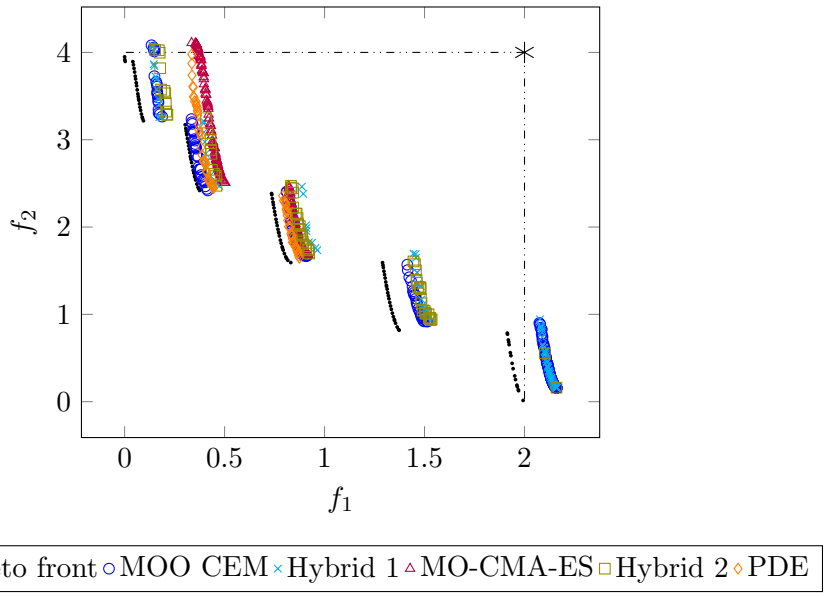


Figure A.65: A sample of Pareto fronts achieved by the algorithms on WFG 2 with 100 variables.

Table A.43: Mann-Whitney U-test results for WFG 2 with four variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	1	1	**4**
Hybrid 1	1	-	0	0	1	2	**3**
MO-CMA-ES	1	1	-	0	1	3	**2**
Hybrid 2	1	1	1	-	1	4	**1**
PDE	0	0	0	0	-	0	**5**

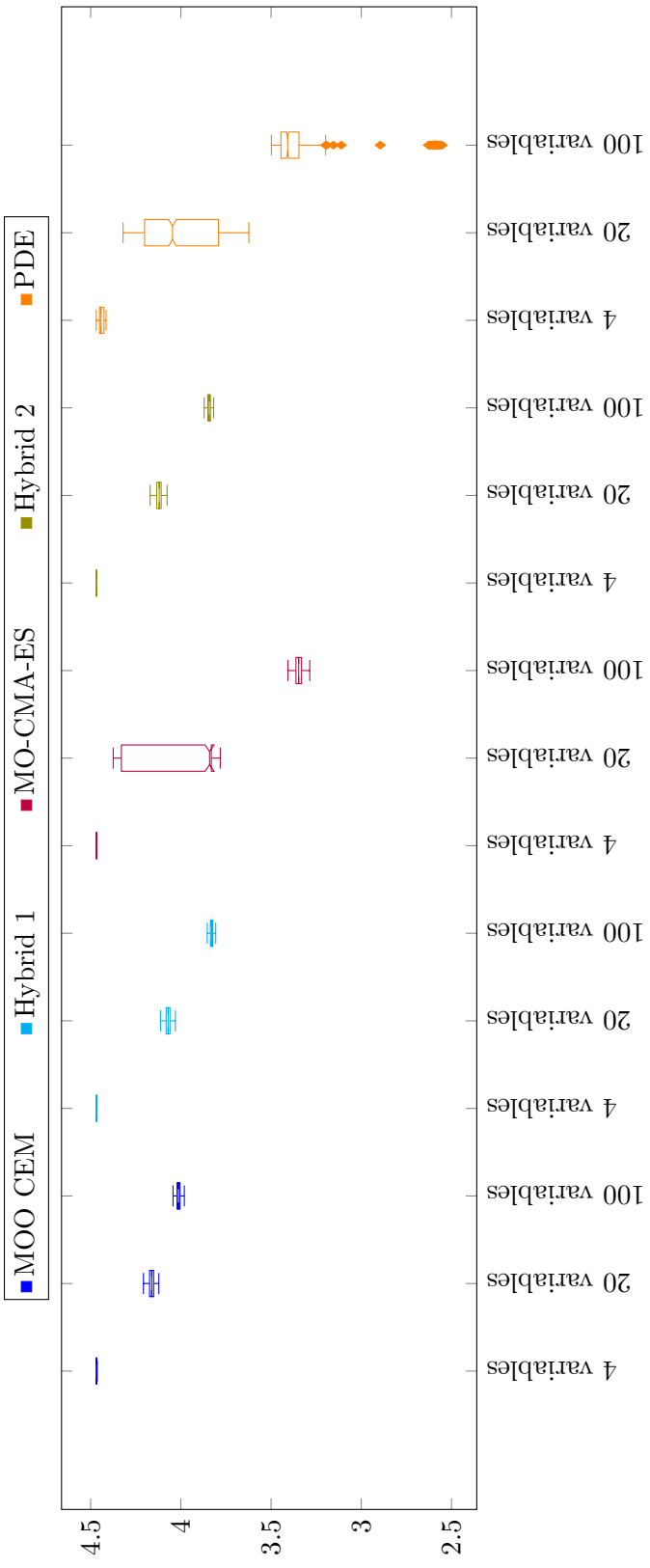


Figure A.66: Box plot of hypervolumes achieved when solving WFG 2 with two objectives.

Table A.44: Mann-Whitney U-test results for WFG 2 with 20 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	1	1	1	4	**1**
Hybrid 1	0	-	1	0	1	2	**3**
MO-CMA-ES	0	0	-	0	1	1	**4**
Hybrid 2	0	1	1	-	1	3	**2**
PDE	0	0	0	0	-	0	**5**

Table A.45: Mann-Whitney U-test results for WFG 2 with 100 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	1	1	1	4	**1**
Hybrid 1	0	-	1	0	1	2	**3**
MO-CMA-ES	0	0	-	0	0	0	**5**
Hybrid 2	0	1	1	-	1	3	**2**
PDE	0	0	1	0	-	1	**4**

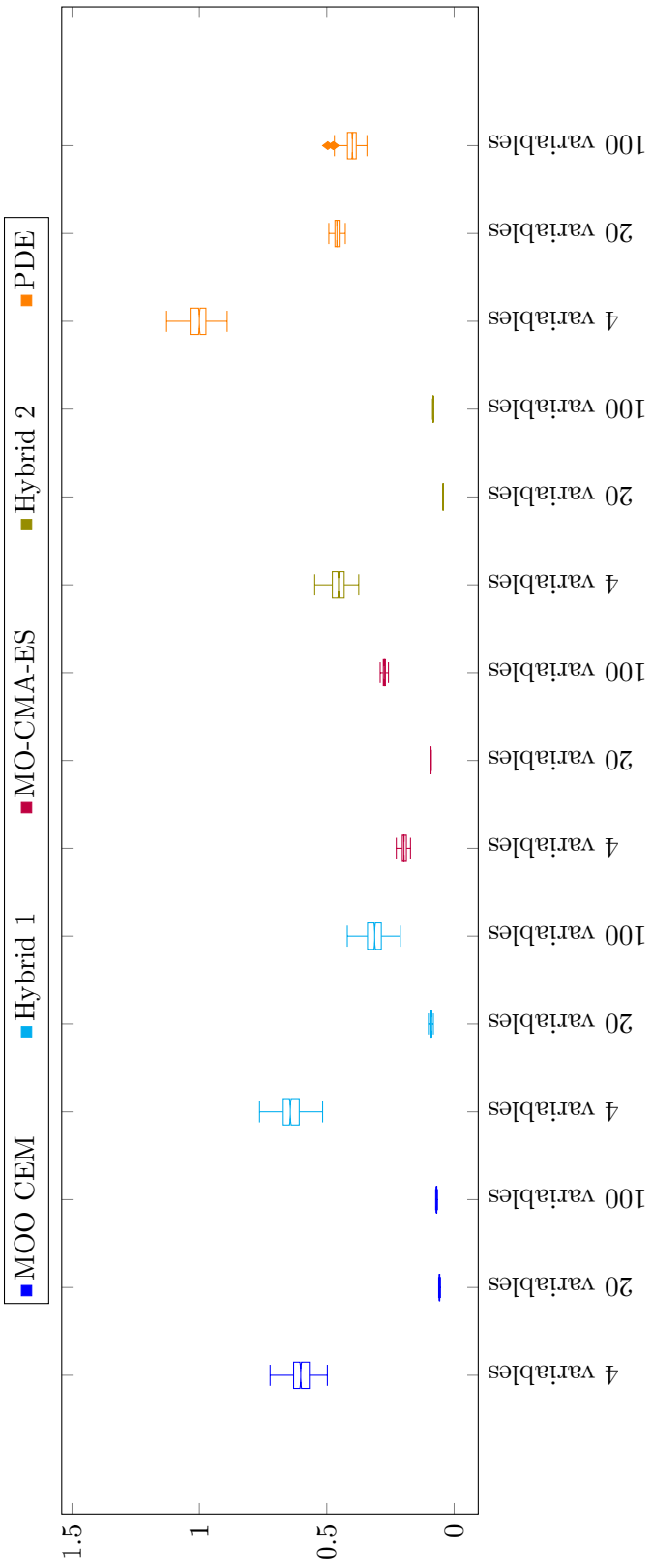


Figure A.67: Box plot of runtimes when solving WFG 2 with two objectives.

A.22 WFG 3

Table A.46: Problem details for WFG 3.

Number of variables	$V = 4$, 20 or 100
Box constraints	$0 \leq x_i \leq 2i, \quad i = 1, \ldots, V$
Geometry	Convex, linear, degenerate
Relationship	Reported relationships
Modality	Unimodal
Function definitions	See Table 4.9

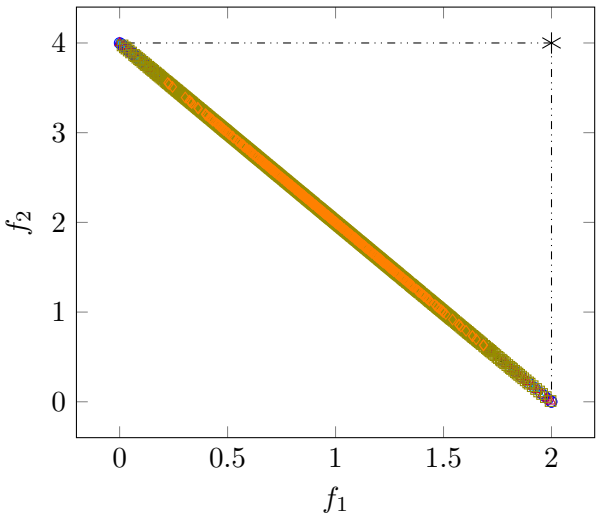


Figure A.68: A sample of Pareto fronts achieved by the algorithms on WFG 3 with four variables.

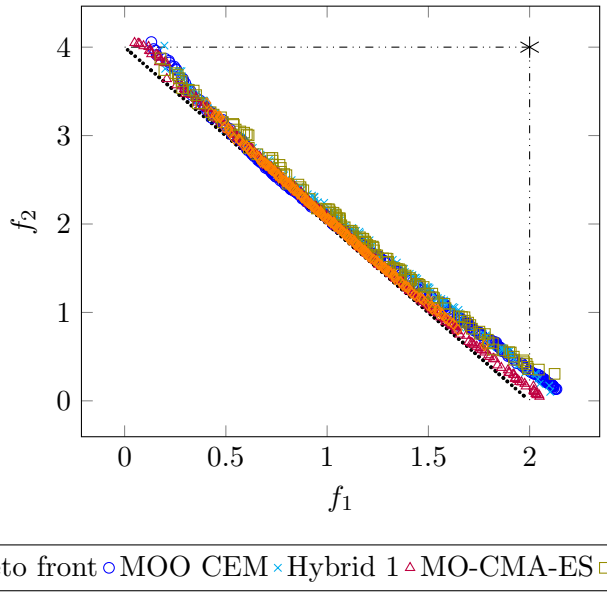


Figure A.69: A sample of Pareto fronts achieved by the algorithms on WFG 3 with 20 variables.

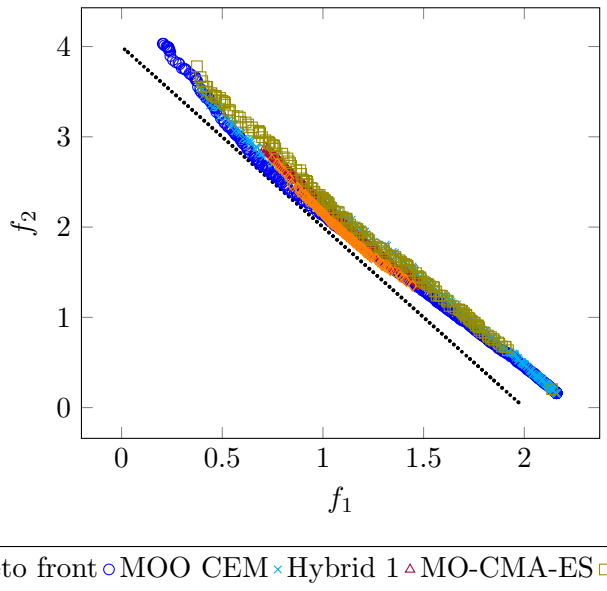


Figure A.70: A sample of Pareto fronts achieved by the algorithms on WFG 3 with 100 variables.

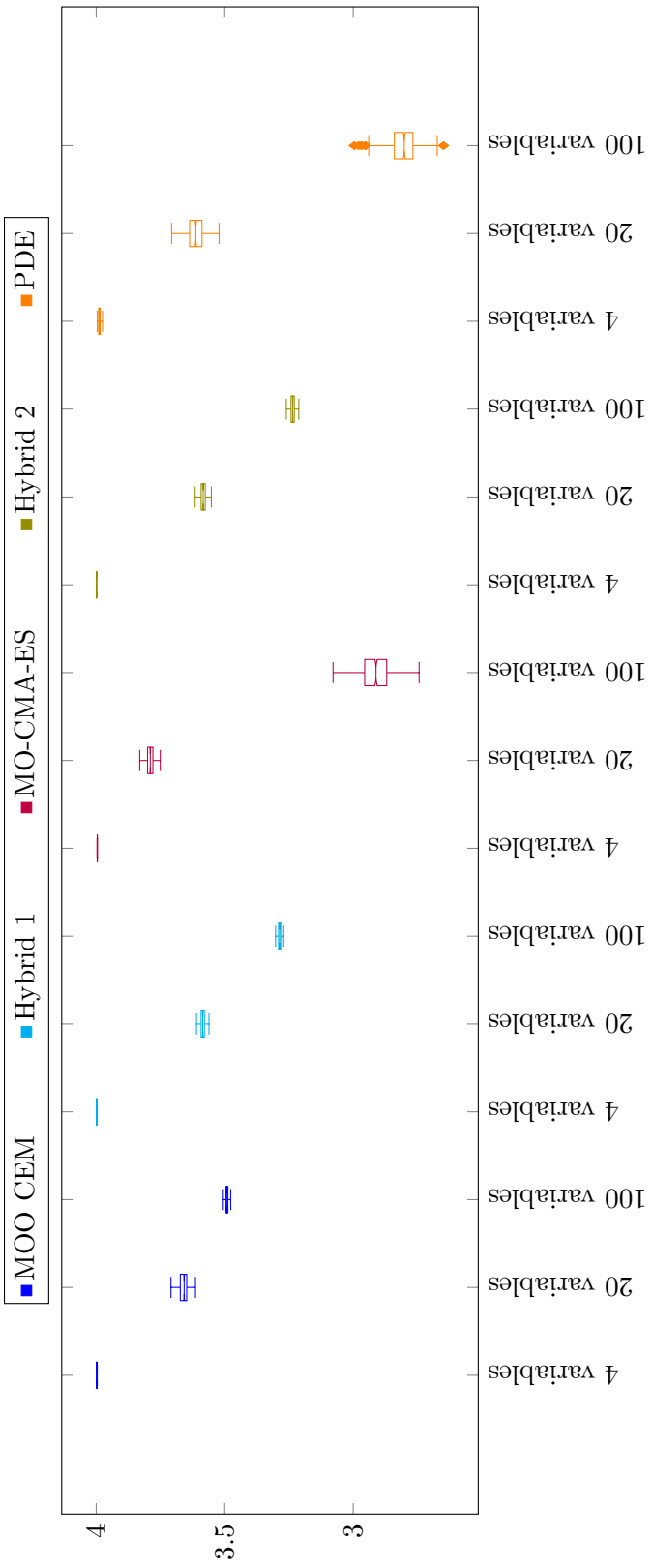


Figure A.71: Box plot of hypervolumes achieved when solving WFG 3 with two objectives.

