Applying the cross-entropy method in multi-objective optimisation of dynamic stochastic systems

James Bekker

Dissertation presented for the degree of Doctor of Philosophy in the Faculty of Engineering at Stellenbosch University

Promoter: Professor JH van Vuuren

December 2012

Declaration

By submitting this dissertation 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 any third-party rights, and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Abstract

A difficult subclass of engineering optimisation problems is the class of optimisation problems which are dynamic and stochastic. These problems are often of a non-closed form and thus studied by means of computer simulation. Simulation production runs of these problems can be time-consuming due to the computational burden implied by statistical inference principles. In multi-objective optimisation of engineering problems, large decision spaces and large objective spaces prevail, since two or more objectives are simultaneously optimised and many problems are also of a combinatorial nature. The computational burden associated with solving such problems is even larger than for most single-objective optimisation problems, and hence an efficient algorithm that searches the vast decision space is required. Many such algorithms are currently available, with researchers constantly improving these or developing more efficient algorithms. In this context, the term "efficient" means to provide near-optimised results with minimal evaluations of objective function values. Thus far research has often focused on solving specific benchmark problems, or on adapting algorithms to solve specific engineering problems.

In this research, a multi-objective optimisation algorithm, based on the cross-entropy method for single-objective optimisation, is developed and assessed. The aim with this algorithm is to reduce the number of objective function evaluations, particularly when time-dependent (dynamic), stochastic processes, as found in Industrial Engineering, are studied. A brief overview of scholarly work in the field of multi-objective optimisation is presented, followed by a theoretical discussion of the cross-entropy method. The new algorithm is developed, based on this information, and assessed considering continuous, deterministic problems, as well as discrete, stochastic problems. The latter include a classical single-commodity inventory problem, the well-known buffer

allocation problem, and a newly designed, laboratory-sized reconfigurable manufacturing system. Near multi-objective optimisation of two practical problems were also performed using the proposed algorithm. In the first case, some design parameters of a polymer extrusion unit are estimated using the algorithm. The management of carbon monoxide gas utilisation at an ilmenite smelter is complex with many decision variables, and the application of the algorithm in that environment is presented as a second case.

Quality indicator values are estimated for thirty-four test problem instances of multi-objective optimisation problems in order to quantify the quality performance of the algorithm, and it is also compared to a commercial algorithm.

The algorithm is intended to interface with dynamic, stochastic simulation models of real-world problems. It is typically implemented in a programming language while the simulation model is developed in a dedicated, commercial software package.

The proposed algorithm is simple to implement and proved to be efficient on test problems.

Opsomming

'n Moeilike deelklas van optimeringsprobleme in die ingenieurswese is optimeringsprobleme van 'n dinamiese en stogastiese aard. Sulke probleme is dikwels nie-geslote en word gevolglik met behulp van rekenaarsimulasie bestudeer. Die beginsels van statistiese steekproefneming veroorsaak dat produksielopies van hierdie probleme tydrowend is weens die rekenlas wat genoodsaak word. Groot besluitnemingruimtes en doelwitruimtes bestaan in meerdoelige optimering van ingenieursprobleme, waar twee of meer doelwitte gelyktydig geoptimeer word, terwyl baie probleme ook 'n kombinatoriese aard het. Die rekenlas wat met die oplos van sulke probleme gepaard gaan, is selfs groter as vir die meeste enkeldoelwit optimeringsprobleme, en 'n doeltreffende algoritme wat die meesal uitgebreide besluitnemingsruimte verken, is gevolglik nodig. Daar bestaan tans verskeie sulke algoritmes, terwyl navorsers steeds poog om hierdie algoritmes te verbeter of meer doeltreffende algoritmes te ontwikkel. In hierdie konteks beteken "doeltreffend" dat naby-optimale oplossings verskaf word deur die minimum evaluering van doelwitfunksiewaardes. Navorsing fokus dikwels op oplossing van standaard toetsprobleme, of aanpassing van algoritmes om 'n spesifieke ingenieursprobleem op te los.

In hierdie navorsing word 'n meerdoelige optimeringsalgoritme gebaseer op die kruis-entropie-metode vir enkeldoelwit optimering ontwikkel en geassesseer. Die mikpunt met hierdie algoritme is om die aantal evaluerings van doelwitfunksiewaardes te verminder, spesifiek wanneer tydafhanklike (dinamiese), stogastiese prosesse soos wat dikwels in die Bedryfsingenieurswese teëgekom word, bestudeer word. 'n Bondige oorsig van navorsing in die veld van meerdoelige optimering word gegee, gevolg deur 'n teoretiese bespreking van die kruis-entropiemetode. Die nuwe algoritme se ontwikkeling is hierop gebaseer, en dit word geassesseer deur kontinue, deterministiese probleme sowel as diskrete,

stogastiese probleme benaderd daarmee op te los. Laasgenoemde sluit in 'n klassieke enkelitem voorraadprobleem, die bekende buffer-toedelingsprobleem, en 'n nuut-ontwerpte, laboratorium-skaal herkonfigureerbare vervaardigingstelsel. Meerdoelige optimering van twee praktiese probleme is met die algoritme uitgevoer. In die eerste geval word sekere ontwerpparameters van 'n polimeer-uittrekeenheid met behulp van die algoritme beraam. Die bestuur van koolstofmonoksiedbenutting in 'n ilmeniet-smelter is kompleks met verskeie besluitnemingveranderlikes, en die toepassing van die algoritme in daardie omgewing word as 'n tweede geval aangebied.

Verskeie gehalte-aanwyserwaardes word beraam vir vier-en-dertig toetsgevalle van meerdoelige optimeringsprobleme om die gehalte-prestasie van die algoritme te kwantifiseer, en dit word ook vergelyk met 'n kommersiële algoritme.

Die algoritme is veronderstel om te skakel met dinamiese, stogastiese simulasiemodelle van regtewêreldprobleme. Die algoritme sal tipies in 'n programmeertaal geïmplementeer word terwyl die simulasiemodel in doelmatige, kommersiële programmatuur ontwikkel sal word. Die voorgestelde algoritme is maklik om te implementeer en dit het doeltreffend gewerk op toetsprobleme.

Acknowledgements

I am grateful to the following people who supported me toward this dissertation:

- Professor Jan H van Vuuren, my promoter, role model and esteemed colleague, for his guidance and wisdom on a strategic level, and his meticulous attention to detail.
- The Chairman of the Department of Industrial Engineering at Stellenbosch University, Doctor André van der Merwe, for affording me the major scarce commodities, namely time and financial support.
- My colleagues, for absorbing a substantial part of my workload and supporting me in many ways.
- My mentors, Nicolaas du Preez and the late Willem van Wijck, for introducing me to computer simulation and supporting me when I needed it the most.
- My parents, for encouraging me to study.
- My family, for their support, encouragement and above all, patience.
- Marlene Rose, for proofreading the document and making valuable suggestions.
- The many students I worked with and will work with, for enriching my life and teaching me more than I can ever teach them.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