Table A.47: Mann-Whitney U-test results for WFG 3 with four variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	1	0	1	2	**3**
Hybrid 1	1	-	1	0	1	3	**2**
MO-CMA-ES	0	0	-	0	1	1	**4**
Hybrid 2	1	1	1	-	1	4	**1**
PDE	0	0	0	0	-	0	**5**

Table A.48: Mann-Whitney U-test results for WFG 3 with 20 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	0	1	1	3	**2**
Hybrid 1	0	-	0	1	0	1	**4**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	0	1	0	1	-	2	**3**

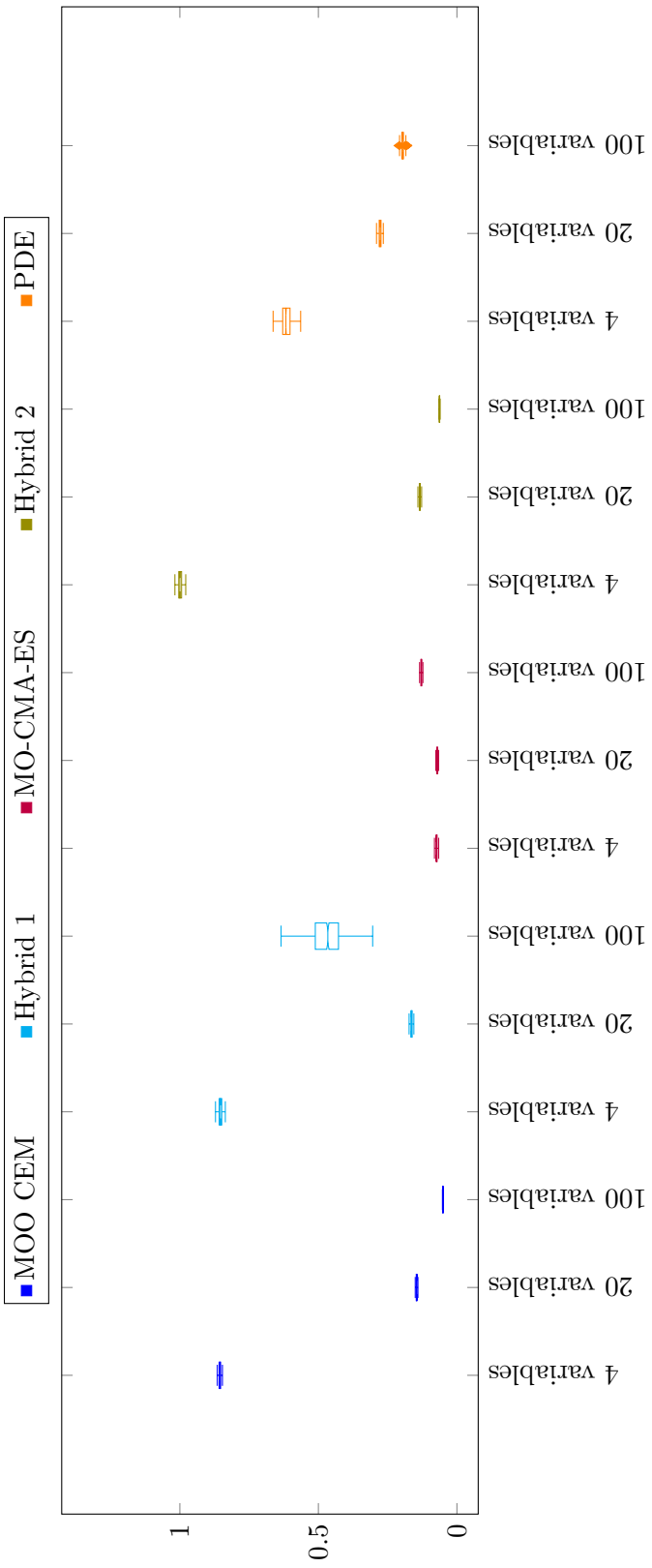


Figure A.72: Box plot of runtimes when solving WFG 3 with two objectives.

Table A.49: Mann-Whitney U-test results for WFG 3 with 100 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	1	0	2	**3**
MO-CMA-ES	1	1	-	1	0	3	**2**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	1	1	1	1	-	4	**1**

A.23 WFG 4

Table A.50: Problem details for WFG 4.

Number of variables	$V = 4$, 20 or 100
Box constraints	$0 \leq x_i \leq 2i, \quad i = 1, \ldots, V$
Geometry	Concave
Relationship	No reported relationships
Modality	Multimodal
Function definitions	See Table 4.9

Table A.51: Mann-Whitney U-test results for WFG 4 with four variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	0	0	1	**4**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	1	1	0	-	0	2	**3**
PDE	1	1	0	1	-	3	**2**

Table A.52: Mann-Whitney U-test results for WFG 4 with 20 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	0	0	1	**4**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	1	1	0	-	0	2	**3**
PDE	1	1	0	1	-	3	**2**

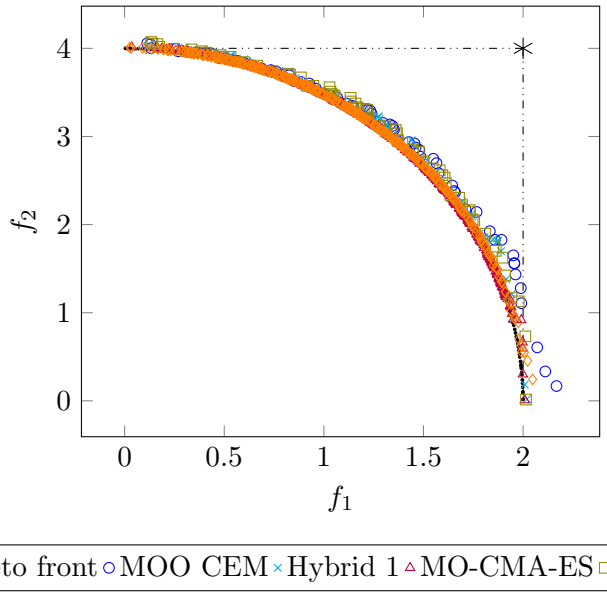


Figure A.73: A sample of Pareto fronts achieved by the algorithms on WFG 4 with four variables.

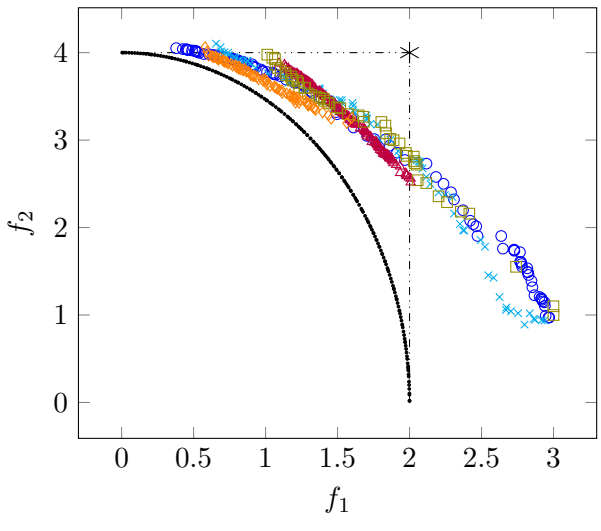


Figure A.74: A sample of Pareto fronts achieved by the algorithms on WFG 4 with 20 variables.

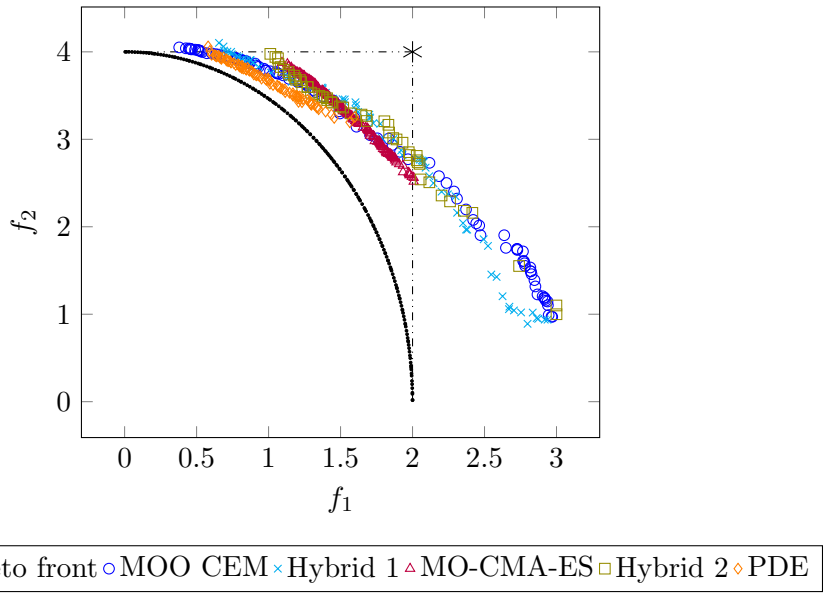


Figure A.75: A sample of Pareto fronts achieved by the algorithms on WFG 4 with 100 variables.

Table A.53: Mann-Whitney U-test results for WFG 4 with 100 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	1	0	2	**3**
MO-CMA-ES	1	1	-	1	0	3	**2**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	1	1	1	1	-	4	**1**

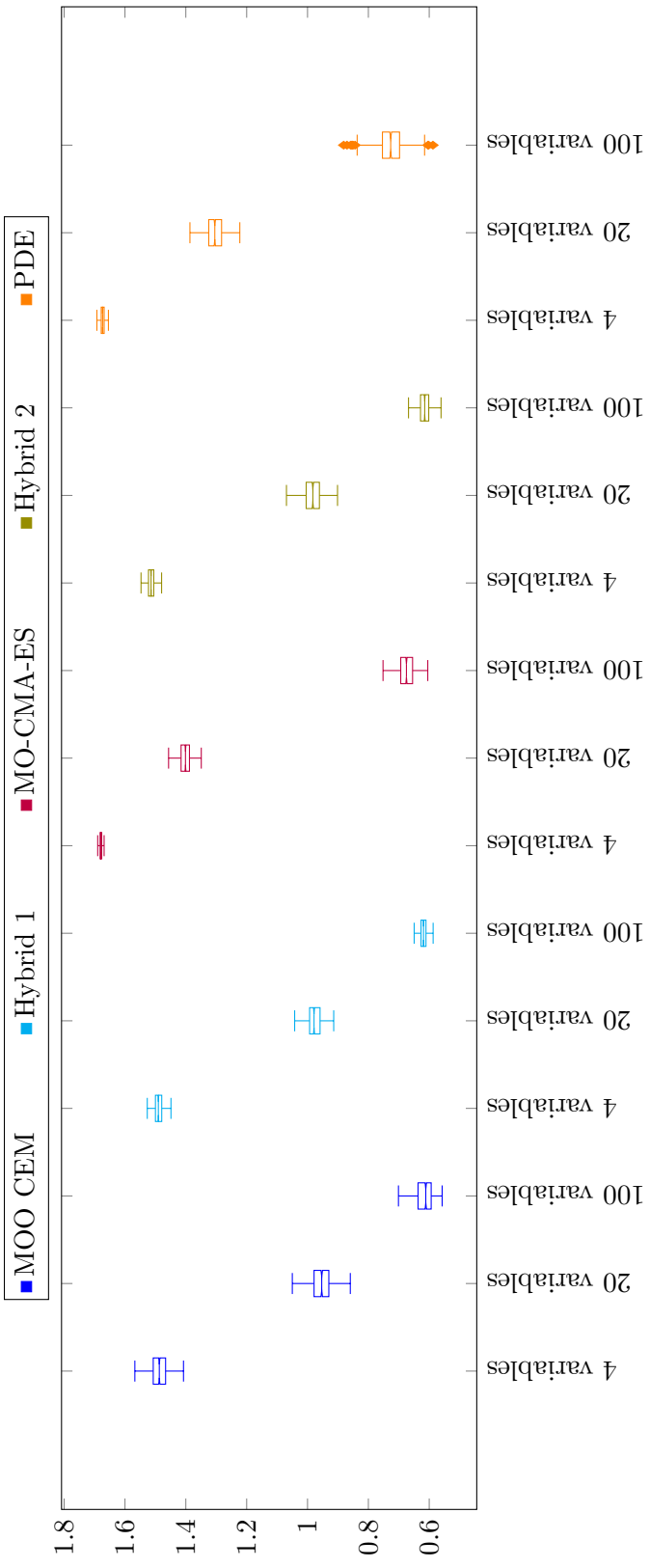


Figure A.76: Box plot of hypervolumes achieved when solving WFG 4 with two objectives.

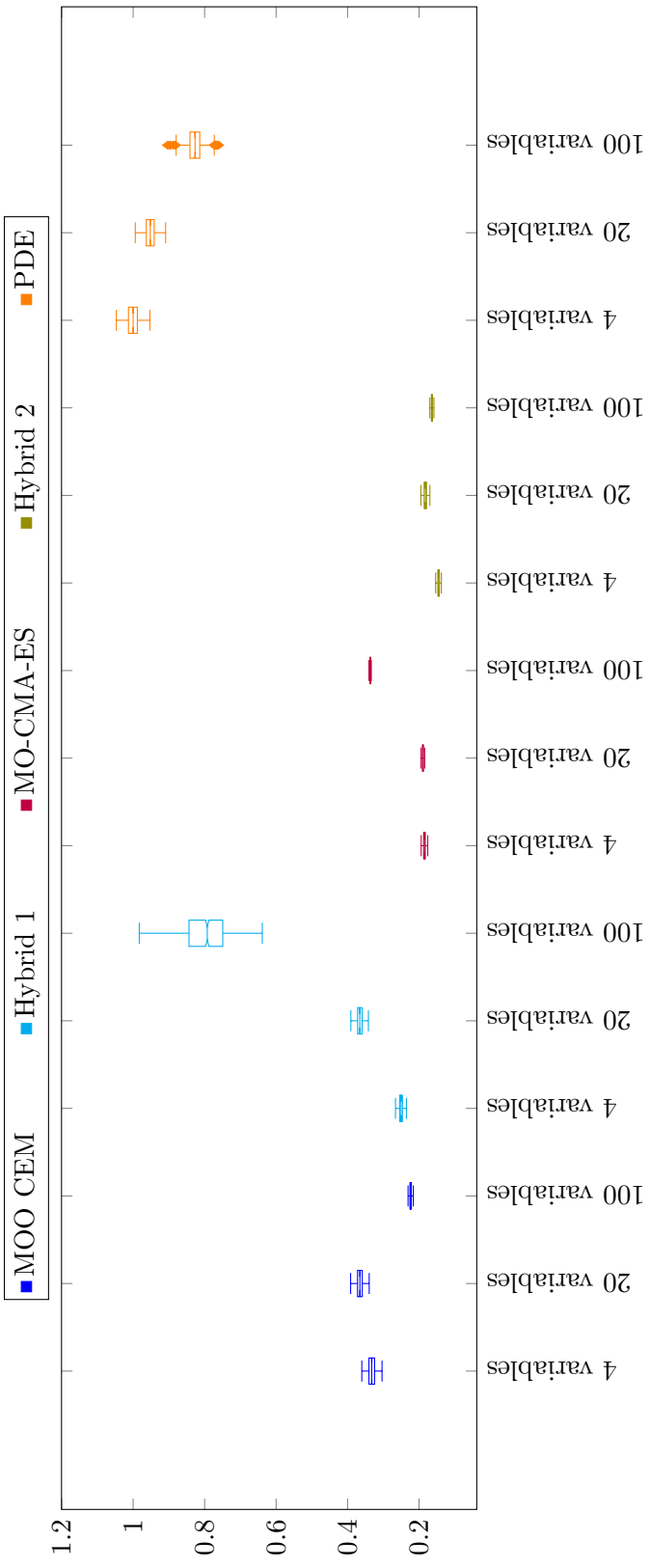


Figure A.77: Box plot of runtimes when solving WFG 4 with two objectives.

A.24 WFG 5

Table A.54: Problem details for WFG 5.

Number of variables	$V = 4$, 20 or 100
Box constraints	$0 \leq x_i \leq 2i, \quad i = 1, \ldots, V$
Geometry	Concave
Relationship	No reported relationships
Modality	Deceptive
Function definitions	See Table 4.9

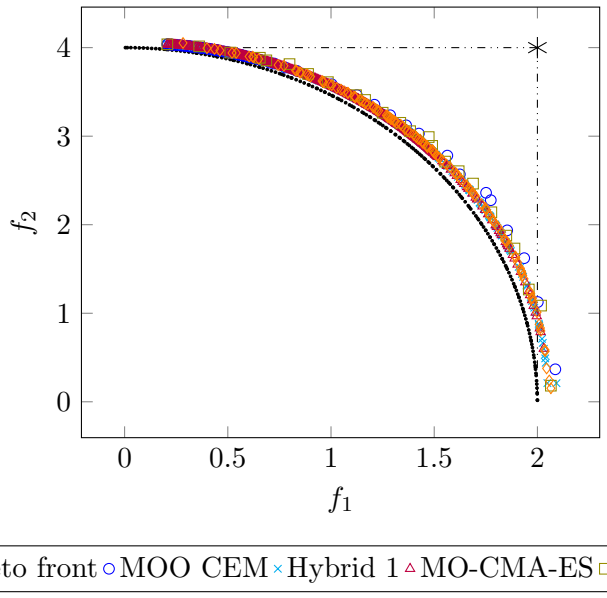


Figure A.78: A sample of Pareto fronts achieved by the algorithms on WFG 5 with four variables.

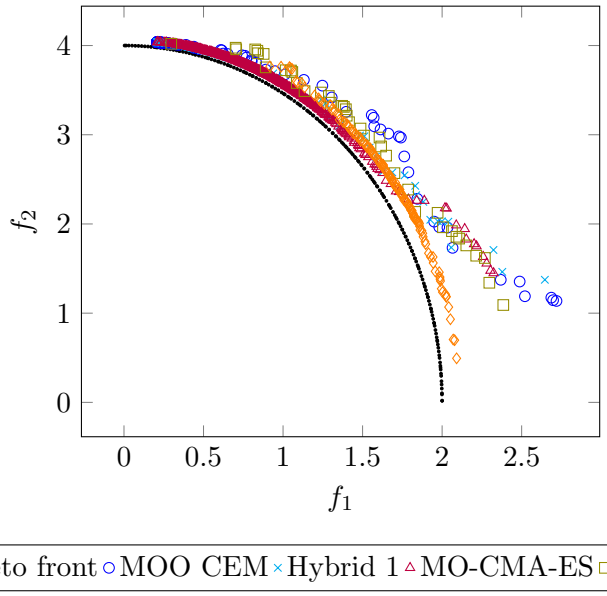


Figure A.79: A sample of Pareto fronts achieved by the algorithms on WFG 5 with 20 variables.

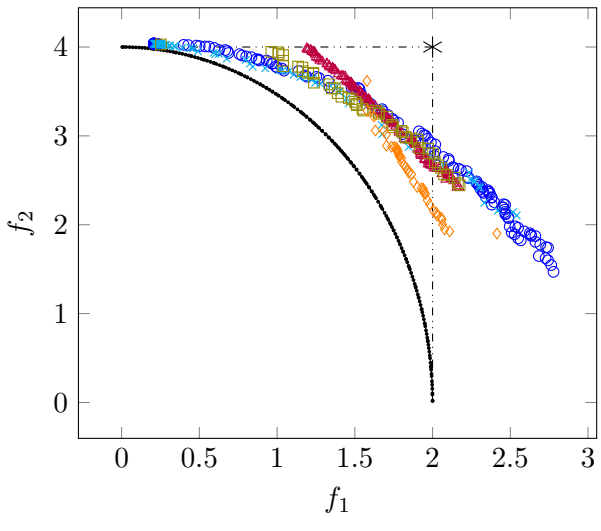


Figure A.80: A sample of Pareto fronts achieved by the algorithms on WFG 5 with 100 variables.

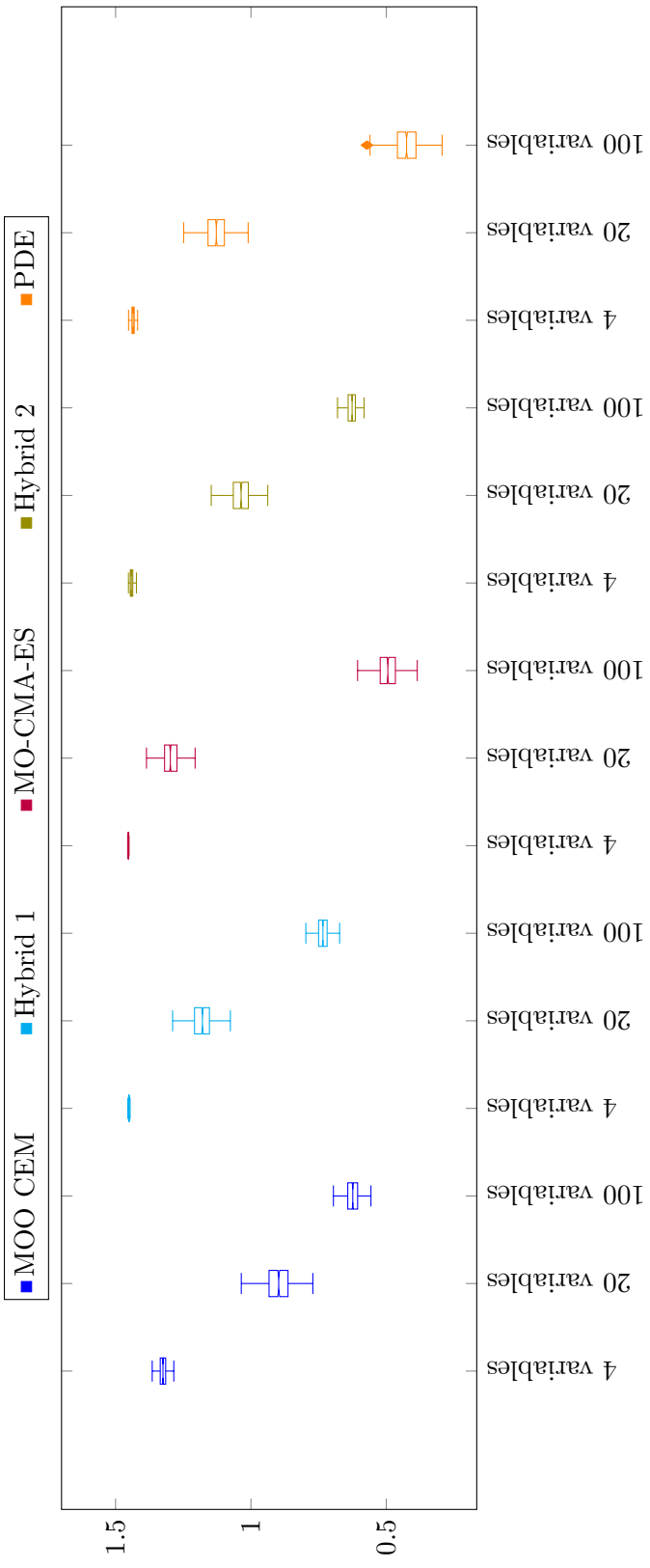


Figure A.81: Box plot of hypervolumes achieved when solving WFG 5 with two objectives.

Table A.55: Mann-Whitney U-test results for WFG 5 with four variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	1	1	3	**2**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	1	0	0	-	1	2	**3**
PDE	1	0	0	0	-	1	**4**

Table A.56: Mann-Whitney U-test results for WFG 5 with 20 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	1	1	3	**2**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	1	0	0	-	0	1	**4**
PDE	1	0	0	1	-	2	**3**

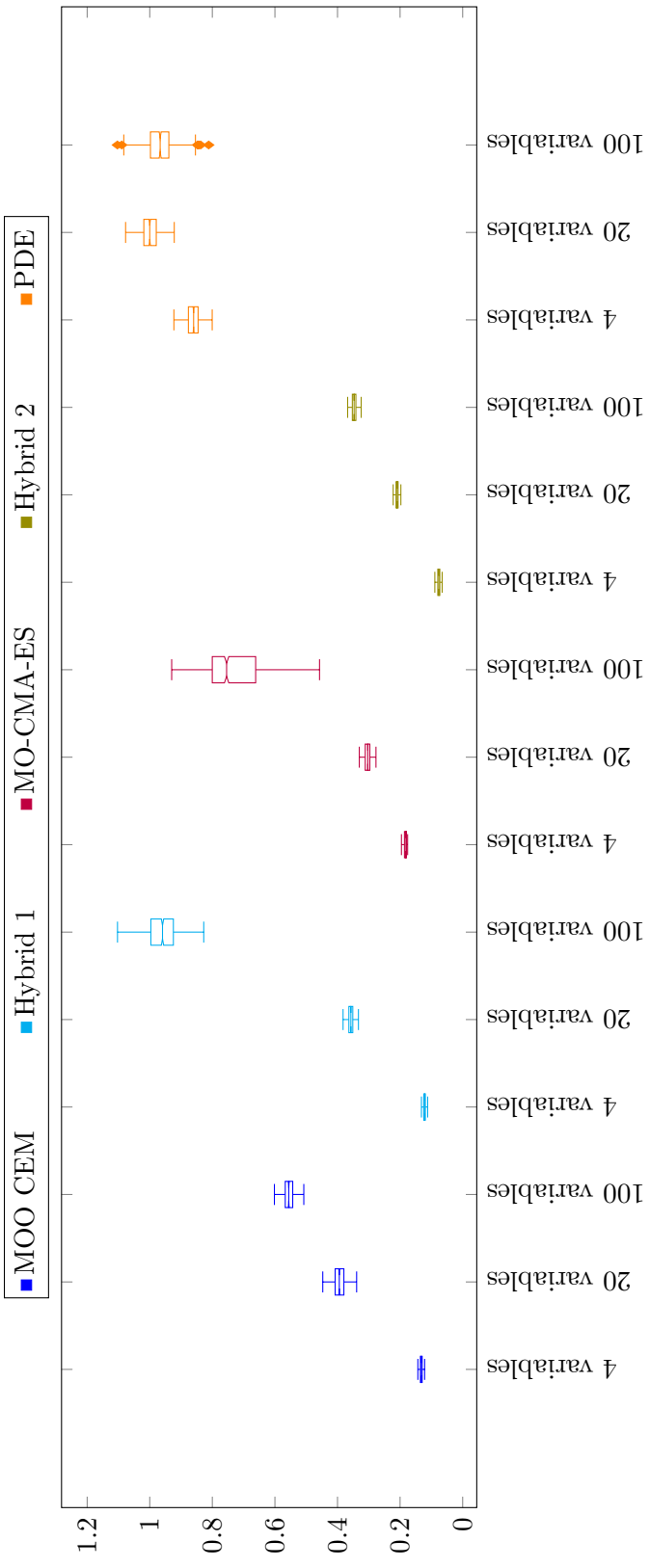


Figure A.82: Box plot of runtimes when solving WFG 5 with two objectives.

Table A.57: Mann-Whitney U-test results for WFG 5 with 100 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	1	0	1	2	**3**
Hybrid 1	1	-	1	1	1	4	**1**
MO-CMA-ES	0	0	-	0	1	1	**4**
Hybrid 2	1	0	1	-	1	3	**2**
PDE	0	0	0	0	-	0	**5**

A.25 WFG 6

Table A.58: Problem details for WFG 6.

Number of variables	$V = 4$, 20 or 100
Box constraints	$0 \leq x_i \leq 2i, \quad i = 1, \ldots, V$
Geometry	Concave
Relationship	Reported relationships
Modality	Unimodal
Function definitions	See Table 4.9

Table A.59: Mann-Whitney U-test results for WFG 6 with four variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	0	0	1	**4**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	1	0	0	-	0	1	**4**
PDE	1	1	0	1	-	3	**2**

Table A.60: Mann-Whitney U-test results for WFG 6 with 20 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	1	1	3	**2**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	1	0	0	-	0	1	**4**
PDE	1	0	0	1	-	2	**3**

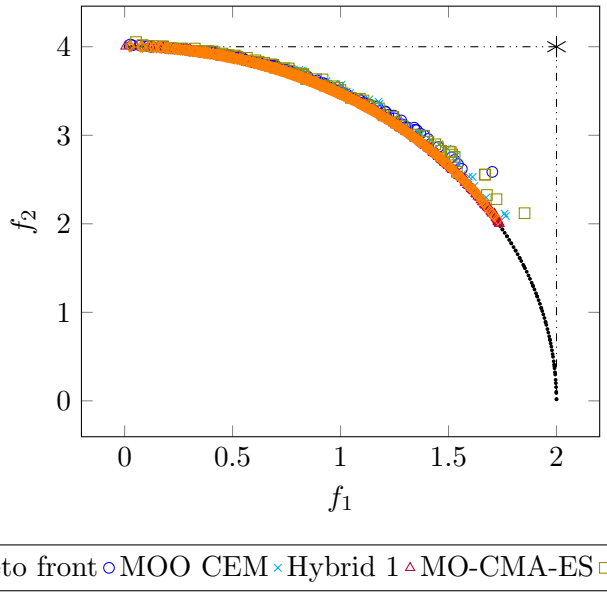


Figure A.83: A sample of Pareto fronts achieved by the algorithms on WFG 6 with four variables.

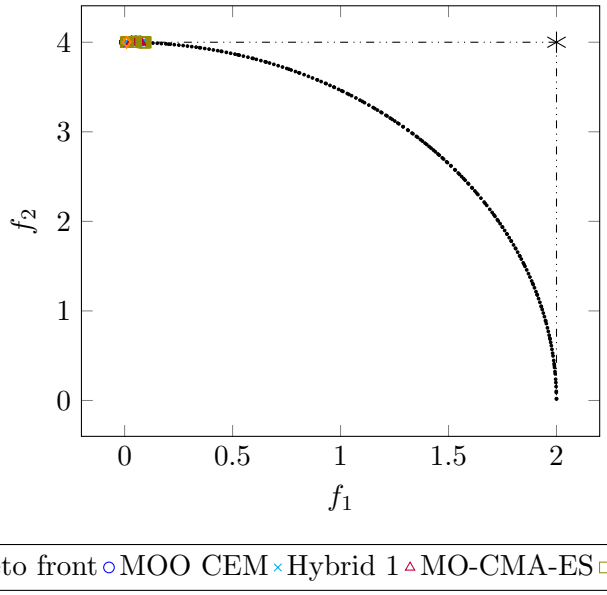


Figure A.84: A sample of Pareto fronts achieved by the algorithms on WFG 6 with 20 variables.

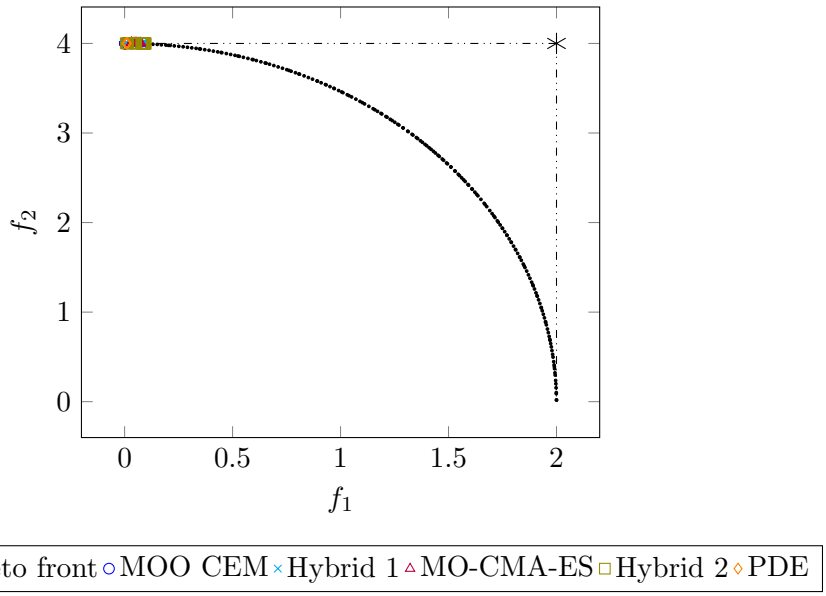


Figure A.85: A sample of Pareto fronts achieved by the algorithms on WFG 6 with 100 variables.

Table A.61: Mann-Whitney U-test results for WFG 6 with 100 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	1	0	0	1	**4**
Hybrid 1	1	-	1	1	1	4	**1**
MO-CMA-ES	0	0	-	0	0	0	**5**
Hybrid 2	1	0	1	-	1	3	**2**
PDE	1	0	1	0	-	2	**3**

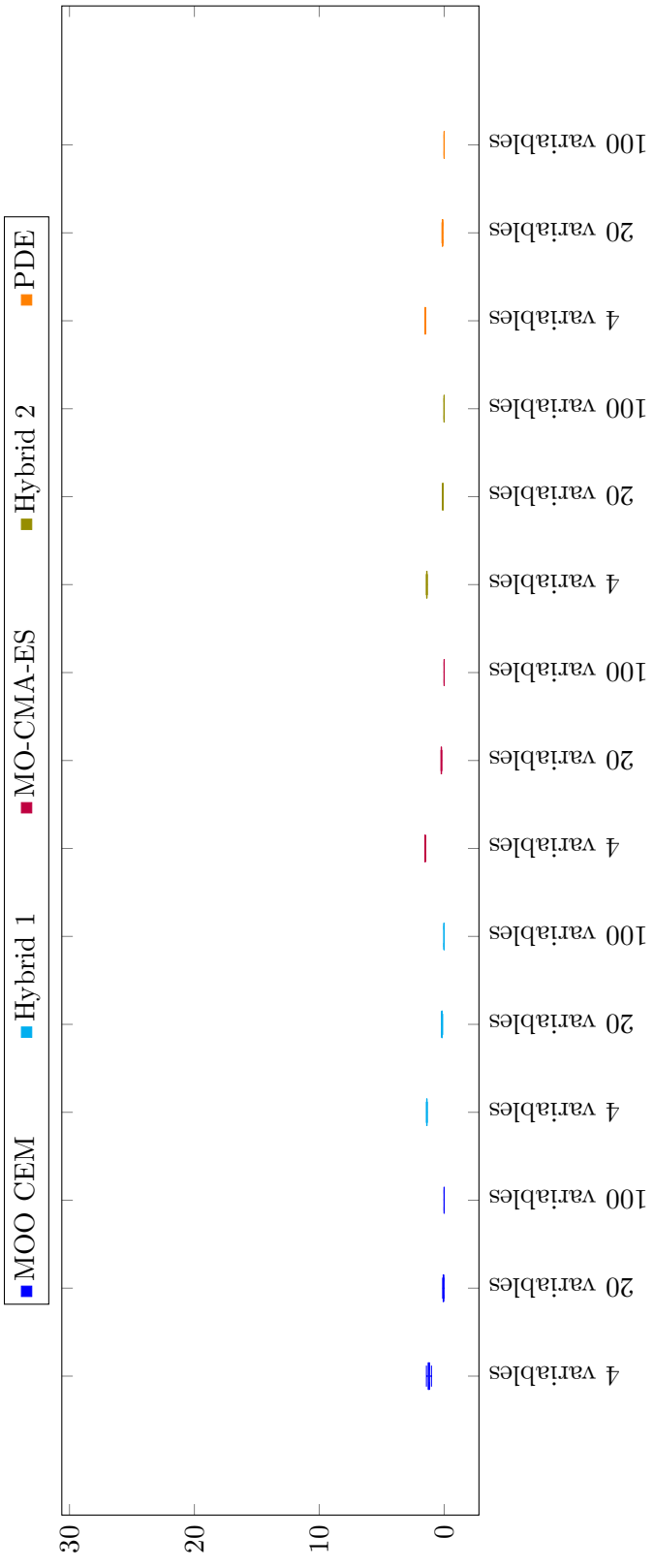


Figure A.86: Box plot of hypervolumes achieved when solving WFG 6 with two objectives.

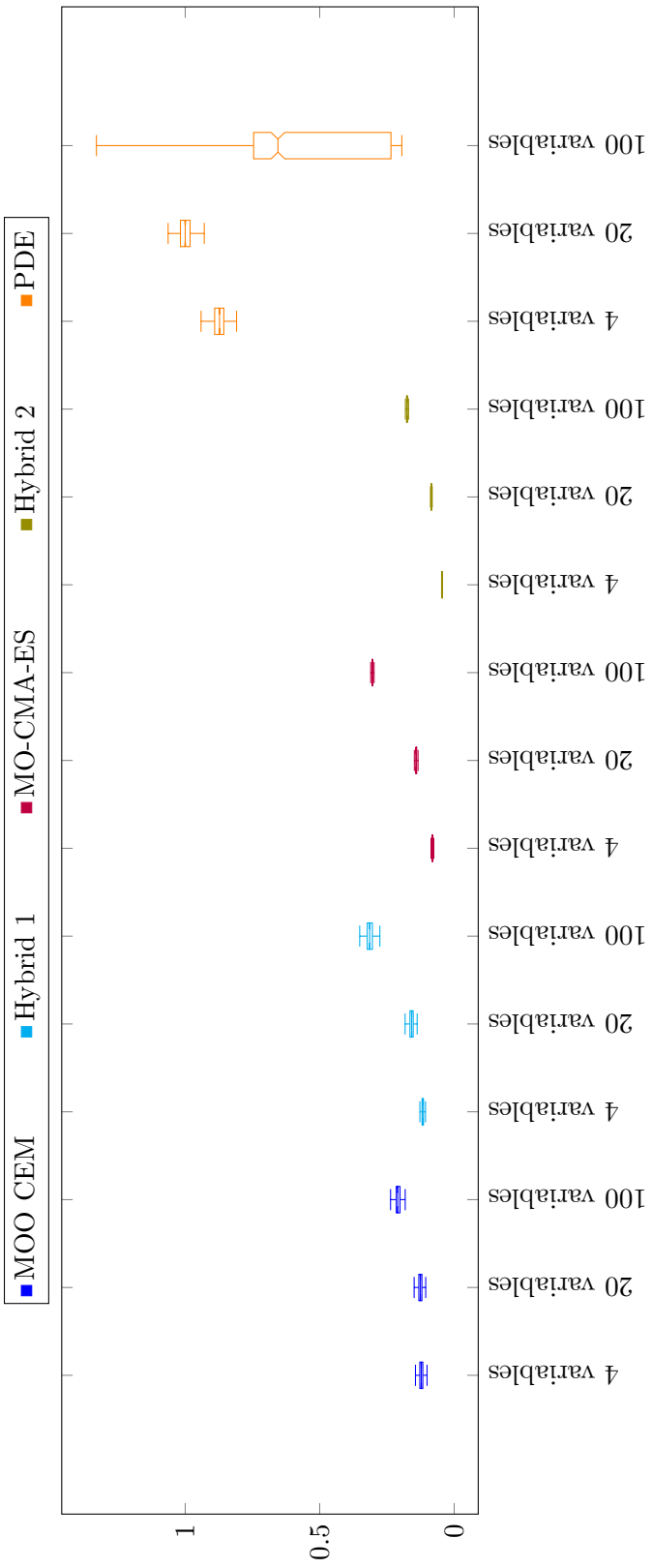


Figure A.87: Box plot of runtimes when solving WFG 6 with two objectives.