Roman Symbols

B_i	Size of buffer space i, page 85
C_v	Vehicle capacity, page 70
CV	Pareto front convergence indicator, page 21
D	Number of decision variables in an optimisation problem, page 4
D_i	Number of units demanded by customer i, page 81
d_i	Euclidian distance, page 20
D_i'	Internal diameters of extrusion equipment, page 120
E_r	Number of rows in elite vector, page 57
e_t	Flight thickness of extrusion equipment, page 120
f, f_i	Mathematical function, including probability mass and density function, page 3
GD	Generation Distance indicator, page 20
g_i	Mathematical function, including probability mass and density function, page 4
h_i	Mathematical function, including probability mass and density function, page 4
h_w	Confidence interval half-width, page 134
I	Indicator function, page 41
$\overline{I}$	Average inventory carried over period T, page 81
$I_{\epsilon+}$	Epsilon quality indicator, page 131
I_H	Hypervolume quality indicator, page 131
I_q	Unary quality indicator, page 130
I_t	Inventory level at time t, page 81
K	Number of objectives in an optimisation problem, page 4
l	Rare-event probability in importance sampling, page 41
L_i	Decision variable upper limit, page 56

l_i	Decision variable lower limit, page 56
ls_i	Section lengths of extrusion equipment, page 120
M	Number of inequality constraints, page 4
m	Number of machines in the buffer allocation problem, page 85
ME	Maximum Pareto front error indicator, page 21
N	User-specified population size for population-based algorithms, page 16
n	Number of members in a set, for example, the number of nodes in a VRP or number of buffers in a BAP, page 70
N_a	Size of solution archive, page 88
N_C	Number of discrete events in period T, page 81
n_d	Number of elements in the discrete decision vector, page 45
N_s	Extrusion equipment screw rotation rate, page 120
P	Probability distribution of a discrete optimisation with the cross-entropy method, page 45
p_h	Probability of inverting MOO CEM histogram counts, page 58
p_j	Probabilities of elements in a discrete decision vector, page 45
Q	Number of equality constraints, page 4
r	Number of classes of the elite vector of the MOO CEM algorithm, page 57
r_i	Mean exponential repair rate, page 89
s	Reorder level, page 80
S	Reorder quantity, page 80
S_c	Inventory shortage per customer, page 81
S_L	Service level, page 81
s_p	Screw pitch of extrusion equipment, page 120
SP	Pareto front spacing indicator, page 21
T	General indicator of end of a time period, page 81
T_{bi}	Barrel temperature of extrusion equipment, page 120
T_j	Time duration of work-in-progress level j, page 104
T_R	Measure of throughput rate, page 87
t_r	Residence time inside extrusion equipment, page 120
t_d^ν	Total delay time on a route in the VRPSTW, page 71
t_w^ν	Total waiting time to start on a route in the VRPSTW, page 71
U	Random number, uniformly distributed on $(0, 1)$, page 58
W_M	Maximum work-in-progress observed over a time period, page 104
W_j	Instantaneous work-in-progress level, page 104
W_P	Measure of work-in-progress, page 87

Greek Symbols

α	Smoothing parameter for the cross-entropy method, page 43
β	Mean interarrival time for a Poisson process, page 80
β_i	Mean exponential failure rate, page 89
δ	Termination counter of the cross-entropy method, page 43
ϵ_c	MOO CEM common termination threshold, page 61
ϵ	Box size of ϵ-dominance to regulate convergence, page 16
γ	Cross-entropy optimisation rare-event threshold value, page 40
κ	MOO CEM histogram class index, page 57
$\overline{\lambda}_n$	Throughput rate estimation of a buffer allocation problem, n niches, page 89
λ^*	Exact throughput rate of a buffer allocation problem, page 89
μ, μ_i	Mean of a distribution, page 46
ω	Stochastic component of a decision problem, page 3
ν	Number of vehicles in the vehicle routing problem, page 70
Ω	Feasible region of an optimisation problem, page 4
Ω_q	Set of all approximation sets, page 131
ϕ_i	Truncated normal distribution, page 55
ρ	Rank value of multi-objective solution vector, page 55
ρ_E	MOO CEM algorithm ranking threshold, page 56
σ	Standard deviation of a distribution, page 46
τ_{ij}	MOO CEM histogram frequency count, decision variable i, class j, page 57
τ_m	Maximum number of evaluations, page 75
θ	Input vector to an optimisation problem, page 3
Θ	The complete input domain of an optimisation problem, page 3
ϱ	User-specified rare-event threshold value for the cross-entropy method, page 42
Υ	Objective function in the case of discrete cross-entropy optimisation, page 43
Z_i	Objectives of the vehicle routing problem, page 70

Other Symbols

$\mathbf{C}_i$	MOO CEM algorithm histogram class boundaries of decision variable x_i, page 57
$\mathcal{C}$	The set of customers in the VRP, page 70
$\mathcal{D}$	Kullback-Leibler distance, page 41
$\mathbb{E}$	Mathematical expectation, page 41
$\mathcal{G}$	A directed graph in the VRP, page 70

$\mathcal{N}$	The set of vertices in the VRP, page 70
$\mathbb{P}$	Probability symbol, page 41
$\mathcal{P}_K$	Approximate Pareto set, page 18
$\mathcal{P}_T$	True Pareto set, representing the true Pareto front, page 18
$\mathcal{P}_R$	Reference Pareto set, page 131
$\mathcal{V}$	The set of vehicles in the VRP, page 70
$\mathcal{V}_p$	Parameter vector set of the cross-entropy method, page 42
$\mathbf{W}$	Working matrix of the MOO CEM algorithm, page 56
$\mathcal{X}$	Feasible region of cross-entropy optimisation problem, page 40

Abbreviations and Acronyms

ARMOGA	Adaptive range multi-objective genetic algorithm, page 15
AS	Ant systems, page 24
BAP	Buffer allocation problem, page 84
CCPSO	Cooperative Co-evolutionary Multi-objective Particle Swarm Optimisation, page 25
CE	Cross-entropy, page 6
CEM	Cross-entropy method, page 6
CMA-ES	Covariance matrix adaptation evolution strategy, page 29
CMOIA	Constrained multi-objective immune algorithm, page 28
CONPIP	Constant number of projects in progress, page 51
DE	Differential evolution, page 27
DEMO	Differential evolution for multi-objective optimisation, page 27
DES	Discrete event simulation, page 2
EA	Evolutionary Algorithm, page 12
EMOO	Evolutionary multi-objective optimisation, page 15
EP	Evolutionary Programming, page 12
ES	Evolution Strategy, page 12
GA	Genetic Algorithm, page 12
HA	Hyperarea, page 134
HMOIA	Hybrid Multi-objective Immune Algorithm, page 28
INVN	Inventory problem, page 134
LRP	Location routing problem, page 23
MADM	Multi-attribute decision-making, page 3
MA	Memetic Algorithm, page 30
MDQL	Multi-objective Distributed Q-learning, page 26
MIMD	Multiple Instruction Multiple Data, page 51
MISA	Multi-objective Immune System Algorithm, page 28
MOAQA	Multi-objective Ant-Q algorithm, page 24

MOCBA	Multi-objective optimal computational budget allocation, page 6
MO-CMA-ES	Multi-objective covariance matrix adaptation evolution strategy, page 29
MOEA	Multi-objective evolutionary algorithm, page 14
MOGA	Multi-objective Genetic Algorithm, page 14
MOMGA	Multi-objective Messy Genetic Algorithm, page 14
MOO CEM	Multi-objective optimisation using the cross-entropy method, page 54
MOO	Multi-objective optimisation, page 4
MOPSO	Multi-objective Particle Swarm Optimisation, page 17
MORS	Multi-objective ranking and selection, page 6
MOSADE	Measure for a self-adaptive differential evolution, page 17
MOSS	Multiple-objective Scatter Search, page 14
MOTS	Multi-objective Tabu Search, page 24
NPGA	Niched-Pareto Genetic Algorithm, page 14
NP	Nondeterministic polynomial; used in computational complexity theory, page 12
NSDE	Non-dominated Sorting Differential Evolution, page 27
NSGA	Non-dominated Sorting Genetic Algorithm, page 14
ODF	Operation dependent failures, page 85
PAES	Pareto Archived Evolution Strategy, page 14
PA	Perturbation Analysis, page 32
ParEGO	Parameterised Efficient Global Optimisation, page 6
pdf	Probability density function, page 42
PESA	Pareto Envelope-based Selection Algorithm, page 14
PF	Pareto front, page 3
pmf	Probability mass function, page 43
PSO	Particle Swarm Optimisation, page 25
RMOO	Robust Multi-objective Optimisation, page 37
RMS	Reconfigurable manufacturing system, page 116
RPSGAe	Reduced Pareto set genetic algorithm with elitism, page 123
RSM	Response Surface Methodology, page 32
SA	Simulated annealing, page 22
SFLA	Shuffled Frog-leaping Algorithm, page 30
SPEA	Strength Pareto Evolutionary Algorithm, page 14
TOPSIS	Technique for preference by similarity to the ideal solution, page 3
TSP	Travelling salesperson problem, page 68
TS	Tabu search, page 23
VRPSTW	Vehicle routing problem with soft time windows, page 68

VRPTW	Vehicle routing problem with time windows, page 68
VRP	Vehicle routing problem, page 68
WIP	Work-in-progress, page 86

CHAPTER *1*

INTRODUCTION

This chapter serves as an introduction to the research presented in this dissertation. The reasons for the inception of the research hypothesis are explained, followed by the formal statement of the research hypothesis. Finally, the structure of the document is explained.

1.1 Background to the research hypothesis

We make decisions daily in our lives and often have to consider several outcomes of a decision all at once. If one for example has to buy a car, several requirements can be considered: the acquisition cost of the car, its maintenance cost, the fuel consumption, its luxury features, power, torque and acceleration. These requirements are conflicting, since one can usually not obtain a luxurious, fast car at a low cost. So the decision maker has to compromise and look for a candidate car that satisfies most of these requirements to some extent. On the other hand, the decision maker may choose to accept the cheapest candidate car and relax or even ignore the other requirements. If one formalises the decision problem of this example, one may refer to the requirements as *objectives* and the stated attributes of candidate cars as *decision variables*. Since there is more than one conflicting objective, it is a *multi-objective decision problem*, while the decision variables are *non-commensurate*. A satisfactory candidate is considered as *near-optimum*, while a set of candidates which cannot be improved upon is the *Pareto-optimal* set. This set contains a few candidates which will all satisfy the decision maker.

The focus of this research is on aspects of multi-objective engineering decision making. Naturally, the approaches in this discipline are much more formal than in

the given example. Mathematical models to support decision making are arguably the preferred approach, and the outcomes of decisions are reflected in the value(s) of an objective function f in the case of single-objective optimisation. Constraints that model practical limitations are specified as part of the optimisation model. Optimisation methods have been developed and studied for decades and many techniques exist to find extremal values of f. The nature of f, e.g. linear, non-linear, deterministic or stochastic, and the nature of the decision variables (deterministic [discrete, continuous], stochastic [discrete, continuous]) are important. Also, the constraint functions are linear or non-linear.

While exact analytical methods have many advantages, decision making problems are often hard to formulate and model using these methods, while many problems have no closed form. In such cases, the decision maker can use *computer simulation*. It is an appealing tool for problem solving and decision making, since it allows one to realistically mimic real-world operations/processes, while it is regarded by some as the "last resort" when other problem-solving tools become inadequate. This is typically the case when studying time-dependent (dynamic) stochastic processes. Computer simulation is a wide discipline, of which the sub-discipline *discrete-event simulation* (DES) (Banks, 1998; Law & Kelton, 2000) of *dynamic*, *stochastic processes* has found its rightful place in engineering. "Dynamic" implies *time*-dependency of some model variables, and "stochastic" implies *distribution*-dependency of some model variables.

Traditionally, DES is used to study point problems. After finding a solution, it is rejected by management, or implemented, or refined and then implemented, or partially implemented. Recently, DES models have been used for *optimisation*. The models are thus often reused due to changes in business and new needs for more answers. More complicated business problems and increasing computing power naturally lead to the combination of multi-objective optimisation and computer simulation.

Usually, a performance measure (objective) in terms of profit and cost is determined and used to determine the quality of a solution. In DES, the approaches to finding the best solution for one or more performance measures (objectives) are as follows:

1. Considering a single objective and a finite number of alternatives (scenarios), the decision maker can apply statistical methods to determine a distinct

scenario, if it exists. The KN-algorithm of Kim & Nelson (2001) is the state-of-the-art approach to find such a solution.

2. The decision maker defines a finite number of scenarios for the set of decision variables and the set of two or more performance measures, and the simulation model is executed for each of these input sets. The estimated objective values of the finite solution set are normalised (output is non-homogeneous) into one value using for example the Technique for Preference by Similarity to the Ideal Solution (TOPSIS) (see Jahanshahloo *et al.* (2006)), and the best scenario is found. This case is known as multi-*attribute* decision-making (MADM).

3. A mathematical programming approach is followed where one of the objectives is selected to be the objective, and the other objectives are treated as constraints. See for example Bettonvil *et al.* (2009).

4. A *Pareto front* (PF) is estimated via some guided search to consider many possible solutions, and conclusions are made from this front. See Gil *et al.* (2007).

The latter area has received little research attention in the context of DES and industrial engineering applications, according to Rosen *et al.* (2008).

Rosen *et al.* (2007) define the traditional simulation optimisation problem as

$$\text{Minimise} \quad f(\theta) \tag{1.1}$$

$$\text{subject to} \quad \theta \in \Theta, \tag{1.2}$$

where $f(\theta) = E[\psi(\theta, \omega)]$ is the expected system performance value, and is estimated by $\hat{f}(\theta)$ from samples of a simulation model using instances of discrete or continuous feasible and possibly constrained input $\theta \in \Theta \subset \mathbb{R}^D$. The stochastic effects of the model are represented by ω.

Since f cannot be mathematically defined (in closed form), computer simulation is used to imitate its behaviour, and f is thus viewed as a black box with inputs and outputs. Also, f can be extended to define multiple objectives. These objectives can have different units of measurement, exhibit different scales, and are usually

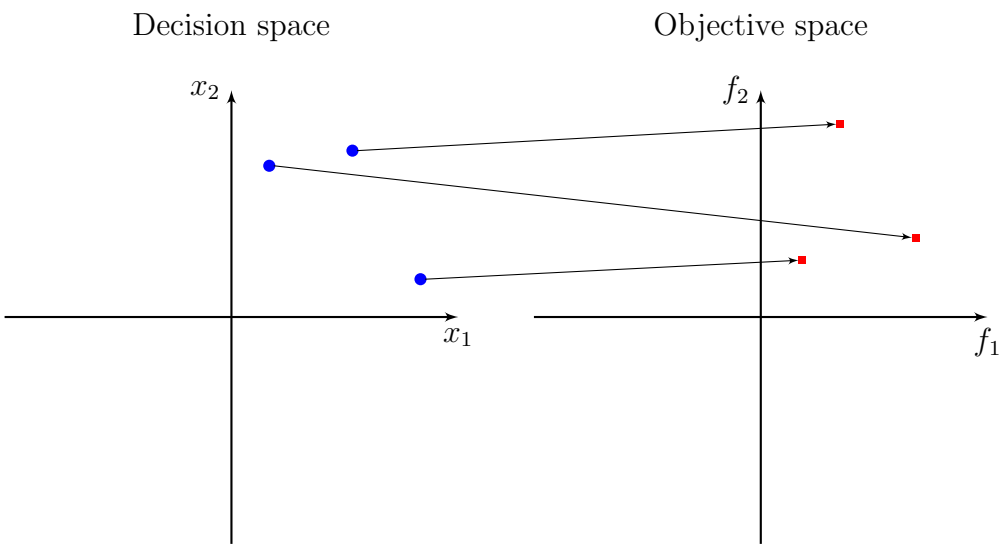

Figure 1.1: MOO mapping.

in conflict. This leads to the *Multi-objective Optimisation* (MOO) problem,

$$\begin{aligned} \text{Minimise } \mathbf{f}(\mathbf{x}) &= [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_K(\mathbf{x})]^T && (1.3)\\ \text{subject to } \mathbf{x} &\in \Omega && (1.4)\\ \Omega &= \{\mathbf{x} \mid g_i(\mathbf{x}) \leq 0, i = 1, 2, \ldots, M; && (1.5)\\ h_j(\mathbf{x}) &= 0, j = 1, \ldots, Q\} && (1.6) \end{aligned}$$

in D decision variables, K objectives and $M + Q$ constraints in (1.3) (Tsou, 2008), where $\mathbf{x} = [x_1, x_2, \ldots, x_D]^T$ is a D dimensional vector of *decision variables*, and each x_i $(i = 1, 2, \ldots, D)$ can be real-valued, integer-valued or boolean-valued. No assumptions in terms of linearity or non-linearity of f_i, g_i and h_j are made.