A.26 WFG 7

Table A.62: Problem details for WFG 7.

Number of variables	$V = 4$, 20 or 100
Box constraints	$0 \leq x_i \leq 2i, \quad i = 1, \ldots, V$
Geometry	Concave
Relationship	No reported relationships
Modality	Unimodal
Function definitions	See Table 4.9

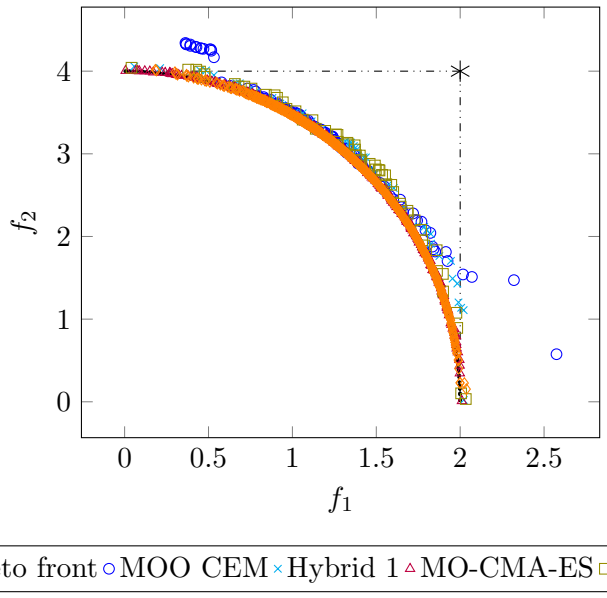


Figure A.88: A sample of Pareto fronts achieved by the algorithms on WFG 7 with four variables.

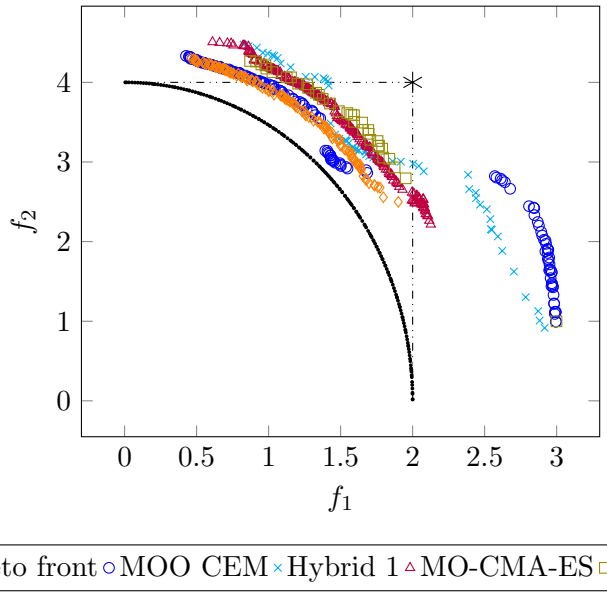


Figure A.89: A sample of Pareto fronts achieved by the algorithms on WFG 7 with 20 variables.

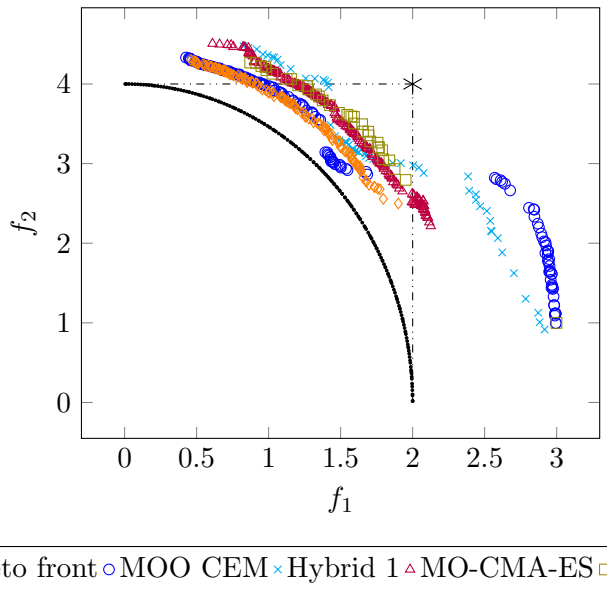


Figure A.90: A sample of Pareto fronts achieved by the algorithms on WFG 7 with 100 variables.

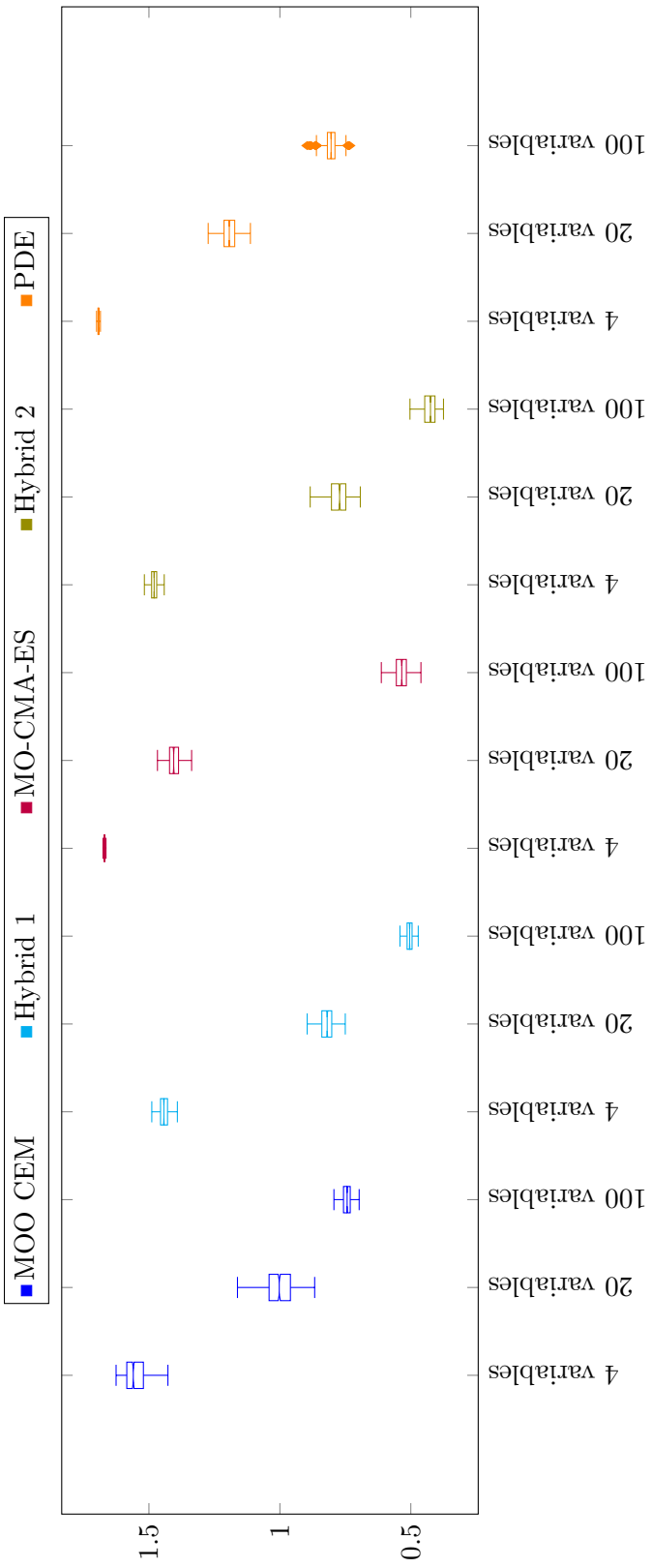


Figure A.91: Box plot of hypervolumes achieved when solving WFG 7 with two objectives.

Table A.63: Mann-Whitney U-test results for WFG 7 with four variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	0	1	0	2	**3**
Hybrid 1	0	-	0	0	0	0	**5**
MO-CMA-ES	1	1	-	1	0	3	**2**
Hybrid 2	0	1	0	-	0	1	**4**
PDE	1	1	1	1	-	4	**1**

Table A.64: Mann-Whitney U-test results for WFG 7 with 100 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	0	1	0	2	**3**
Hybrid 1	0	-	0	1	0	1	**4**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	1	1	0	1	-	3	**2**

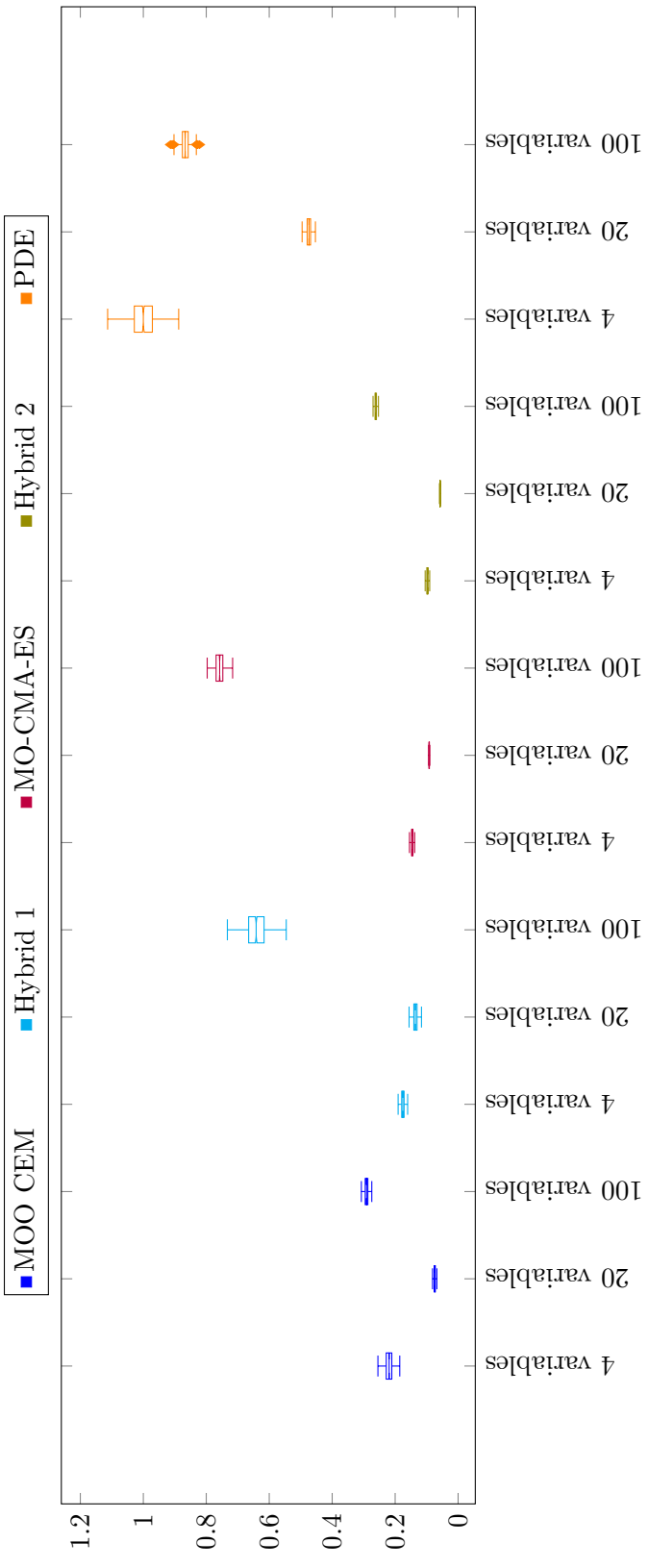


Figure A.92: Box plot of runtimes when solving WFG 7 with two objectives.

Table A.65: Mann-Whitney U-test results for WFG 7 with 100 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	1	1	0	3	**2**
Hybrid 1	0	-	0	1	0	1	**4**
MO-CMA-ES	0	1	-	1	0	2	**3**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	1	1	1	1	-	4	**1**

A.27 WFG 8

Table A.66: Problem details for WFG 8.

Number of variables	$V = 4$, 20 or 100
Box constraints	$0 \leq x_i \leq 2i, \quad i = 1, \ldots, V$
Geometry	Concave
Relationship	Reported relationships
Modality	Unimodal
Function definitions	See Table 4.9

Table A.67: Mann-Whitney U-test results for WFG 8 with four variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	0	0	0	1	**4**
Hybrid 1	0	-	0	0	0	0	**5**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	1	1	0	-	0	2	**3**
PDE	1	1	0	1	-	3	**2**

Table A.68: Mann-Whitney U-test results for WFG 8 with 20 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	0	1	0	2	**3**
Hybrid 1	0	-	0	1	0	1	**4**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	1	1	0	1	-	3	**2**

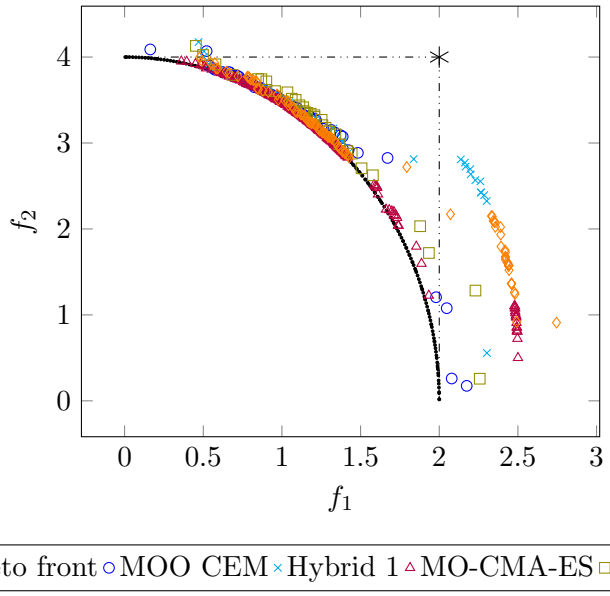


Figure A.93: A sample of Pareto fronts achieved by the algorithms on WFG 8 with four variables.

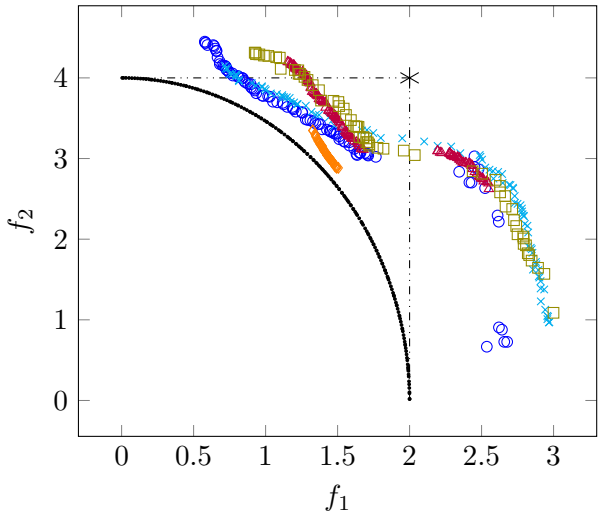


Figure A.94: A sample of Pareto fronts achieved by the algorithms on WFG 8 with 20 variables.

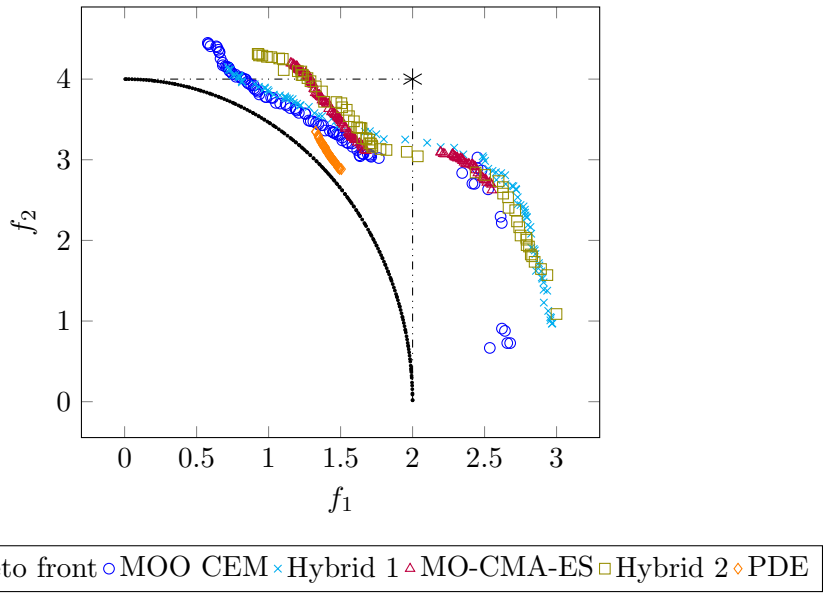


Figure A.95: A sample of Pareto fronts achieved by the algorithms on WFG 8 with 100 variables.

Table A.69: Mann-Whitney U-test results for WFG 8 with 100 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	1	1	0	3	**2**
Hybrid 1	0	-	1	1	0	2	**3**
MO-CMA-ES	0	0	-	1	0	1	**4**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	1	1	1	1	-	4	**1**

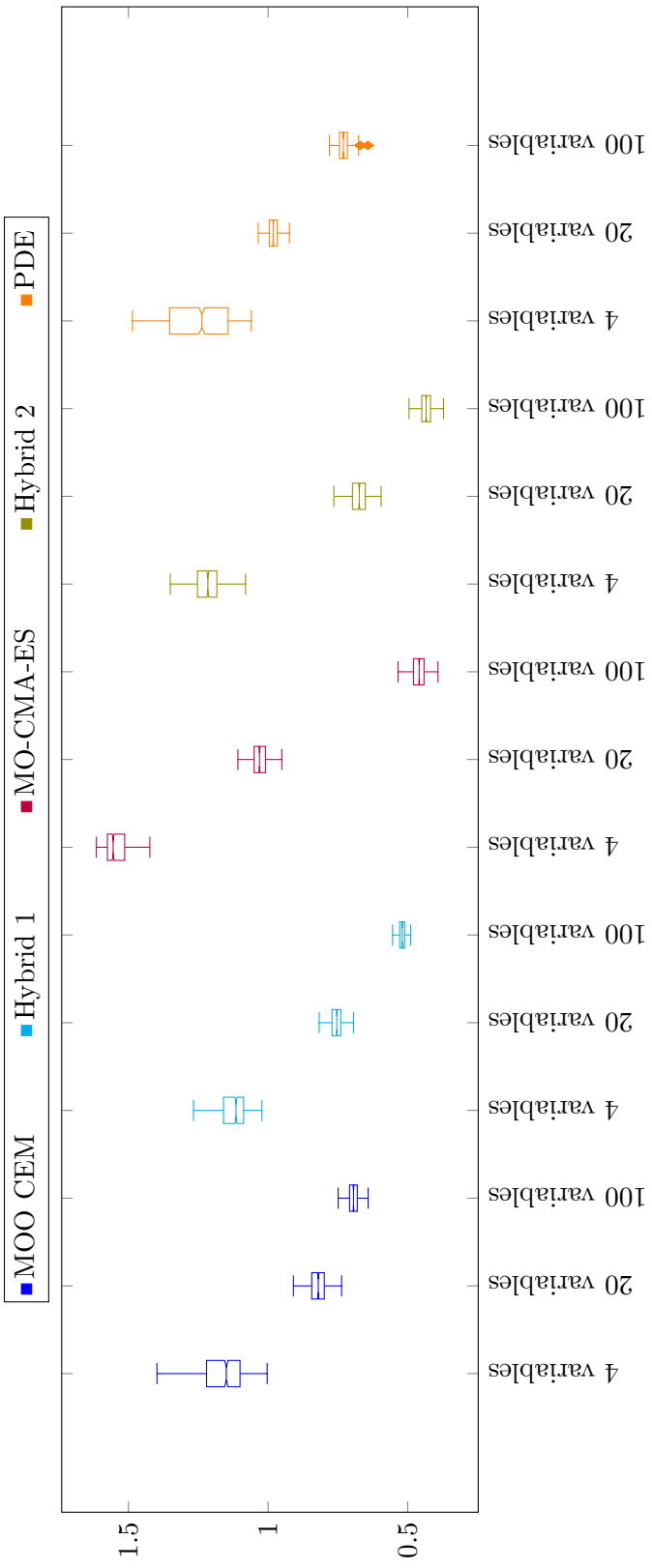


Figure A.96: Box plot of hypervolumes achieved when solving WFG 8 with two objectives.