Many combinations of decision variables in the domain $\mathbb{R}^D$ form solutions in the domain $\mathbb{R}^K$. This is illustrated in Figure 1.1, for $D = 2$ and $K = 2$ (Coello Coello *et al.*, 2007). The multi-objective optimisation problem is solved if a vector $\mathbf{x}^* = [x_1^*, x_2^*, \ldots, x_D^*]^T$ is found which satisfies the $M + Q$ constraints g_i and h_j while minimising $\mathbf{f}$. This set of solutions forms the **Pareto set** of Pareto-optimal solutions, and formal definitions will be presented in Subsection 2.2. These solutions can be shown graphically as the **Pareto front**.

The main task in MOO is to find the Pareto-optimal solutions, or the Pareto front (Coello Coello *et al.*, 2007; Deb, 2001). There are many approaches to this problem; many of them are based on for example *metaheuristics*, while other

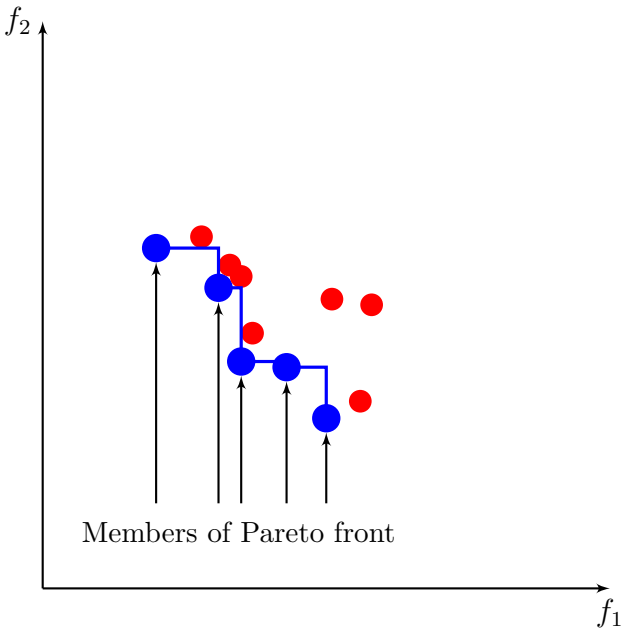

Figure 1.2: Pareto front explained for two minimised objectives.

methods are also used. An example of a Pareto front (blue dots) is shown in Figure 1.2, where both objectives are to be minimised.

The dots in the figure are the result of evaluating both objective functions in terms of a given set of decision variables, each representing a solution vector (f_1, f_2) for a given decision vector (x_1, x_2). Note that a "good" solution method will return dots that are *near*, or better, *on* the Pareto front, but also sufficiently *widely* distributed.

When a decision-making problem with many objectives can only be modelled using computer simulation, each solution vector (represented by the blue and red dots in Figure 1.2) is estimated by means of a simulation run. If the problem is stochastic, such a run can be computationally expensive because the stochastic components ω in (1.3) (the "noise"), must be sufficiently estimated to control the statistical estimation error. Goh & Tan (2007) investigated the effect of noisy environments in evolutionary multi-objective optimisation, and state that for many problems, the evolutionary optimisation process degenerates into a random search when the noise level in a problem increases. Estimation errors (and by implication the choice of sample size) and outliers contribute to a slower convergence rate and possibly sub-optimal solutions. Finding the Pareto front in MOO is generally a

difficult task, and a stochastic component makes it even harder.

Lee *et al.* (2010) proposed a method for finding the non-dominated Pareto set for multi-objective simulation models using *Multi-objective Ranking and Selection* (MORS). They consider a set of scenarios (they term it "designs"), each having K independent, normal distributed objectives, and find an optimal allocation of simulation replications to each design through a sequential procedure, the *Multi-objective Optimal Computational Budget Allocation* (MOCBA) algorithm. This algorithm ensures that the Pareto set is found with high confidence and at the least simulation computation expense. The algorithm has proved to be very economical, but the number of scenarios must be known beforehand and must be relatively small.

However, in many MOO problems the Pareto front is unknown when analysis commences, and since the solution space is also potentially very large, the MOCBA algorithm is not generally applicable. To reduce the computational burden and time to obtain results, an efficient algorithm which dictates the search for the Pareto front is desirable. The term "efficient" means that the algorithm must find the Pareto front with as few as possible evaluation trials. Knowles (2006) studied this problem in the context of *wet experiments*, in which very few evaluations are possible. In such experiments, the time required to perform one evaluation of an experiment is of the order of minutes or hours, only one evaluation is possible at a time (parallel work is not possible), no realistic simulator for approximating the evaluation is available, and the total number of evaluations is limited by financial, time or resource constraints. He proposes the *Parameterised Efficient Global Optimisation* (ParEGO) algorithm in this work, and assumes that noise is low, the search landscape is locally smooth but multimodal and the dimensionality of the search space is low-to-medium, among others.

In the study presented in this dissertation, problems with similar characteristics but with *high noise* will be studied using computer simulation. A preliminary literature survey indicated that the *Cross-entropy method* (CEM) for optimisation, developed by Rubinstein & Kroese (2004), converges fairly fast when performing single-objective optimisation. This leads to the question: can the cross-entropy method be adapted for multi-objective optimisation, and will it still converge fast? It is presumed that if such an adaptation can be made, then solutions to multi-objective stochastic problems can be obtained with a relatively low computational effort and in acceptable time.

The research hypothesis is based on these considerations and is presented next.

1.2 The research hypothesis

The research hypothesis considered in this study is:

> **The cross-entropy method reduces the computational burden when applied to multi-objective optimisation of dynamic, stochastic processes.**

If the research hypothesis can be substantiated, the contribution to the body of knowledge will be achieved in the following two ways:

1. Extension of the cross-entropy method for *multi*-objective optimisation. Currently, a *single* objective is formulated for each of the cross-entropy-based problems found in the literature.

2. Application of the cross-entropy method in *dynamic*, *stochastic* processes. The emphasis here is on the word *dynamic*, which refers to processes that evolve over time such as in for example a manufacturing process. The term *stochastic* refers to the statistical variation in such processes. If the cross-entropy method converges fast in the multi-objective case, then optimisation problems in this domain with a large computational burden can be studied.

The **aim** of the research and the **objectives** pursued serve to support the hypothesis, and these are discussed next.

1.3 Aim and objectives

The research aim, which is the macropurpose of the study (Muller, 2008), is to demonstrate that the cross-entropy method can be used in multi-objective optimisation, with application to dynamic, stochastic processes, in the context of the industrial engineering problem domain. Problems that include constraints are implied.

The research objectives are the specific research tasks that need to be performed (Muller, 2008), which are:

1. Review the literature.

2. Determine if the CEM can speed up the evaluation of objective functions of dynamic, stochastic processes.

3. Determine if the Pareto front for a given problem can be approximated economically in terms of computational effort and time.

4. Determine if the Pareto fronts obtained are effective and efficient using appropriate performance quality indicators.

5. Determine if the CEM can be applied to problems with discrete stochastic as well as continuous deterministic decision variables.

The report on the research task execution forms the core of this document. The structure of the document is presented next.

1.4 Structure of the document

This chapter contains a contextual description of MOO and the problem when objective functions have to be evaluated by *time-consuming* means, typically via *computer simulation*. This led to the formulation of a research hypothesis, a research aim and objectives.

In **Chapter 2**, a literature study on multi-objective optimisation is presented. This includes references to methods of multi-objective optimisation, test problems, application areas and the latest research trends in this field.

Chapter 3 contains a description of the *cross-entropy method* (CEM). The theoretical foundation of the method is presented, as well as its formulation and application to optimisation. Some single-objective optimisation studies are included.

The theoretical background and literature surveys culminate in the *development* of the *multi-objective optimisation using the cross-entropy method* (MOO CEM) algorithm, as described in **Chapter 4**. The proposed method is assessed using known benchmark problems from the literature. These are all continuous or piece-wise continuous mathematical functions that exhibit specific characteristics. Four basic quality indicators are provided to judge the quality of the solutions.

Since the aim of the research is to assess the suitability of the MOO method developed with respect to *dynamic*, *stochastic problems*, the classical stochastic inventory problem of a single commodity is studied in **Chapter 5** as a first and fairly simple problem of this nature. Further applications in buffer allocation in queueing networks, a reconfigurable manufacturing system, and a polymer extrusion unit are also reported. Each of these application descriptions contains

focused literature surveys, problem descriptions, quality indicators, results and conclusions. This structure was followed since some of these application studies were also submitted for publication in research journals. In a final experiment, a dynamic stochastic process at a heavy minerals mining operation was studied.

The quality performance of the proposed algorithm is compared to that of two commercially available products, and an extensive experiment is presented in **Chapter 6**. This experiment serves to provide evidence that supports the research hypothesis.

The summary and general conclusions of the research are presented in **Chapter 7**. The chapter and this study are concluded with some philosophical remarks. Graphical test results are included in **Appendix A**, **Appendix B** and **Appendix C**, and algorithm implementation guidelines are outlined in **Appendix D**.

This concludes **Chapter 1**; the scholarly overview on multi-objective optimisation is presented next.

CHAPTER 2

MULTI-OBJECTIVE OPTIMISATION: OVERVIEW OF SCHOLARLY LITERATURE

Information has become accessible to humanity from almost any place on the planet, and a large part of the collective information base is free. With that in mind, a brief overview of the scholarly literature on multi-objective optimisation (MOO) is presented in this chapter. The aim is to provide the reader with pointers to the major topics in the field, which include a short development history, some cornerstone definitions used in the field that are required for better understanding, and a discussion of the various algorithms or approaches used to perform MOO. These include the ubiquitous evolutionary algorithms and other metaheuristics like simulated annealing and particle swarm optimisation. Hyperheuristics are discussed in a section of its own. Each MOO approach is presented according to a common micro-structure, as far as possible: a very brief outline of the mechanism of each algorithm, how it was adapted to MOO (where applicable), a few applications and, where available, reference to recent survey(s) and relevant textbooks.

Topics like ranking of solutions, fitness assignment, proximity and diversity of solutions, test problems for multi-objective optimisation algorithms and test indicators for algorithm performance are discussed under evolutionary algorithms, but these are also applicable to other approaches. Some applications of MOO algorithms are discussed in general, but also specifically in the domain of Industrial Engineering and Process Engineering.

A summary at the end of the chapter includes the author's views and interpretation of what was observed while doing this overview.

2.1 Introduction to MOO

Practical decision-making requires evaluation of different decision objectives that are conflicting and often measured in different units. An example is an investment decision, where two objectives are present, namely risk and profit. If one wants to increase the profit, one has to accept increased risk, while low risk usually yields low profit. In this decision problem, risk is dimensionless and profit is measured in monetary units. Problems of this nature often have a mutual feature, namely a set of acceptable trade-off solutions. The risk range of the investment problem has an associated profit range, and the decision maker has to choose one of the solutions.

Multi-objective theory originated in the field of economics, and since it is part of economic equilibrium, its origin can be traced back to 1776 when Adam Smith's work *The Wealth of Nations* was published (Coello Coello *et al.*, 2007). Léon Walras introduced the concept of economic equilibrium and Vilfredo Pareto, among others, did important work in this regard between 1874 and 1906. Game strategy is related to multi-objective optimisation and Félix Édouard Émile Borel established *Game Theory* in 1921. The origin of game theory is attributed by most to the famous mathematician and computer scientist John von Neumann who presented work on this topic in 1926, followed by a publication in 1928. Tjalling C Koopmans was the first to apply multi-objective optimisation to domains outside of economics. He worked on production theory and established the concept of an "efficient" vector in 1951 (Koopmans, 1951). The first engineering application seems to be by Lofti Zadeh (Zadeh, 1963). John Buzacott in Lu *et al.* (2009) introduced the term "line-specifc output curve" for production lines in 1967.

The work by Harold W Kuhn and Albert W Tucker in 1951 in the context of the *vector maximum problem* laid the mathematical foundation of multi-objective optimisation (Kuhn & Tucker, 1951). The *Kuhn-Tucker Conditions for Non-inferiority* are often applied in research papers such as Kleijnen & Wan (2007). A further significant development was the introduction of *Goal Programming* by Abraham Charnes and William Cooper (Coello Coello *et al.*, 2007).

The search and optimisation techniques developed over the past decades to solve decision problems are classified by Coello Coello *et al.* (2007) into three main categories: enumerative, deterministic and stochastic (p. 21). Deterministic approaches include greedy, hill-climbing, branch-and-bound, depth-first, breadth-first, best-first and calculus-based algorithms. These algorithms have been successfully

applied to many problems, but they have drawbacks. Generally, the presence of local optima, discontinuities, plateaus and ridges in the solution space reduces algorithm effectiveness. Problems can be discontinuous, high-dimensional, multi-modal and/or NP-complete. A problem with one or more of these properties is called *irregular* (Coello Coello *et al.*, 2007), and many real-world scientific and engineering problems are irregular. Deterministic algorithms, when applied to this problem type, often suffer due to their requirement for problem domain-specific knowledge to direct or limit their search.

Stochastic search and optimisation methods such as Simulated Annealing (SA), Tabu Search (TS), Monte Carlo Methods (MCM) and Evolutionary Computation (EC), were developed to address irregular problems. EC is a generic term for those algorithms that computationally imitate the natural evolutionary process. Specifically, *Evolutionary Algorithms* (EAs) include the techniques of *Genetic Algorithms* (GAs), *Evolution Strategies* (ESs) and *Evolutionary Programming* (EP) (Coello Coello *et al.*, 2007). Stochastic methods provide good solutions to a wide range of problems, but their results cannot be guaranteed to be optimal. The decision maker can only assume the results are *near-optimal*.

2.2 Definitions used in MOO

In MOO there is usually no single optimal solution, but rather a set of good solutions which form the Pareto optimal front (Gil *et al.*, 2007). Rosen *et al.* (2007) provide a good overview of literature in this field. The terms *Pareto front*, *Pareto optimal* and *dominance* have been used before, and these and other terms are now formally defined.

Definition 1: Decision variables: *The vector* $\mathbf{x} = [x_1, x_2, \ldots, x_D]^T$ *of variables for which numerical quantities are to be chosen in the optimisation problem.*

Restrictions are often imposed on an optimisation problem due to practical requirements, which must be satisfied for a solution to be acceptable. The constraints define the dependencies among decision variables and problem parameters (constants). The M inequality constraints are described by

$$g_i(\mathbf{x}) \leq 0,\ i = 1, \ldots, M \tag{2.1}$$

and the Q equality constraints by

$$h_j(\mathbf{x}) = 0,\ j = 1, \ldots, Q. \tag{2.2}$$

The *degrees of freedom* is given by $M - Q$, and it is required that $Q < M$ to avoid an *overconstrained* problem.

The MOO problem with K objectives and $M + Q$ constraints was formulated in Chapter 1 and is repeated here (Tsou, 2008):

$$\begin{aligned} \text{Minimise } \mathbf{f}(\mathbf{x}) &= [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_K(\mathbf{x})]^T && (2.3)\\ \text{subject to } \mathbf{x} &\in \Omega && (2.4)\\ \Omega &= \{\mathbf{x} \mid g_i(\mathbf{x}) \leq 0,\ i = 1, 2, \ldots, M; && (2.5)\\ h_j(\mathbf{x}) &= 0,\ j = 1, \ldots, Q\}. && (2.6) \end{aligned}$$

In multi-objective optimisation, two Euclidian spaces are considered:

1. In the D-dimensional space in which each coordinate axis corresponds to a component of the vector $\mathbf{x}$.

2. In the M-dimensional space in which each coordinate axis corresponds to a component of the objective function vector $\mathbf{f}(\mathbf{x})$.