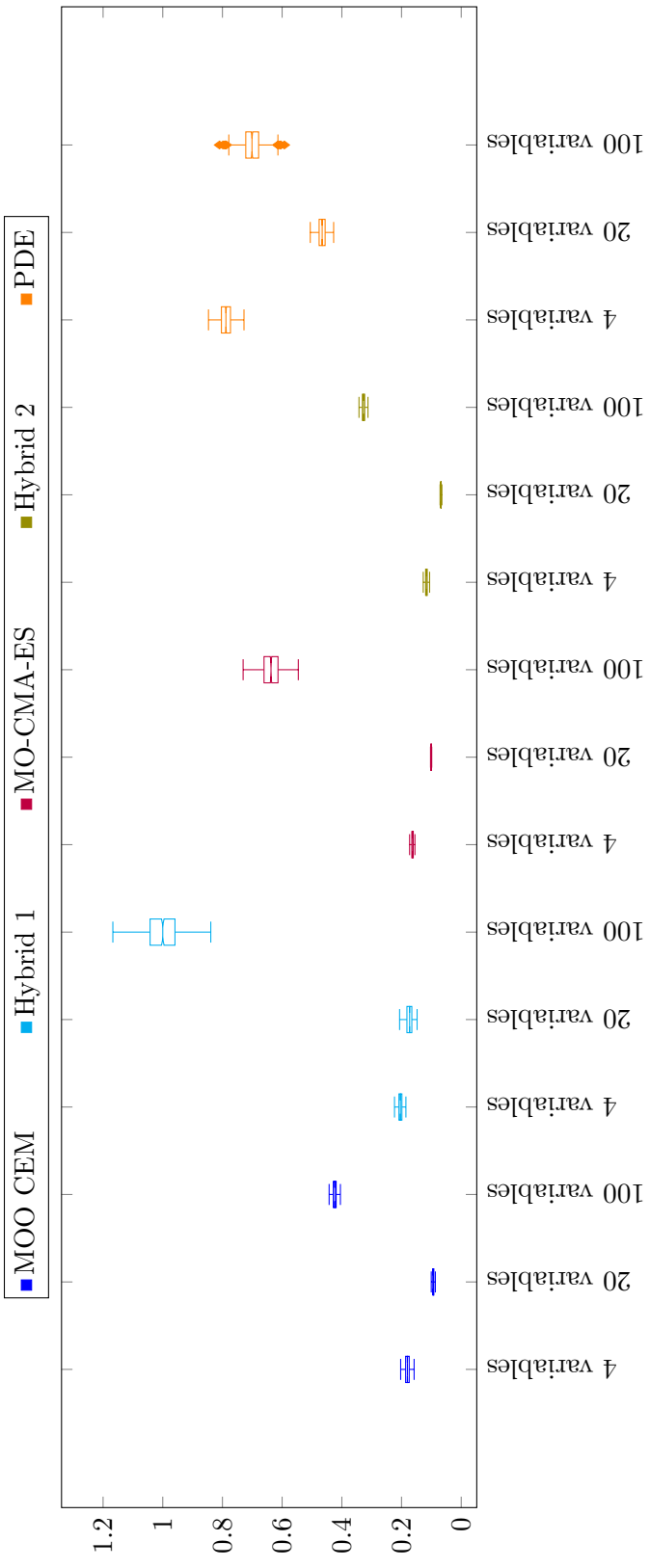


Figure A.97: Box plot of runtimes when solving WFG 8 with two objectives.

A.28 WFG 9

Table A.70: Problem details for WFG 9.

Number of variables	$V = 4$, 20 or 100
Box constraints	$0 \leq x_i \leq 2i, \quad i = 1, \ldots, V$
Geometry	Concave
Relationship	Reported relationships
Modality	Multimodal, deceptive
Function definitions	See Table 4.9

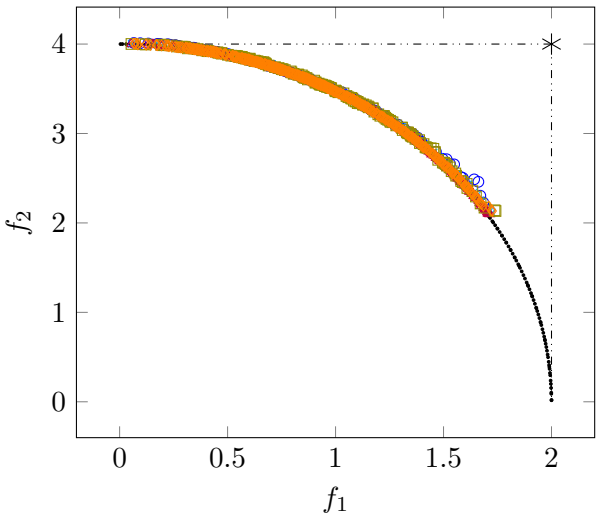


Figure A.98: A sample of Pareto fronts achieved by the algorithms on WFG 9 with four variables.

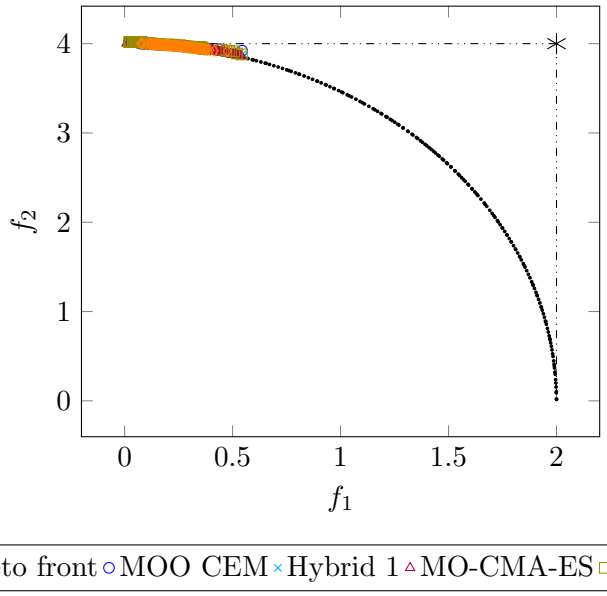


Figure A.99: A sample of Pareto fronts achieved by the algorithms on WFG 9 with 20 variables.

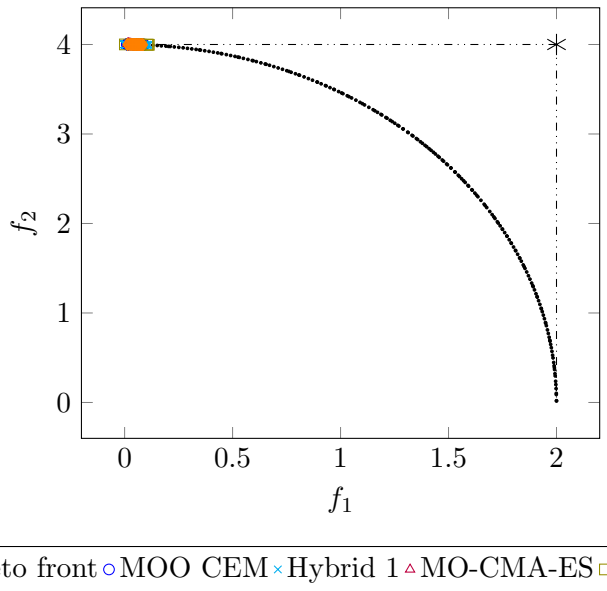


Figure A.100: A sample of Pareto fronts achieved by the algorithms on WFG 9 with 100 variables.

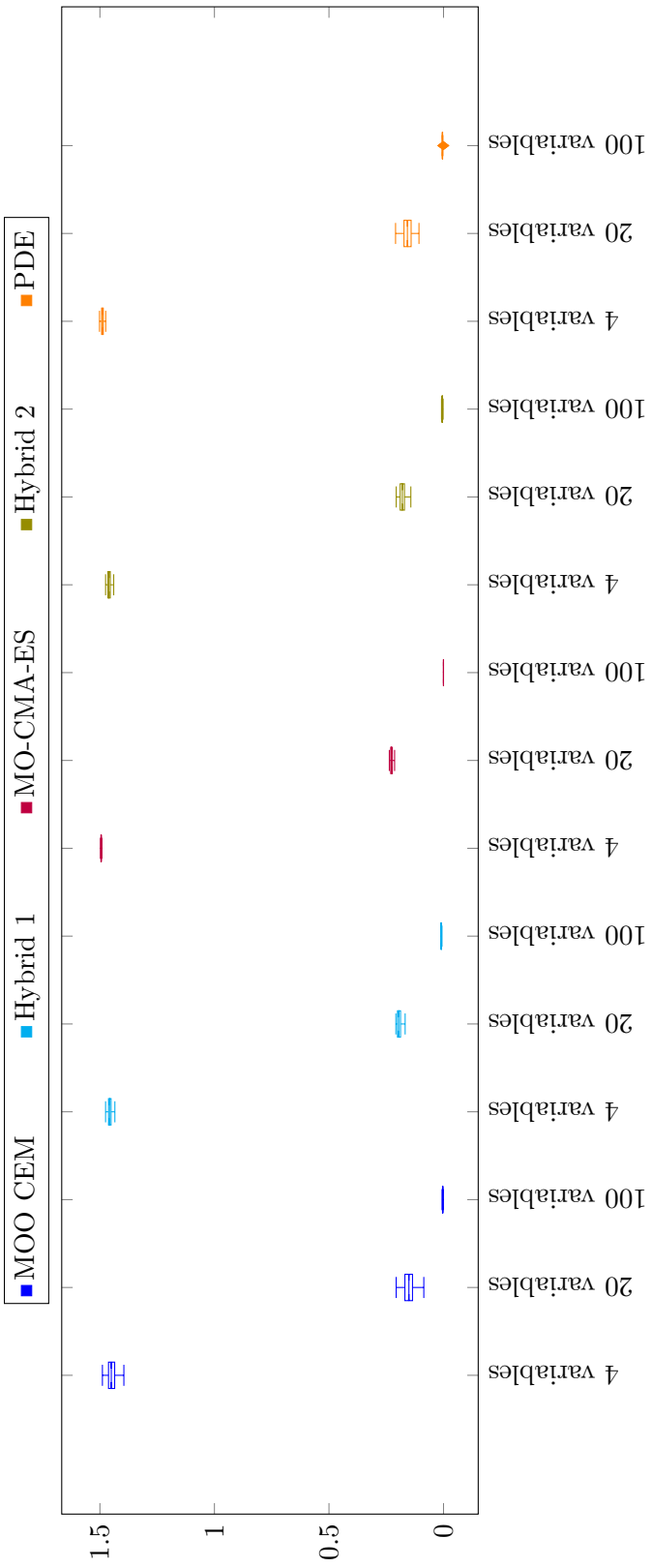


Figure A.101: Box plot of hypervolumes achieved when solving WFG 9 with two objectives.

Table A.71: Mann-Whitney U-test results for WFG 9 with four variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	0	0	1	**4**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	1	1	0	-	0	2	**3**
PDE	1	1	0	1	-	3	**2**

Table A.72: Mann-Whitney U-test results for WFG 9 with 20 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	1	1	3	**2**
MO-CMA-ES	1	1	-	1	1	4	**1**
Hybrid 2	1	0	0	-	1	2	**3**
PDE	1	0	0	0	-	1	**4**

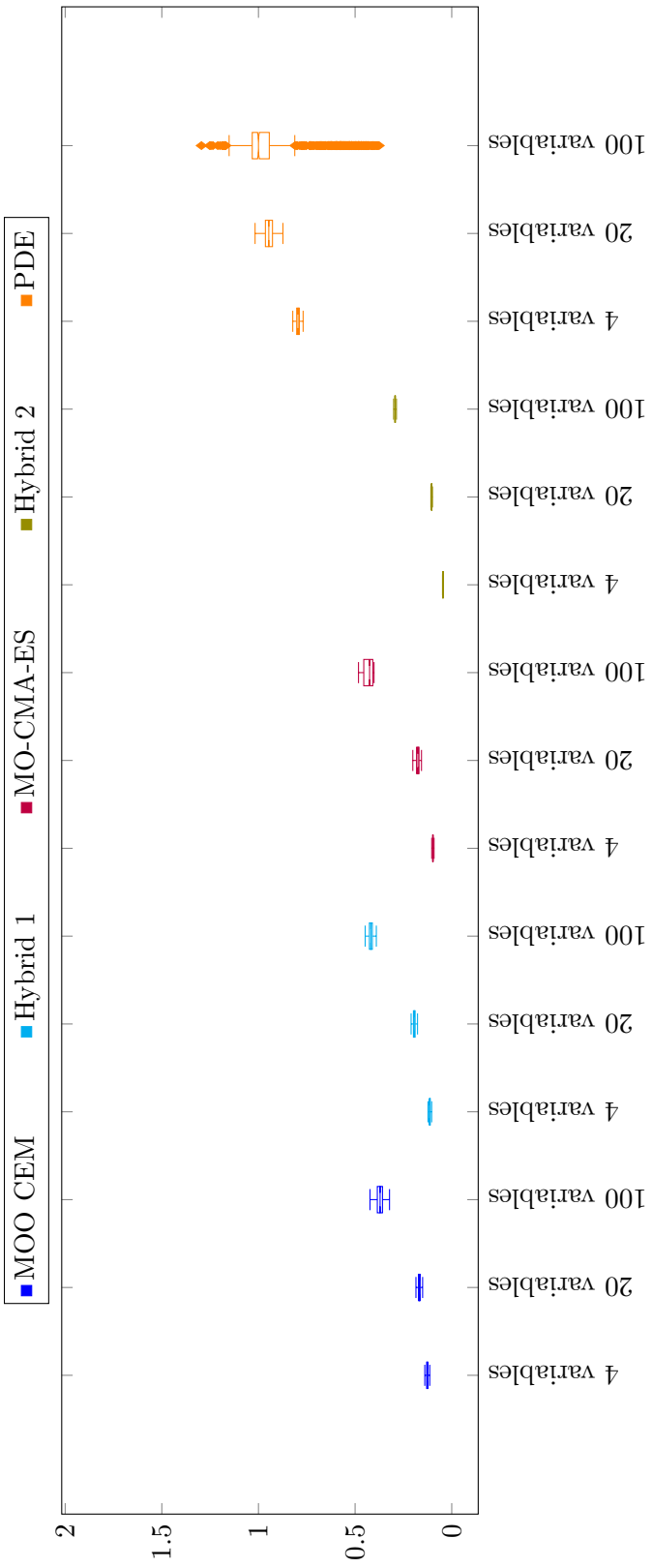


Figure A.102: Box plot of relative run times when solving WFG 9 with two objectives.

Table A.73: Mann-Whitney U-test results for WFG 9 with 100 variables.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	1	0	0	1	**4**
Hybrid 1	1	-	1	1	1	4	**1**
MO-CMA-ES	0	0	-	0	0	0	**5**
Hybrid 2	1	0	1	-	1	3	**2**
PDE	1	0	1	0	-	2	**3**

Appendix B

Results for the MRR cases

This appendix presents the results for the mission-ready resource (MRR) cases. For each case, a sample of the Pareto fronts achieved is presented. Two sets of box plots are also shown: a summary of the hypervolumes achieved and a summary of the relative run times of the algorithms. The results of the Mann-Whitney U-tests as performed on the achieved hypervolumes are presented for all the cases. For the Mann-Whitney U-test results, if a matrix entry $ij = 1$, it indicates that Algorithm i achieved a significantly higher hypervolume than Algorithm j at a 5% significance level. The *Outperformed* column sums the total number of algorithms that Algorithm i outperformed, whereas the *Rank* column indicates which algorithm performed best on the problem at hand (with 1 being the best and 5 being the worst).