Since MOO problems usually have at least two conflicting objectives, many acceptable solutions for a given problem exist. These form the *Pareto optimal set*. A few definitions pertaining to Pareto optimality are necessary, and the basic definitions in Coello Coello (2009) are repeated here for convenience (assuming minimisation):

Definition 2: Given two vectors $\mathbf{u} = (u_1, \ldots, u_K)$ and $\mathbf{v} = (v_1, \ldots, v_K) \in \mathbb{R}^K$, then $\mathbf{u} \leq \mathbf{v}$ if $u_i \leq v_i$ for $i = 1, 2, \ldots, K$, and $\mathbf{u} < \mathbf{v}$ if $\mathbf{u} \leq \mathbf{v}$ and $\mathbf{u} \neq \mathbf{v}$.

Definition 3: Given two vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{R}^K$, then $\mathbf{u}$ dominates $\mathbf{v}$ (denoted by $\mathbf{u} \prec \mathbf{v}$) if $\mathbf{u} < \mathbf{v}$.

Definition 4: A vector of decision variables $\mathbf{x}^* \in \Omega$ (Ω is the feasible region) is *Pareto optimal* if there does not exist another $\mathbf{x} \in \Omega$ such that $\mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}^*)$.

Definition 5: The *Pareto optimal set* $\mathcal{P}^*$ is defined by $\mathcal{P}^* = \{\mathbf{x} \in \Omega \mid \mathbf{x} = \mathbf{x}^*\}$.

Definition 6: The *Pareto front* $\mathcal{P}_T^*$ is defined by $\mathcal{P}_T^* = \{\mathbf{f}(\mathbf{x}) \in \mathbb{R}^K \mid \mathbf{x} \in \mathcal{P}^*\}$. The vectors in $\mathbf{P}^*$ are called *nondominated*, and there is no $\mathbf{x} \in \Omega$ such that $f(\mathbf{x})$ dominates $f(\mathbf{x}^*)$.

Solving an MOO problem requires that the Pareto optimal set be found from the set of all decision variable vectors that satisfy constraints (2.1) and (2.2).

With these definitions in mind, the focus now changes to some MOO algorithms and some of their properties.

2.3 Evolutionary algorithms and MOO

Multi-objective Optimisation using Evolutionary Algorithms (MOEAs) has been widely used and actively researched over the past 25 years (see Coello Coello *et al.*, 2007:64). The best-known references are those by Coello Coello *et al.* (2007) and Deb (2001), while a survey of the state of the art of MOEAs was performed by Zhou *et al.* (2011). In a recent article, Coello Coello (2009) highlighted current research trends and open topics in the field of MOEAs, which include a discussion of alternative metaheuristics for solving MOO problems. It is also noted that there is much focus on designing MOEAs that reduce the number of objective function evaluations, because these evaluations can be very expensive when solving some real-world optimisation problems.

Some of the topics pertaining to MOEAs are also applicable to other approaches in MOO, for example ensuring proximity and diversity (Subsection 2.3.2), design and use of test functions (Subsection 2.3.3), and the development and use of performance quality indicators (Subsection 2.3.4).

GAs and other biologically inspired metaheuristics (e.g. Ant Colony and Particle Swarm Optimisation) have been widely applied in solving MOO problems. Arguably the best-known evolutionary-based algorithms are the *Multi-objective Genetic Algorithm* (MOGA) of Fonseca & Fleming (1993), the *Niched-Pareto Genetic Algorithm* (NPGA) of Erickson *et al.* (1999), the *Strength Pareto Evolutionary Algorithm* (SPEA) of Zitzler & Thiele (1999), the *Pareto Archived Evolution Strategy* (PAES) of Knowles & Corne (2000), the *Multi-objective Messy Genetic Algorithm* (MOMGA) of Van Veldhuizen & Lamont (2000), the *Pareto Envelope-based Selection Algorithm* (PESA) of Corne *et al.* (2000) and the *Non-dominated Sorting Genetic Algorithm* (NSGA-II) of Deb *et al.* (2002). These algorithms and some of their variants are discussed in Coello Coello *et al.* (2007).

EAs are widely used in MOO research and applications. Beausoleil (2006) applies a *Multiple-objective Scatter Search* (MOSS) to test problems from the literature, and Deb *et al.* (2002) improve on existing algorithms with their NSGA-II. This algorithm includes a dominance principle, diversity preservation principle and elite preserving principle, and is currently the most widely used algorithm. In

Coello Coello *et al.* (2004), they apply particle swarm optimisation while incorporating Pareto dominance. Summanwar *et al.* (2002) solve constrained optimisation problems using MOGAs, while Zitzler & Thiele (1999) apply the SPEA to the 0/1 knapsack problem. Gil *et al.* (2007) developed a hybrid method for solving MOO problems by combining PESA and NSGA-II, for example.

In other applications, specific methods are developed to solve MOO problems. For example, Lee (2007) developed a trajectory-informed search methodology and applies it to several test problems. The Adaptive Range Multi-objective Genetic Algorithm (ARMOGA) of Sasaki & Obayashi (2005) requires relatively few objective function evaluations to find the approximate Pareto front. The ParEGO algorithm of Knowles (2006) was mentioned in Section 1.1 in this context: few evaluations are performed because of very tight resource constraints. This algorithm uses a normalised objective function set, and so the range of each objective must be known.

Chapter 7 in Coello Coello *et al.* (2007) contains comprehensive references to applications in engineering, science, industry and miscellaneous fields (e.g. investment portfolio optimisation and stock ranking). A summary of applications of MOEA is also provided in Zhou *et al.* (2011). This includes scheduling, data mining, assignment and management, communication, bio-informatics, control systems and robotics, image processing, artificial neural networks, manufacturing, traffic and transportation, and others. A comprehensive list of references is also maintained at the Evolutionary Multi-objective Optimisation (EMOO) home page (`www.lania.mx/~ccoello/`, cited on 10 August 2012).

Current research trends in evolutionary MOO are discussed by Coello Coello (2009). He notes that researchers focus on new algorithms, efficiency, relaxed forms of dominance, scalability and alternative metaheuristics. Researchers propose *new algorithms* but only some became widely used, as was pointed out earlier in this section.

The term *efficiency* refers to algorithm design which reduces the number of instructions performed. This is typically attempted by aiming to make the ranking algorithm more efficient and reduce the number of objective function evaluations. This is also an objective of the research presented in this dissertation, as was motivated in Chapter 1. The *relaxed forms* of Pareto dominance attempt to regulate convergence, and ϵ-dominance is perhaps the most popular of those. A set of boxes is supposed to cover the Pareto front, and the box size is determined by

the user-defined parameter ϵ. Only one non-dominated solution is allowed within each box, and a large value of ϵ speeds up convergence, but the quality of the Pareto front might suffer as a result. A small value of ϵ results in a high-quality Pareto front obtained at the cost of convergence speed. Choosing the value of ϵ is still an open problem, also when nothing is known about the true Pareto front, which is the case in practical problems.

MOO algorithms are almost always sensitive to *scalability*, as they do not automatically scale to problems with many objectives. It was shown that the proportion of non-dominated solutions increases proportionally with the number of objectives (Purshouse & Fleming, 2007).

Apart from genetically inspired algorithms, there are *alternative* biologically inspired metaheuristics like artificial immune systems, ant colony optimisation and particle swarm optimisation. Non-biologically inspired algorithms include simulated annealing, tabu search and scatter search. The algorithm proposed in this dissertation is of non-biological nature and is based on statistical principles.

Coello Coello (2009) recommends that constraint-handling, incorporation of users' preferences and parameter control be further researched in future work. The idea with parameter control is that the MOEA adapts its parameters automatically without user-intervention. Incorporating user preferences may render MOO more suitable to practical problems, and algorithms may even become more efficient since preferences may reduce the problem size (solution space).

2.3.1 Fitness assignment and ranking of solutions

An EA has both *objective* and *fitness* functions associated with it. The values of the objective functions give an indication of attainment of the various optimality criteria, while the fitness function assumes a real value, indicating how well a particular set of objective function values satisfy the optimality condition (Coello Coello *et al.*, 2007). A population has to be ranked to distinguish good solutions from bad ones, and the fitness values are used for this purpose.