B.1 MRR Case A

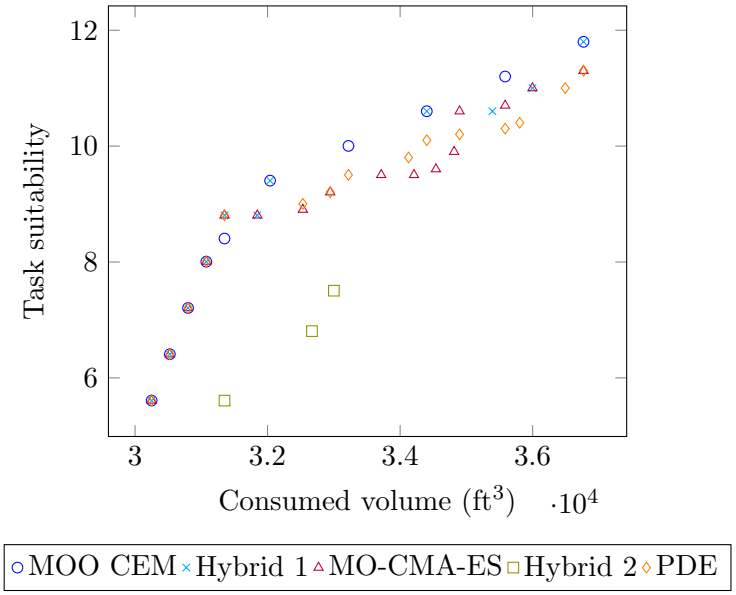


Figure B.1: A sample of Pareto fronts achieved by the algorithms on MRR Case A

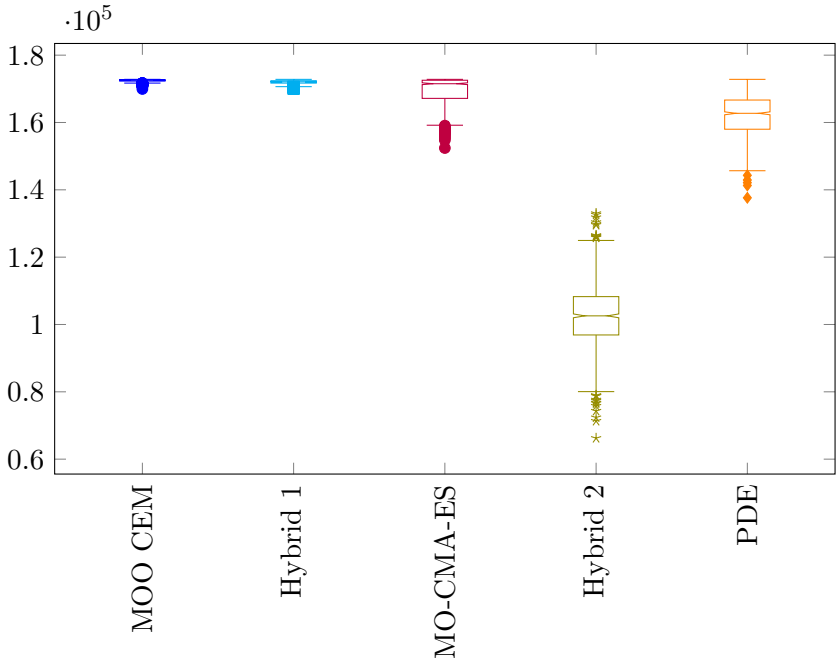


Figure B.2: Box plot of hypervolumes achieved when solving MRR Case A.

Table B.1: Mann-Whitney U-test results for MRR Case A.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	1	1	1	4	**1**
Hybrid 1	0	-	1	1	1	3	**2**
MO-CMA-ES	0	0	-	1	1	2	**3**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	0	0	0	1	-	1	**4**

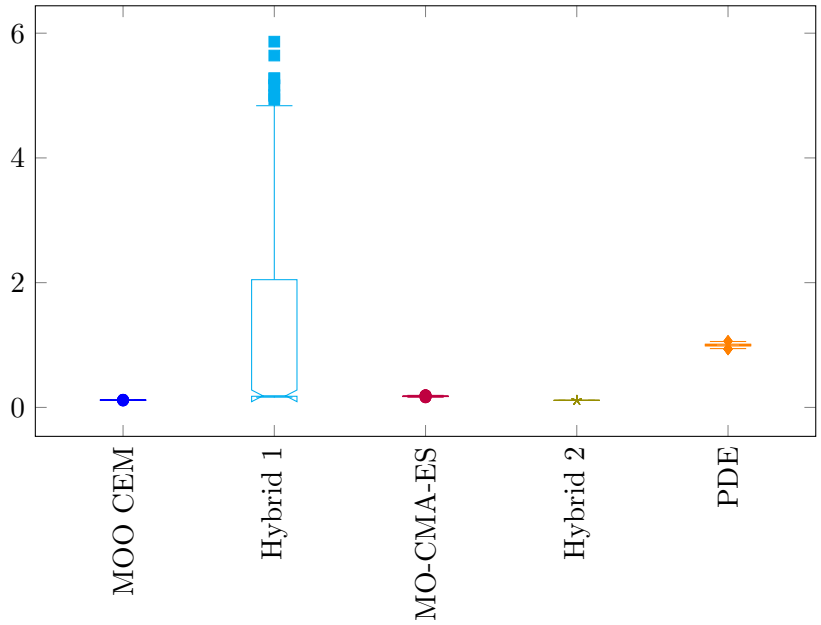


Figure B.3: Box plot of relative run times when solving MRR Case A.

B.2 MRR Case B

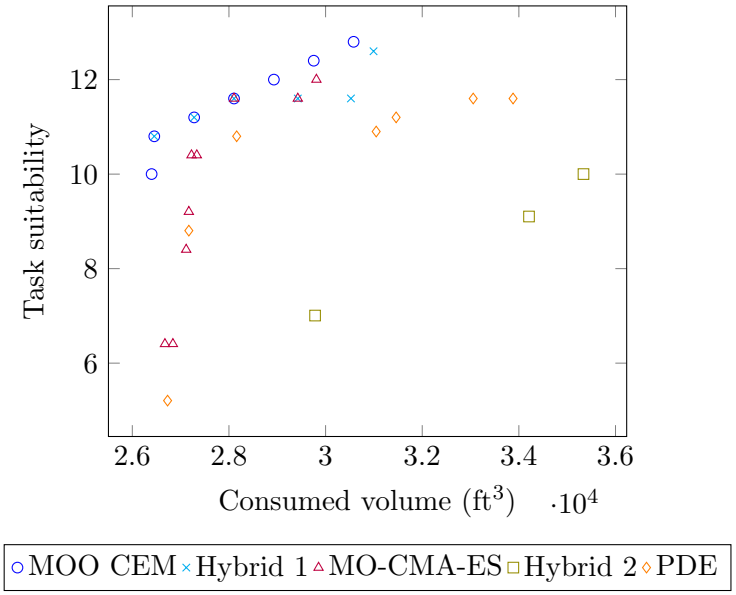


Figure B.4: A sample of Pareto fronts achieved by the algorithms on MRR Case B

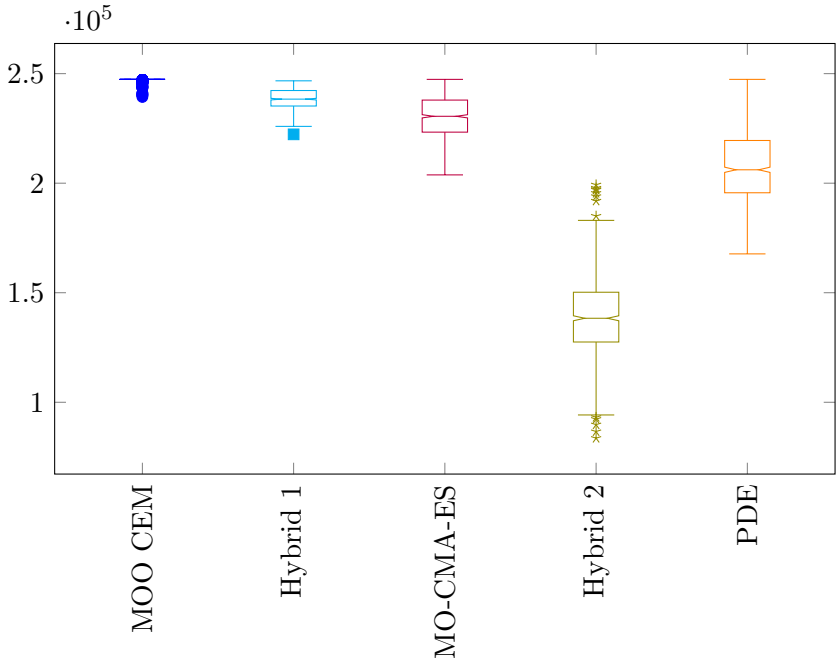


Figure B.5: Box plot of hypervolumes achieved when solving MRR Case B.

Table B.2: Mann-Whitney U-test results for MRR Case B.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	1	1	1	4	**1**
Hybrid 1	0	-	1	1	1	3	**2**
MO-CMA-ES	0	0	-	1	1	2	**3**
Hybrid 2	0	0	0	-	0	0	**5**
PDE	0	0	0	1	-	1	**4**

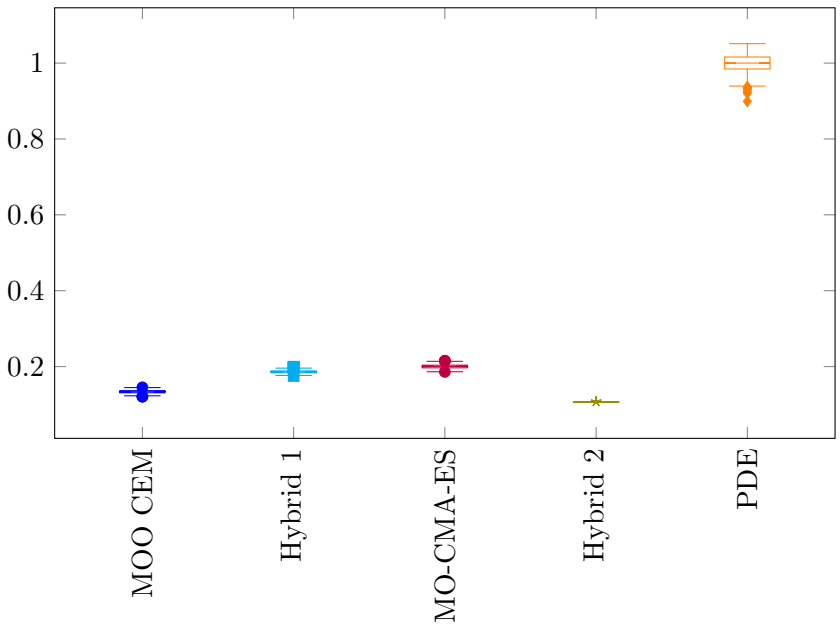


Figure B.6: Box plot of relative run times when solving MRR Case B.

B.3 MRR Case C

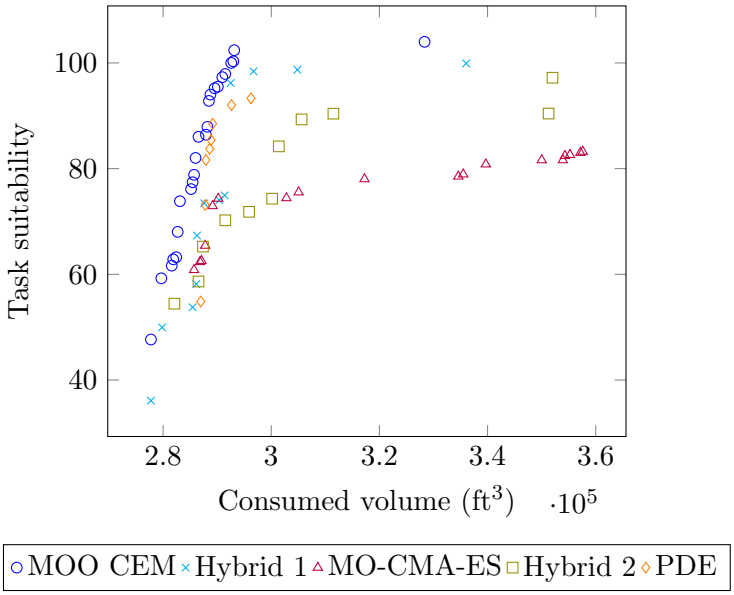


Figure B.7: A sample of Pareto fronts achieved by the algorithms on MRR Case C

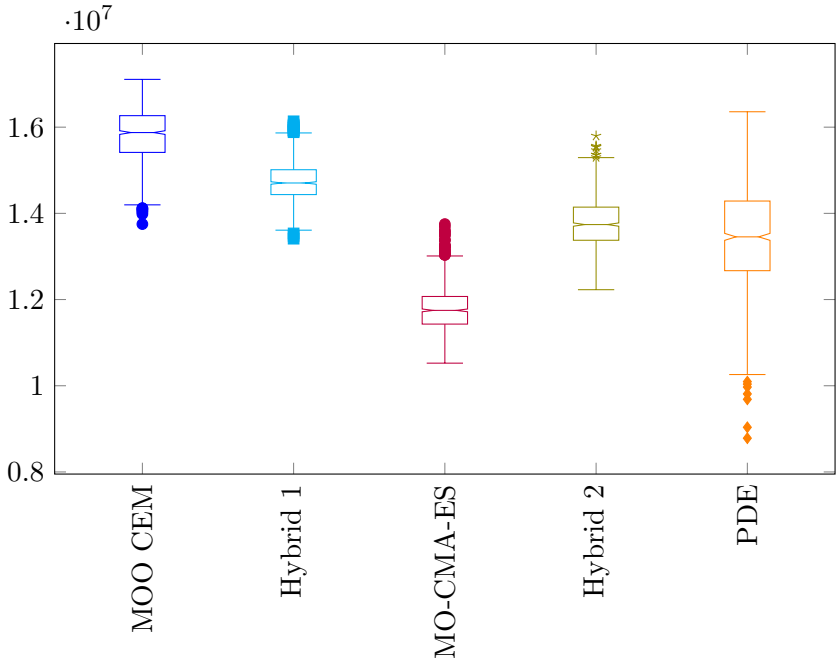


Figure B.8: Box plot of hypervolumes achieved when solving MRR Case C.

Table B.3: Mann-Whitney U-test results for MRR Case C.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	1	1	1	4	**1**
Hybrid 1	0	-	1	1	1	3	**2**
MO-CMA-ES	0	0	-	0	0	0	**5**
Hybrid 2	0	0	1	-	1	2	**3**
PDE	0	0	1	0	-	1	**4**

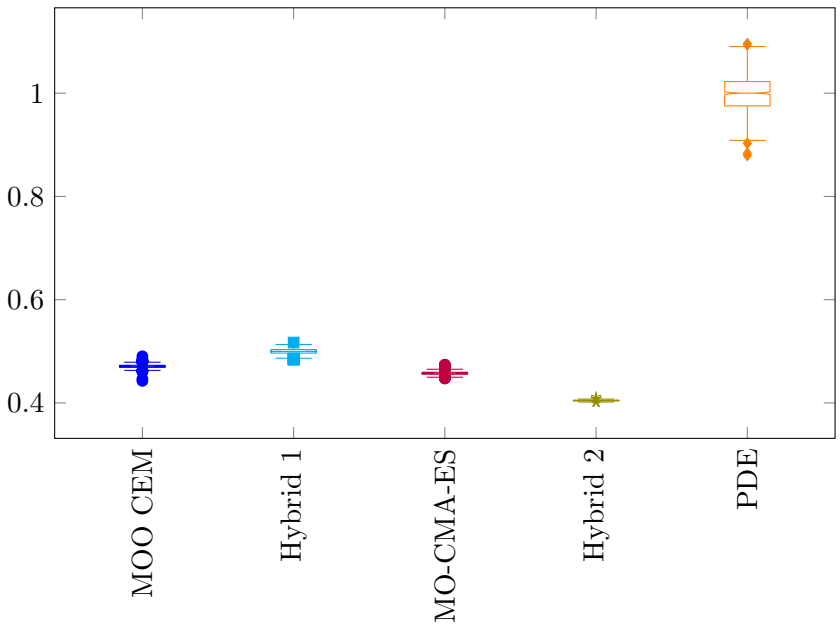


Figure B.9: Box plot of relative run times when solving MRR Case C.

B.4 MRR Case D

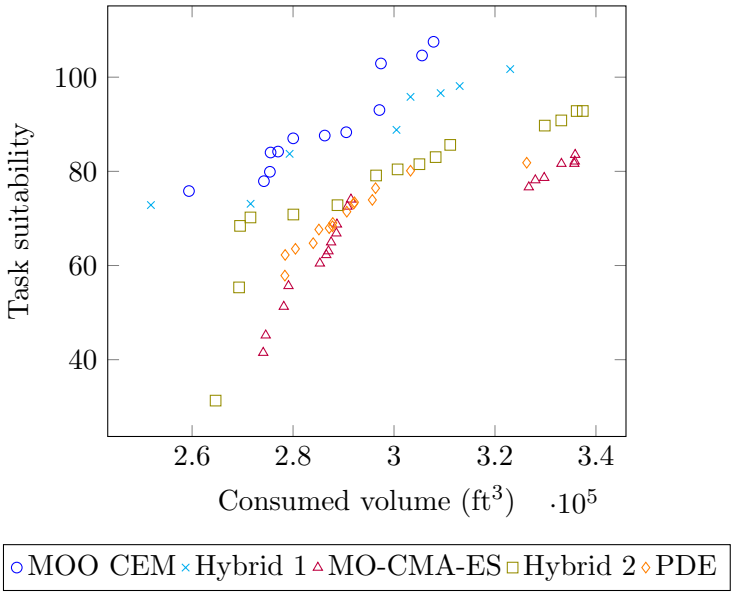


Figure B.10: A sample of Pareto fronts achieved by the algorithms on MRR Case D

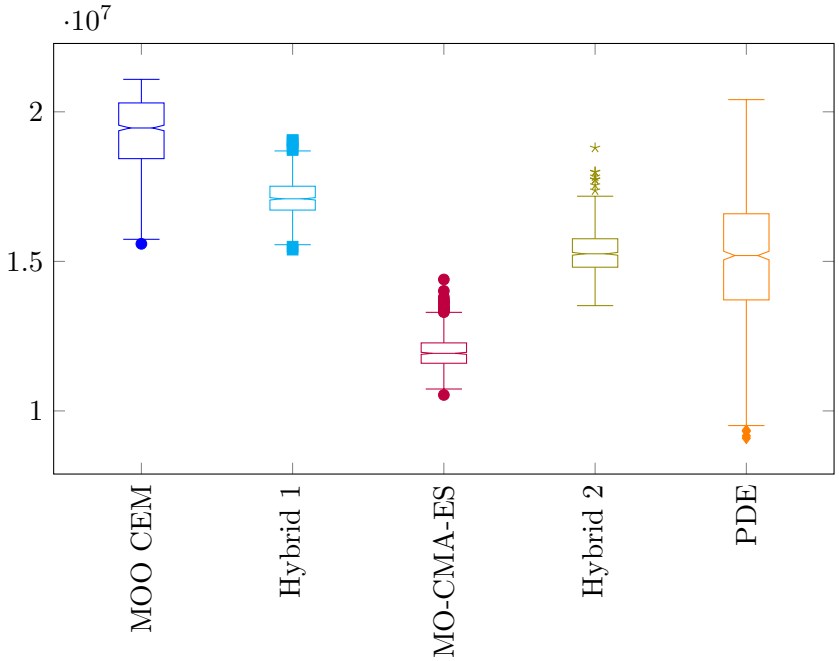


Figure B.11: Box plot of hypervolumes achieved when solving MRR Case D.

Table B.4: Mann-Whitney U-test results for MRR Case D.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	1	1	1	4	**1**
Hybrid 1	0	-	1	1	1	3	**2**
MO-CMA-ES	0	0	-	0	0	0	**5**
Hybrid 2	0	0	1	-	0	1	**4**
PDE	0	0	1	0	-	1	**4**

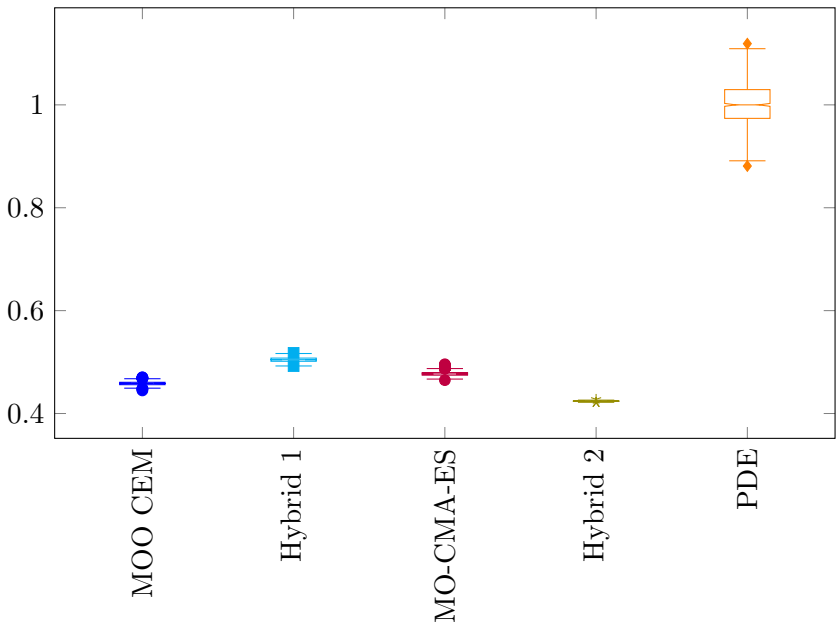


Figure B.12: Box plot of relative run times when solving MRR Case D.

B.5 MRR Case E

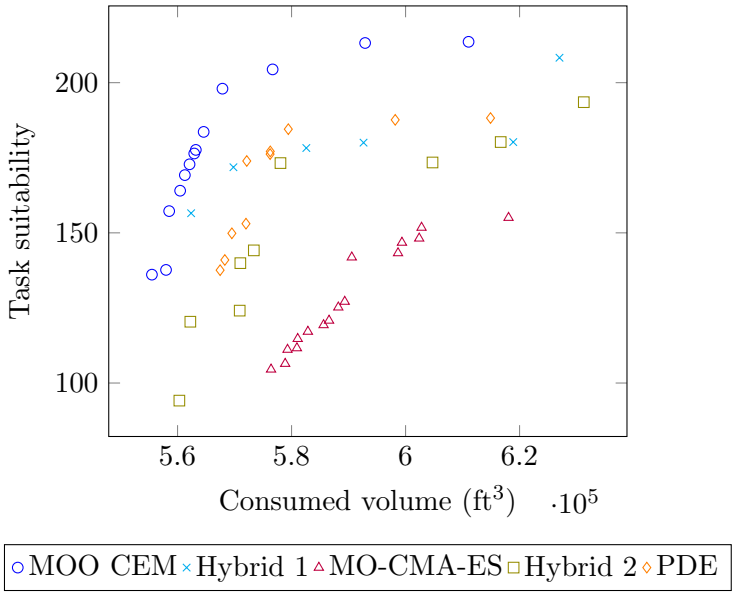


Figure B.13: A sample of Pareto fronts achieved by the algorithms on MRR Case E

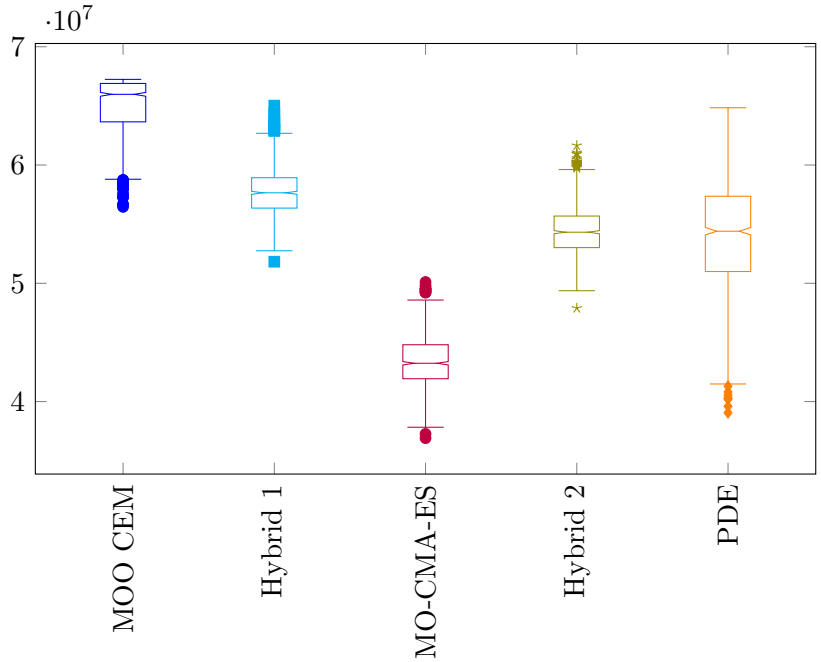


Figure B.14: Box plot of hypervolumes achieved when solving MRR Case E.

Table B.5: Mann-Whitney U-test results for MRR Case E.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	1	1	1	4	**1**
Hybrid 1	0	-	1	1	1	3	**2**
MO-CMA-ES	0	0	-	0	0	0	**5**
Hybrid 2	0	0	1	-	0	1	**4**
PDE	0	0	1	0	-	1	**4**

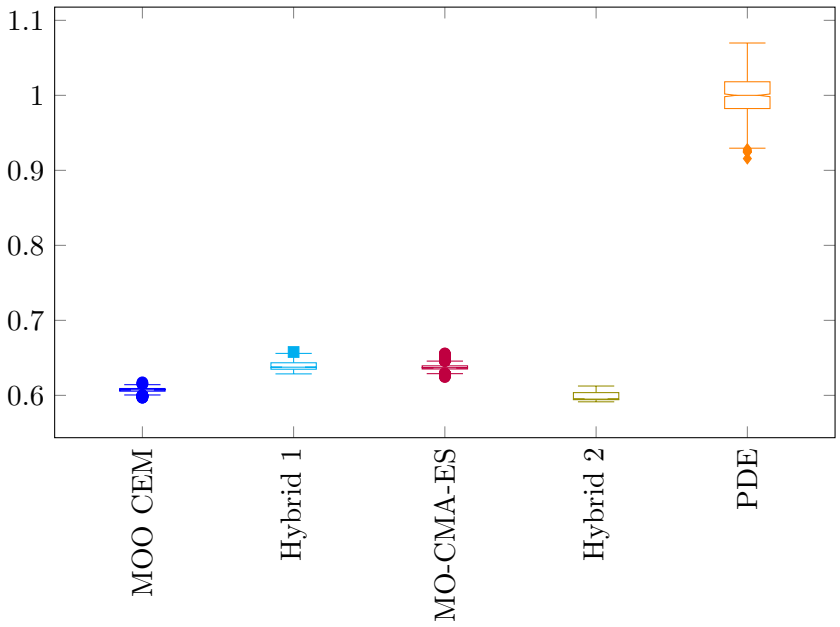


Figure B.15: Box plot of relative run times when solving MRR Case E.

B.6 MRR Case F

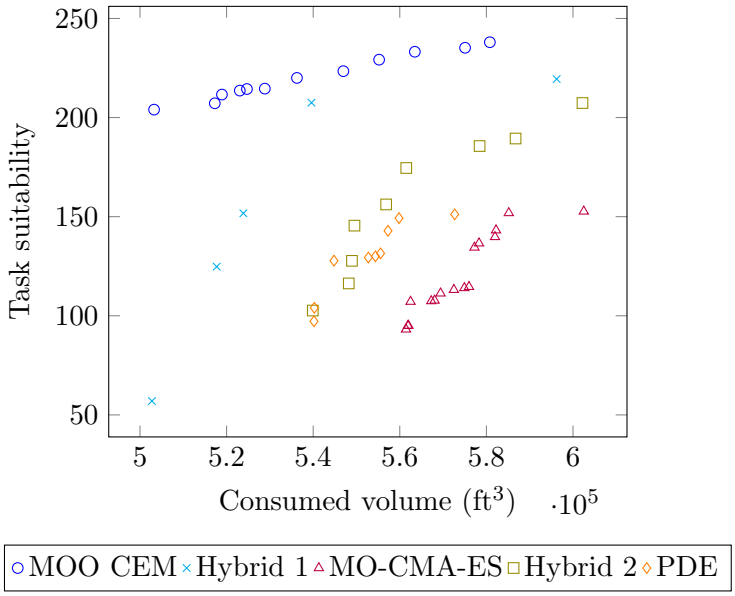


Figure B.16: A sample of Pareto fronts achieved by the algorithms on MRR Case F

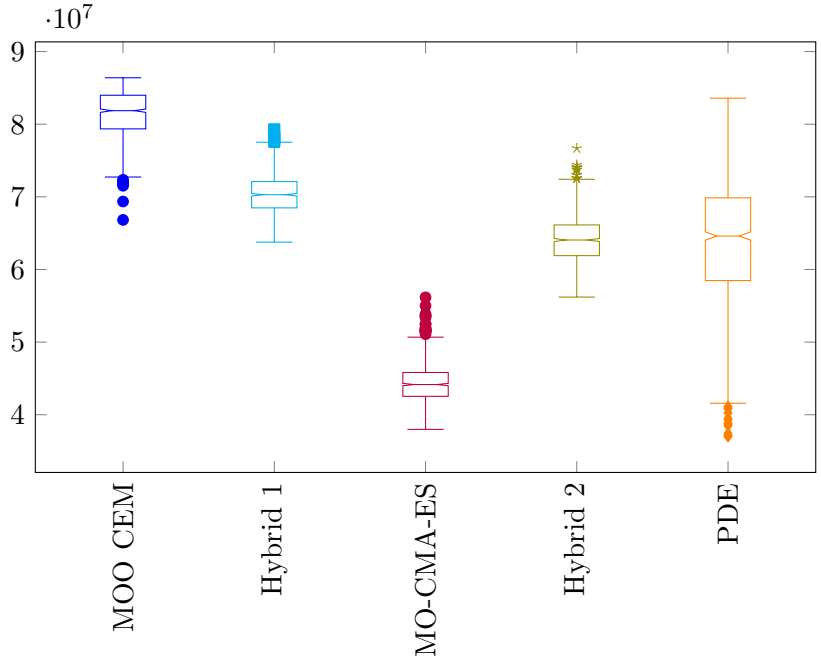


Figure B.17: Box plot of hypervolumes achieved when solving MRR Case F.

Table B.6: Mann-Whitney U-test results for MRR Case F.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	1	1	1	4	**1**
Hybrid 1	0	-	1	1	1	3	**2**
MO-CMA-ES	0	0	-	0	0	0	**5**
Hybrid 2	0	0	1	-	0	1	**4**
PDE	0	0	1	0	-	1	**4**

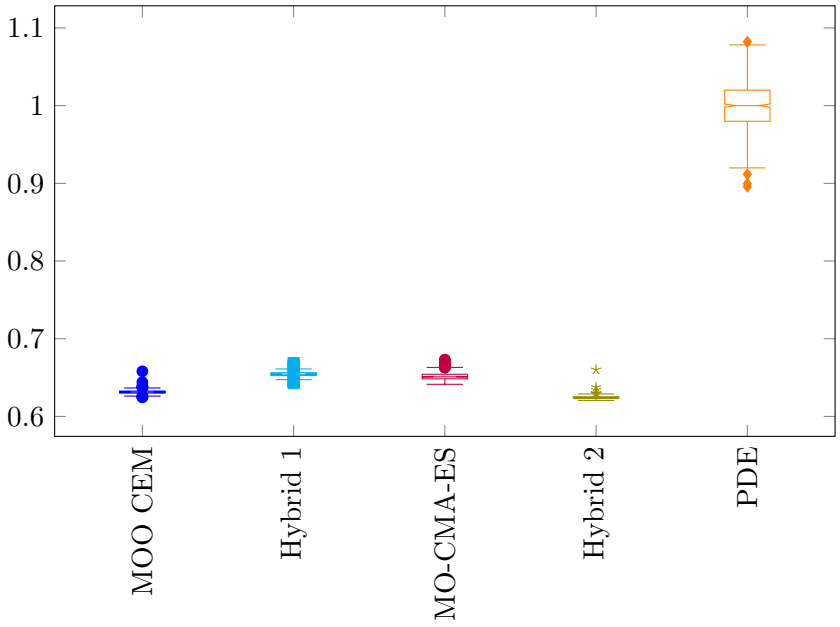


Figure B.18: Box plot of relative run times when solving MRR Case F.

Appendix C

Integrating Matlab and Simio

This appendix provides guidelines for integrating Matlab and Simio.

The dynamic, stochastic buffer allocation problem (BAP), discussed in Chapter 6, was implemented in Simio. In order to use the simulation model for function evaluation, Simio and Matlab had to be integrated.

Simio provides an application programming interface (API) which allows users to call Simio models from other programs. Unfortunately, at the time of this study, using the Simio API in combination with Matlab was impossible.

The Simio support team suggested using C# and comma separated value (CSV) files to integrate Matlab and Simio. Figure C.1 shows how Simio was integrated with Matlab using C# and CSV files. A snippet of applicable Matlab source code is shown in Listing C.1, while Listing C.2 shows the C# source code.

The C# code for writing to and reading from CSV files is attributed to Wood (2012).

Listing C.1: Matlab code snippet for integrating Matlab and Simio using C# and CSV files.

```
function f=callCSharp(DecisionVariables)
    %File path for CSV file to which decision variables are written
    ControlCSV='C:\Users\15431967\Dropbox\MrBekker_ResearchGroup\
        EsmarieScholtz\SIMIO\ControlCSV.csv';
    %Write to CSV file
    csvwrite(ControlCSV,DecisionVariables);
    %Call C# executable
```

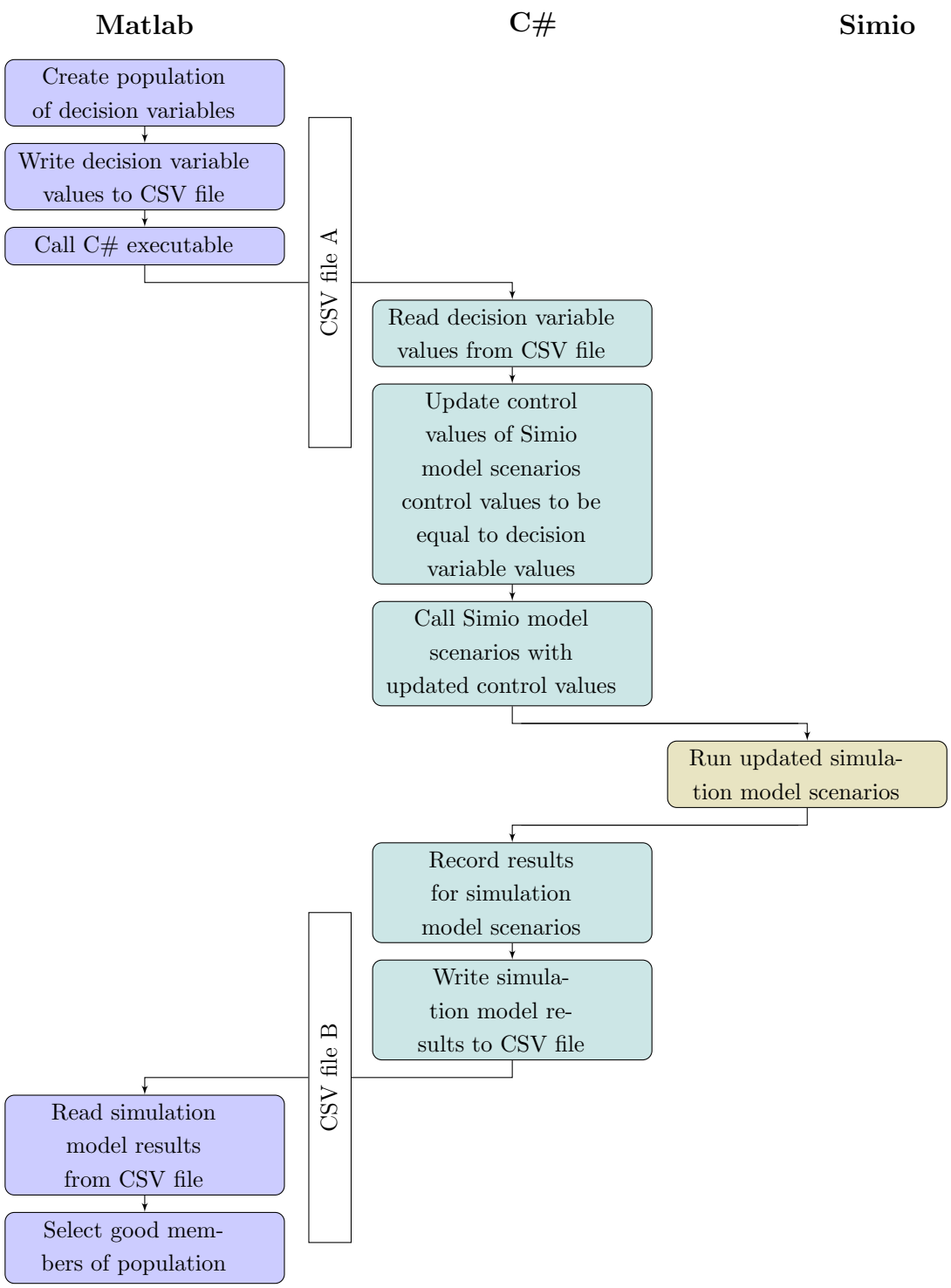


Figure C.1: Integrating Simio with Matlab using C# and CSV files.

```
    system('C:\Users\15431967\Dropbox\MrBekker_ResearchGroup\
        EsmarieScholtz\SIMIO\RunExperimentsConsole\bin\Release\
        RunExperimentsConsole.exe');
    %File path for CSV file from which results are read
    ResponseCSV='C:\Users\15431967\Dropbox\MrBekker_ResearchGroup\
        EsmarieScholtz\SIMIO\ResponseCSV.csv';
    %Read CSV file
    f=csvread(ResponseCSV);
end
```