The best-known ranking method is the Pareto ranking based on work by Goldberg (1989), which is of complexity $O(KN^2)$, where N is the user-specified population size. Faster algorithms have been developed by Qu & Suganthan (2009) and Fang *et al.* (2008). A new fast sorting algorithm by Mishra & Harit (2010) has worst-case complexity of $O(KN^2)$ and best-case complexity of $O(N \log N)$. In an application Wang & Yang (2009) developed a particle swarm optimisation

algorithm using the preference order scheme (Das, 1999; Pierro *et al.*, 2007) which is more efficient than the Pareto ranking particularly when the number of objectives is large. D'Souza *et al.* (2010) improved the NSGA-II by reducing the time complexity through a better ranking scheme.

Jaimes *et al.* (2009) present a comparative study of several ranking methods, and also provide a useful taxonomy of ranking methods. This includes ranking methods with and without parameters, favour ranking, preference order ranking and Pareto ranking. They have found that the preference order ranking method achieves the best scalability, while different ranking methods produce different subsets of the Pareto optimal set. The quality of the solutions produced by an MOO algorithm may thus be affected by the ranking method selected.

2.3.2 Proximity and diversity

A good MOO algorithm ensures that the Pareto approximation set is close to the true front, and that it is also well populated with solutions. It is expected of an algorithm to properly *explore* and *exploit* the solution space in order to fulfil these two requirements. Laumanns *et al.* (2002) have developed the concept of ϵ-dominance and constructed updating strategies for iterative searches that allow for the desired convergence and distribution of solutions. Finding a close and dense Pareto front approximation is in itself a multi-objective problem, as seen in the performance of MOEAs (Bosman & Thierens, 2003). Wang *et al.* (2010) have proposed a crowding entropy diversity measure for a self-adaptive differential evolution algorithm called MOSADE. Their algorithm performed better than NSGA-II, SPEA2 and *Multi-objective Particle Swarm Optimisation* (MOPSO) on 18 different test problems, measured in terms of convergence and diversity.

Purshouse & Fleming (2007) showed that the behaviour of MOEAs change with increasing numbers of conflicting objectives. The configuration of an algorithm for few objectives cannot necessarily be generalised to larger numbers of objectives, and they found that diversity-promoting mechanisms can be highly influential and even harmful to the optimisation outcome. Also, *dominance resistance*, the phenomenon which makes it difficult to produce new solutions that will dominate poor solutions, also contributes to preserving locally non-dominated solutions, which in turn confines diversity. Other researchers (Purshouse & Fleming, 2007) have confirmed that dominance resistance may increase with increasing solution space. Purshouse and Fleming suggested that the non-dominated set be pruned

on a solution-by-solution basis to reduce the dominance-resistant solutions. Hájek *et al.* (2010) developed a mechanism to improve diversity and implemented it with the μARMOGA of Szőllős *et al.* (2009).

Good proximity and diversity create confidence in the approximation solution set. Test problems and quality indicators to assess the ability of an algorithm to achieve these are the topics of the next two sections.

2.3.3 Test problems for MOO

Several standard MOO test problems with known Pareto fronts are proposed in the literature. These have been consolidated in Chapter 4 of the book by Coello Coello *et al.* (2007). These test problems were designed to embody a mixture of non-linear, time-independent and deterministic properties, two or more objective functions, disconnected and asymmetric regions in solution space, and a mixture of concave and convex Pareto front shapes. Some of the test functions are listed in Table 2.1 and are referred to in the literature as belonging to the Van Veldhuizen test suite (Veldhuizen, 1999) (MOP1–MOP6), while ZDT1–ZDT3 were developed by Zitzler *et al.* (2000) (see Table 2.1). Test problems and their requirements were analysed in detail by Huband *et al.* (2006) who proposed test problems in the Walking Fish Group (WFG) Toolkit. Igel *et al.* (2007) developed the IHR test suite which allows for testing if an algorithm is invariant against rescaling and rotation. Problems should be non-separable in general, have no extremal parameters and have a scalable number of parameters and objectives, to name a few.

The test problems selected for this study all have known true Pareto fronts $\mathcal{P}_T$ and may be obtained from the EMOO home page (`www.lania.mx/~ccoello/`, cited on 10 August 2012). These test functions will be used for the algorithm assessment described in Chapter 4. Quality indicators which assume numeric values exist to evaluate the quality of the solutions generated compared to the known solutions. Some of these are discussed next.

2.3.4 Quantifying the performance of MOO algorithms

Several quality performance indicators for MOO algorithms exist. They typically consider some estimation of the deviation between the approximate front found by the test algorithm ($\mathcal{P}_K$) and a true Pareto front ($\mathcal{P}_T$) of a benchmark problem, as depicted in Table 2.1. The term *quality performance* will be used in this dissertation instead of the general term *performance*, because the latter pertains to *time* and

Function	Definition			Constraints
MOP1	$f_1(x)$	=	x^2	$-10^5 \leq x \leq 10^5$
(Min)	$f_2(x)$	=	$(x-2)^2$	
MOP2	$f_1(\mathbf{x})$	=	$1-\exp(-\sum_{i=1}^{n}(x_i-\frac{1}{\sqrt{n}})^2)$	$-4 \leq x_i \leq 4$
(Min)	$f_2(\mathbf{x})$	=	$1-\exp(-\sum_{i=1}^{n}(x_i+\frac{1}{\sqrt{n}})^2)$	$i=1,\ldots,n,\ n=3$
MOP3	$f_1(x,y)$	=	$-[1+(A_1-B_1)^2+(A_2-B_2)^2]$	$-\pi \leq x,y \leq \pi$
(Max)	$f_2(x,y)$	=	$-[(x+3)^2+(y+1)^2]$	$A_1=0.5\sin 1-2\cos 1+\sin 2-1.5\cos 2$, $A_2=1.5\sin 1-\cos 1+2\sin 2-0.5\cos 2$, $B_1=0.5\sin x-2\cos x+\sin y-1.5\cos y$ $B_2=1.5\sin x-\cos x+2\sin y-0.5\cos y$
MOP4	$f_1(\mathbf{x})$	=	$\sum_{i=1}^{n-1}(-10\exp(-0.2)\sqrt{x_i^2+x_{i+1}^2})$,	$-5 \leq x_i \leq 5$
(Min)	$f_2(\mathbf{x})$	=	$\sum_{i=1}^{n}(\lvert x_i\rvert^a+5\sin(x_i)^b)$	$i=1,2,3,\ a=0.8,\ b=3$
MOP6	$f_1(x,y)$	=	x	$0 \leq x,y \leq 1$
(Min)	$f_2(x,y)$	=	$(1+10y)[1-(\frac{x}{1+10y})^{\alpha}-\frac{x}{1+10y}\sin(2\pi q x)]$	$q=6,\ \alpha=2$
ZDT1	$f_1(\mathbf{x})$	=	x_1	$0 \leq x_i \leq 1,\ n=30$
(Min)	$f_2(\mathbf{x},g)$	=	$g(\mathbf{x})\cdot(1-\sqrt{f_1/g(\mathbf{x})})$	
	$g(\mathbf{x})$	=	$1+\frac{9}{n-1}\cdot\sum_{i=2}^{n}x_i$	
ZDT2	$f_1(\mathbf{x})$	=	x_1	$0 \leq x_i \leq 1,\ n=30$
(Min)	$f_2(\mathbf{x},g)$	=	$g(\mathbf{x})\cdot(1-(f_1/g(\mathbf{x}))^2)$	
	$g(\mathbf{x})$	=	$1+\frac{9}{n-1}\cdot\sum_{i=2}^{n}x_i$	
ZDT3	$f_1(\mathbf{x})$	=	x_1	$0 \leq x_i \leq 1,\ n=30$
(Min)	$f_2(\mathbf{x},g)$	=	$g(\mathbf{x})\cdot(1-\sqrt{f_1/g(\mathbf{x})}-f_1/g(\mathbf{x})\cdot\sin(10\pi f_1))$	
	$g(\mathbf{x})$	=	$1+\frac{9}{n-1}\cdot\sum_{i=2}^{n}x_i$	

Table 2.1: Some of the standard MOO test functions used for evaluation of MOO algorithms.

quality, while only quality is assessed, as will be shown later in this chapter as well as in Chapter 6. The time quality indicator is assessed by implication, as the number of objective evaluations is restricted for each problem studied in this research. Also, the term *quality performance indicator* is used in this dissertation, while terms like "performance measure" and "performance metric" are used in the literature. According to Knowles *et al.* (2006) the term "indicator" is more correct, since "measure" and "metric" have specific mathematical meanings.