Listing C.2: C# code for integrating Matlab and Simio using CSV files.

```
using System;
using System.IO;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using SimioAPI;
using System.Data;
using System.Data.Common;
using System.Globalization;

namespace RunExperimentsConsole
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] warnings;
            //Load Simio project
            ISimioProject project = SimioProjectFactory.LoadProject(
                "C:\\Users\\15431967\\Dropbox\\MrBekker_ResearchGroup
                \\EsmarieScholtz\\SIMIO\\JB_Test_BAP.spfx", out
                warnings); //MOP 63, 64, 65
            //Load the specific model
            IModel myModel = project.Models["Model"];
            //Load the specific experiment
            IExperiment myExperiment = myModel.Experiments["
                Experiment1"];
            //Some declarations
            string scenarioName;
```

```
IScenario myScenario;
string controlName;
IExperimentControl thisControl;
string valueRead;
int repsReq = 5;
int b = 1;
myExperiment.Reset();
// Read control variable data from CSV file
string ReadPath = "C:\\Users\\15431967\\Dropbox\\
    MrBekker_ResearchGroup\\EsmarieScholtz\\SIMIO\\
    ControlCSV.csv";
using (ReadWriteCsv.CsvFileReader reader = new
    ReadWriteCsv.CsvFileReader(ReadPath))
{
     ReadWriteCsv.CsvRow row = new ReadWriteCsv.CsvRow();
     while (reader.ReadRow(row))
     {
          for (int a = 0; a < myExperiment.Controls.Count;
              a++)
          {
               scenarioName = "Scenario" + b;
               myScenario = myExperiment.Scenarios[
                   scenarioName];
               if (myScenario==null)
               {
                    myExperiment.Scenarios.Create(
                        scenarioName);
                    myScenario = myExperiment.Scenarios[
                        scenarioName];
               }
               myScenario.ReplicationsRequired = repsReq;
               controlName = myExperiment.Controls[a].Name.
                   ToString();
               valueRead = row[a]
               //Set the control value to be equal to the
                   decision variable value
               thisControl = myExperiment.Controls[
                   controlName];
               myScenario.SetControlValue(myExperiment.
                   Controls[controlName], valueRead.ToString
                   ());
```

```
            }
            b = b + 1;
        }
    }
    //Delete superfluous scenarios
    int OrigCount = myExperiment.Scenarios.Count;
    if ( OrigCount> b - 1)
    {
        for (int a = b; a <= OrigCount; a++)
        {
            scenarioName = "Scenario" + a;
            myScenario = myExperiment.Scenarios[scenarioName
                ];
            myExperiment.Scenarios.Remove(myScenario);
        }
    }
    //Run simulation model scenarios
    myExperiment.Run();
    // Write output data to CSV file
    string WritePath = "C:\\Users\\15431967\\Dropbox\\
        MrBekker_ResearchGroup\\EsmarieScholtz\\SIMIO\\
        ResponseCSV.csv";
    using (ReadWriteCsv.CsvFileWriter writer = new
        ReadWriteCsv.CsvFileWriter(WritePath))
    foreach (IScenario scenario in myExperiment.Scenarios)
    {
        double responseValue = 0.0;
        ReadWriteCsv.CsvRow row = new ReadWriteCsv.CsvRow();
        for (int j = 0; j < myExperiment.Responses.Count; j
            ++)
        {
            myExperiment.Scenarios[scenario.Name].
                GetResponseValue(myExperiment.Responses[j],
                ref responseValue);
            row.Add(String.Format(responseValue.ToString()))
                ;
        }
            writer.WriteRow(row);

    }
```

```
        }
    }
}

namespace ReadWriteCsv
{
    /// <summary>
    /// Class to store one CSV row
    /// </summary>
    public class CsvRow : List<string>
    {
        public string LineText { get; set; }
    }

    /// <summary>
    /// Class to write data to a CSV file
    /// </summary>
    public class CsvFileWriter : StreamWriter
    {
        public CsvFileWriter(Stream stream)
            : base(stream)
        {
        }

        public CsvFileWriter(string filename)
            : base(filename)
        {
        }

        /// <summary>
        /// Writes a single row to a CSV file.
        /// </summary>
        /// <param name="row">The row to be written</param>
        public void WriteRow(CsvRow row)
        {
            StringBuilder builder = new StringBuilder();
            bool firstColumn = true;
            foreach (string value in row)
            {
                // Add separator if this isn't the first value
                if (!firstColumn)
```

```
                builder.Append(',');
            // Implement special handling for values that
                contain comma or quote
            // Enclose in quotes and double up any double quotes
            if (value.IndexOfAny(new char[] { '"', ',' }) != -1)
                builder.AppendFormat("\"{0}\"", value.Replace("
                    \"", "\"\""));
            else
                builder.Append(value);
            firstColumn = false;
        }
        row.LineText = builder.ToString();
        WriteLine(row.LineText);
    }
}

/// <summary>
/// Class to read data from a CSV file
/// </summary>
public class CsvFileReader : StreamReader
{
    public CsvFileReader(Stream stream)
        : base(stream)
    {
    }

    public CsvFileReader(string filename)
        : base(filename)
    {
    }

    /// <summary>
    /// Reads a row of data from a CSV file
    /// </summary>
    /// <param name="row"></param>
    /// <returns></returns>
    public bool ReadRow(CsvRow row)
    {
        row.LineText = ReadLine();
        if (String.IsNullOrEmpty(row.LineText))
            return false;
```

```
int pos = 0;
int rows = 0;

while (pos < row.LineText.Length)
{
    string value;

    // Special handling for quoted field
    if (row.LineText[pos] == '"')
    {
        // Skip initial quote
        pos++;

        // Parse quoted value
        int start = pos;
        while (pos < row.LineText.Length)
        {
            // Test for quote character
            if (row.LineText[pos] == '"')
            {
                // Found one
                pos++;

                // If two quotes together, keep one
                // Otherwise, indicates end of value
                if (pos >= row.LineText.Length || row.
                    LineText[pos] != '
                    "')
                {
                    pos--;
                    break;
                }
            }
            pos++;
        }
        value = row.LineText.Substring(start, pos -
            start);
        value = value.Replace("\"\"", "\"");
    }
    else
```

```
                {
                    // Parse unquoted value
                    int start = pos;
                    while (pos < row.LineText.Length && row.LineText
                        [pos] != ',')
                        pos++;
                    value = row.LineText.Substring(start, pos -
                        start);
                }

                // Add field to list
                if (rows < row.Count)
                    row[rows] = value;
                else
                    row.Add(value);
                rows++;

                // Eat up to and including next comma
                while (pos < row.LineText.Length && row.LineText[pos
                    ] != ',')
                    pos++;
                if (pos < row.LineText.Length)
                    pos++;
            }
            // Delete any unused items
            while (row.Count > rows)
                row.RemoveAt(rows);

            // Return true if any columns read
            return (row.Count > 0);
        }
    }
}
```

Appendix D

Simulation case results

This appendix presents the results for the buffer allocation problem (BAP) cases. For each case, a sample of the Pareto fronts achieved is presented. Only one set of box plots are also shown: a summary of the hypervolumes achieved by the algorithms. The relative run times are not shown since the function evaluations are very time consuming. As a result, differences in the run times of the algorithms are very small. The results of the Mann-Whitney U-tests as performed on the achieved hypervolumes are presented for all the cases. For the Mann-Whitney U-test results, if a matrix entry $ij = 1$, it indicates that Algorithm i achieved a significantly higher hypervolume than Algorithm j at a 5% significance level. The *Outperformed* column sums the total number of algorithms that Algorithm i outperformed, whereas the *Rank* column indicates which algorithm performed best on the problem at hand (with 1 being the best and 5 being the worst).

D.1 BAP Case A

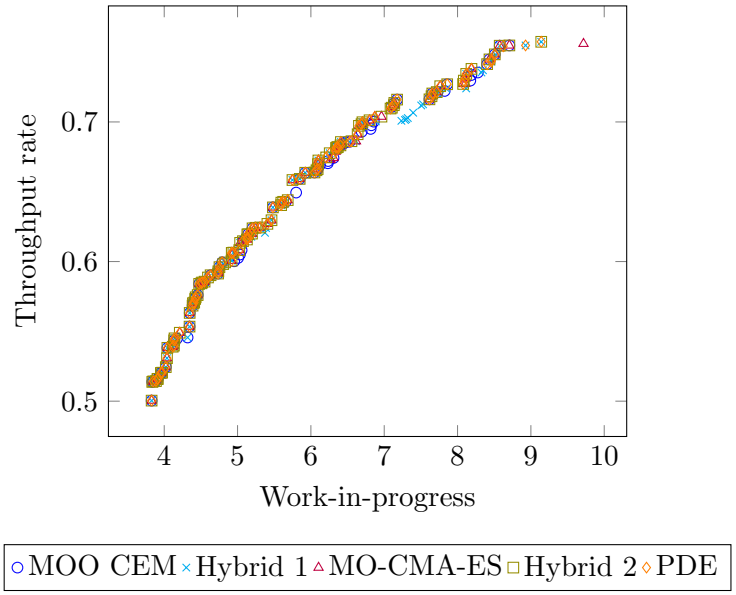


Figure D.1: A sample of Pareto fronts achieved by the algorithms on BAP Case A.

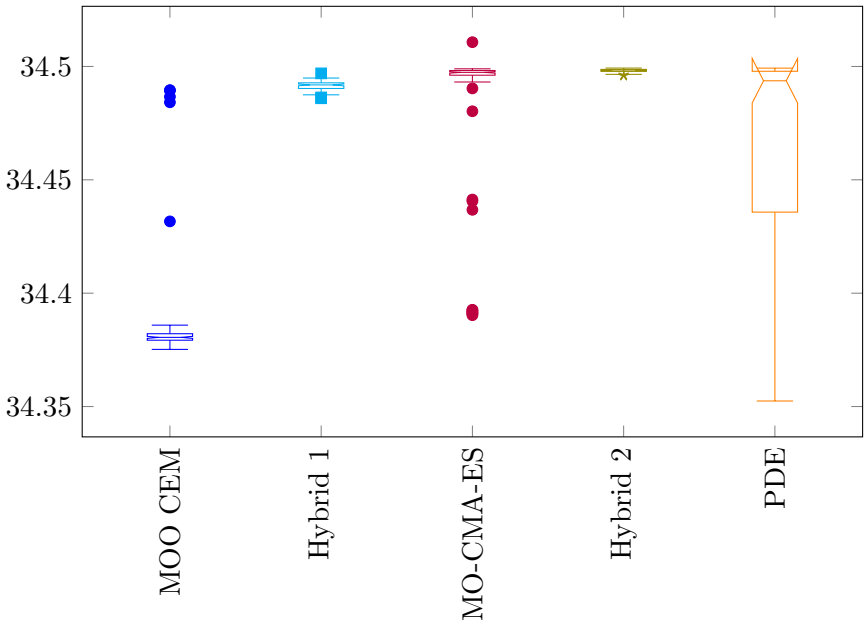


Figure D.2: Box plot of hypervolumes achieved when solving BAP Case A.

Table D.1: Mann-Whitney U-test results for BAP Case A.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	0	0	1	**4**
MO-CMA-ES	1	1	-	0	1	3	**2**
Hybrid 2	1	1	1	-	1	4	**1**
PDE	1	0	0	0	-	1	**4**

D.2 BAP Case B

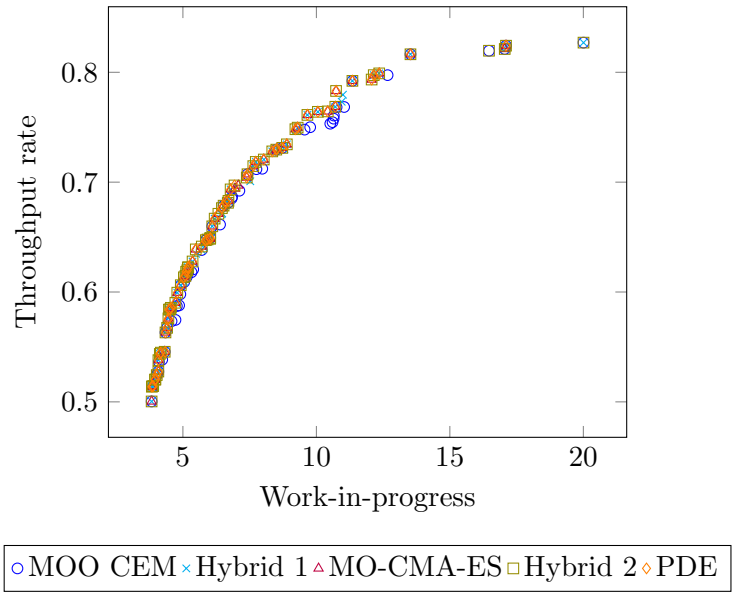


Figure D.3: A sample of Pareto fronts achieved by the algorithms on BAP Case B.

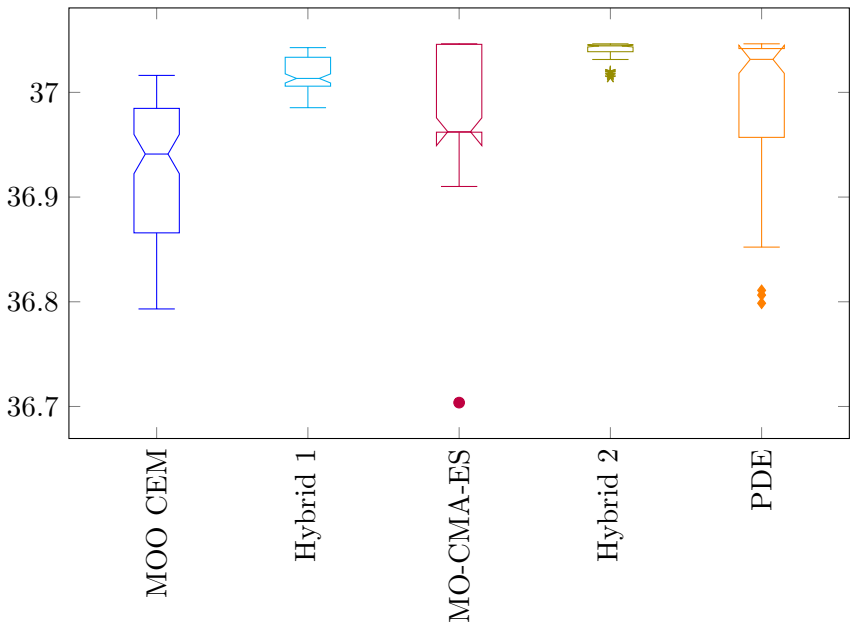


Figure D.4: Box plot of hypervolumes achieved when solving BAP Case B.

Table D.2: Mann-Whitney U-test results for BAP Case B.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	0	0	0	0	0	**5**
Hybrid 1	1	-	0	0	0	1	**4**
MO-CMA-ES	1	0	-	0	1	2	**3**
Hybrid 2	1	1	1	-	1	4	**1**
PDE	1	0	0	0	-	1	**4**

D.3 BAP Case C

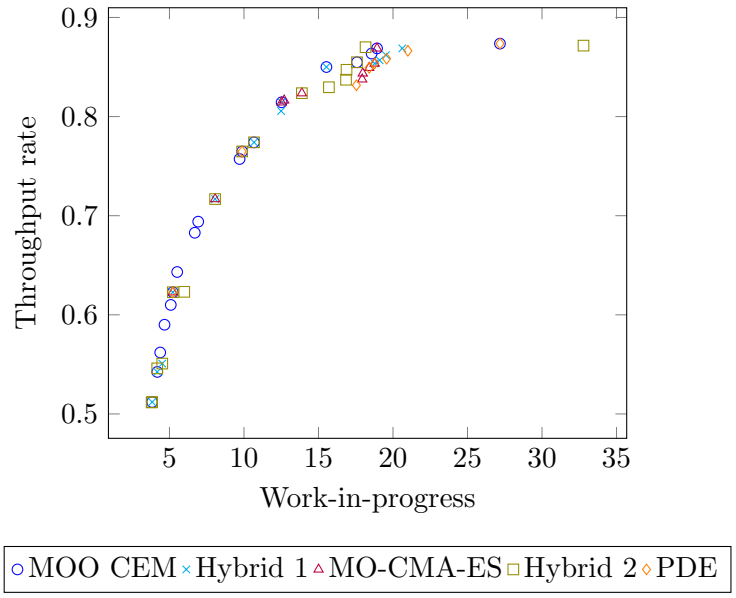


Figure D.5: A sample of Pareto fronts achieved by the algorithms on BAP Case C.

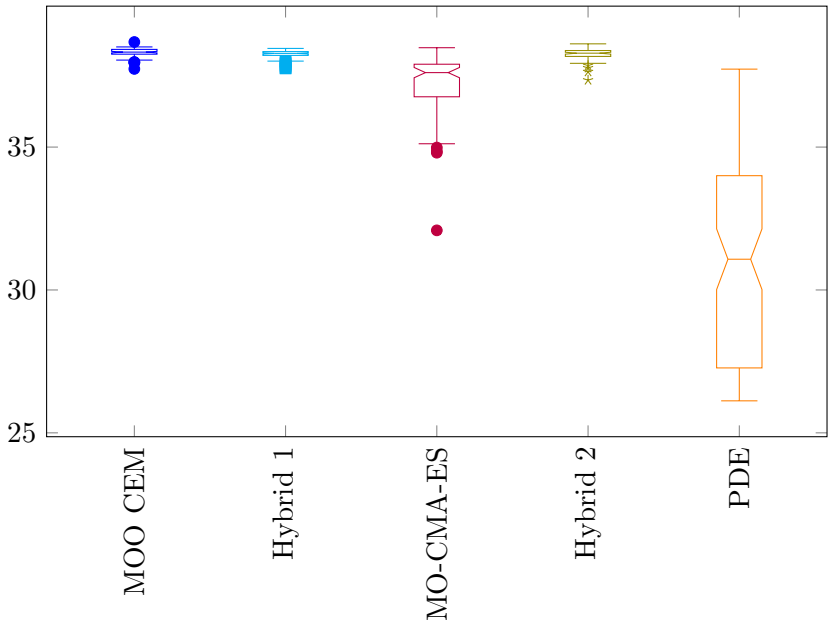


Figure D.6: Box plot of hypervolumes achieved when solving BAP Case C.

Table D.3: Mann-Whitney U-test results for BAP Case C.

	MOO CEM	*Hybrid 1*	*MO-CMA-ES*	*Hybrid 2*	*PDE*	*Outperformed*	*Rank*
MOO CEM	-	1	1	1	1	4	**1**
Hybrid 1	0	-	1	0	1	2	**3**
MO-CMA-ES	0	0	-	0	1	1	**4**
Hybrid 2	0	0	1	-	1	2	**3**
PDE	0	0	0	0	-	0	**5**