Two approaches may be followed to quantify quality of performance: the indicator approach quantifies the result produced by an algorithm as a numerical value, while the attainment approach models the outcome as a probability density function (Knowles *et al.*, 2006). The indicators assume unary numerical values, which often include some subjectivity. Research to propose performance indicators has been conducted since the 1990s, and is consolidated in Coello Coello *et al.* (2007). The indicators are mainly used to obtain the measures of quality of convergence and diversity (see Subsection 2.3.2). A good approximation set must be close to or, even better, coincide with the true front, while it must also extend uniformly and be well populated over the front (Purshouse & Fleming, 2007).

Zitzler *et al.* (2002) and Zitzler *et al.* (2003) provide theoretical depth of performance or quality assessment of MOO algorithms. Common to both works is that proposed quality indicators only allow certain conclusions when testing one or more algorithms, and at best, one can in general state that "Approximation set A is better than approximation set B". Such a statement is based on numerical values which were estimated for one or more quality indicators, and one can only conjure *whether* or not A is better, *but not by how much.* In this dissertation the recommendations of the various researchers are accepted and followed without reporting on their works in detail.

Some unary performance indicator values for MOO algorithms can be calculated as follows:

1. *Generation Distance* (GD), which measures the average distance between $\mathcal{P}_K$ and $\mathcal{P}_T$. It is defined as

$$GD \triangleq \frac{(\sum_{i=1}^{|\mathcal{P}_K|} d_i^2)^{\frac{1}{2}}}{|\mathcal{P}_K|}, \tag{2.7}$$

   where d_i denotes the Euclidean distance between solution value i of $\mathcal{P}_K$ and the closest member in $\mathcal{P}_T$ to solution i. When $\mathcal{P}_K = \mathcal{P}_T$, then $GD = 0$.

2. *Spacing* (SP), which numerically describes the spread of the vectors in $\mathcal{P}_K$. It is defined as

$$SP \triangleq \sqrt{\frac{1}{|\mathcal{P}_K| - 1} \sum_{i=1}^{|\mathcal{P}_K|} (\overline{d} - d_i)^2}, \tag{2.8}$$

where

$$d_i = \min_j \sum_{k=1}^{K} |f_k^i(\mathbf{x}) - f_k^j(\mathbf{x})| \tag{2.9}$$

with $i, j = 1, \ldots, |\mathcal{P}_K|$, K the number of objectives, and $\overline{d}$ the mean of all d_i. The members of the approximation front are equally spaced if $SP = 0$. The true Pareto front is not required for this test measure.

3. *Maximum Pareto Front Error* (ME), which measures how well two vector sets conform in terms of shape and distance apart. It is determined with

$$ME \triangleq \max_j \left\{ \left\{ \min_i \left(\sum_{k=1}^{M} |f_k^i(\mathbf{x}) - f_k^j(\mathbf{x})|^2 \right)^{1/2} \right\} \right\}. \tag{2.10}$$

4. Deb & Jain (2002) proposed a running indicator for convergence. The value of the indicator is calculated while an MOO algorithm is executed, and $\mathcal{F}(t)$ is the non-dominated set of population t. Then from each point i in $\mathcal{F}(t)$, the smallest normalised Euclidean distance to the true Pareto set $\mathcal{P}_T$

$$d_i = \min_{j=1}^{|\mathcal{P}_T|} \sqrt{\sum_{k=1}^{K} \frac{f_k(i) - f_k(j)}{f_k^{max} - f_k^{min}}} \tag{2.11}$$

is calculated, with f_k^{max} and f_k^{min} being the maximum and minimum function values of objective k in $\mathcal{P}_T$. Now the convergence CV is calculated by averaging the normalised distance for all points in $\mathcal{F}(t)$, that is

$$CV = \frac{\sum_{i=1}^{|\mathcal{F}(t)|} d_i}{|\mathcal{F}(t)|}. \tag{2.12}$$

The indicator value can be normalised on $[0, 1]$ by keeping record of the maximum value of CV, say CV^*, then calculating CV/CV^*. In this research, the indicator is calculated at the end of algorithm execution, i.e. when the final approximation set $\mathcal{P}_K$ is obtained, so in (2.11), $\mathcal{F}(t) = \mathcal{P}_K$.

Other indicators are listed in Appendix A in Knowles *et al.* (2006).

The quality performance indicators are by implication used in post-analysis of algorithm performance, except for the CV indicator of Deb & Jain (2002) which may be used as a running performance evaluation of an algorithm.

Performance indicators have limitations. Some true Pareto fronts are infinite in size, while algorithms return finite solution sets, and so subsets of the true fronts must be used for comparison. When a true front is continuous, it must be divided into discrete units for comparison if the obtained approximate front is a discrete, finite set. Both convergence and spread/spacing indicators should be considered when evaluating the solution quality of an algorithm, as a good proximity value does not necessarily imply a good spread, while a dense approximation set may be far from the true front. When evaluating stochastic algorithms, the value of the indicator will vary if different random numbers are used, and single-observed indicator values cannot be used, unless a sufficient sample of indicator outcomes is taken. This requires that small-sample theory be applied for a valid analysis.

When comparing the performance quality of two or more algorithms, the approaches recommended in Knowles *et al.* (2006) should be followed (also see Chapter 6). The quality performance indicators above were discussed in the context of MOEAs, but can be used for other types of algorithms as well. Some will be applied in Chapter 4 and in Chapter 6.

This concludes the discussion of MOEAs. Next, other metaheuristics for solving MOO problems are discussed.

2.4 Other MOO metaheuristics

Other metaheuristics exist for MOO, and since each represents a research field of its own, they are discussed briefly by presenting a conceptual description of their working, who adapted them for MOO, recent surveys (if available) and some applications.

2.4.1 Simulated annealing

Simulated Annealing (SA) was proposed by NC Metropolis, while Kirkpatrick (Kirkpatrick *et al.*, 1983) and Černý (Černý, 1985) independently showed the analogy to combinatorial optimisation (Coello Coello *et al.*, 2007). SA can be applied to arbitrary search and problem spaces and is not population-based; it thus

only needs a single individual as starting point, while a unary search operation works from this point.

Annealing is a technique applied in materials science to alter a property of a material, such as its hardness. Metal, for example, may have dislocations in its structure which weakens the specimen, and by heating it, the energy of the atoms increases and they diffuse, thus destroying the weakness, and while cooling down, the structure is reformed and eventually a state of equilibrium is reached.

The thermodynamic state of a system is defined by a certain temperature and energy, which are related by $\exp \frac{-E_i}{k_B T_s}$ with k_B the Boltzmann constant, T_s the cooling temperature, which is varied, and E_i the energy associated with state i. To mimic annealing during optimisation, a new state $i+1$ is formed, and the energy change ΔE is calculated. The objective function is simply associated with the energy E. The new energy state is $\Delta E = E_{i+1} - E_i$ and accepted or rejected with probability

$$\mathbb{P}(\Delta E) = \begin{cases} e^{-\frac{\Delta E}{k_B T_s}} & \text{if } \Delta E > 0; \\ 1, & \text{otherwise.} \end{cases} \tag{2.13}$$

If the energy of a nearby state is lower (ΔE is negative), the transition will be allowed, otherwise it will be accepted if a uniform random number is less than $\mathbb{P}(\Delta E)$. Initially, the temperature is high and many transitions are accepted, but as the temperature decreases, fewer and fewer transitions are accepted until equilibrium is reached at zero temperature, and it is assumed that the system has come to rest in an optimal state.

For a detailed description and references to applications, see Gendreau & Potvin (2010), Weise (2009), and Coello Coello *et al.* (2007:548–557). A survey of SA in single- and multi-objective optimisation has been performed by Suman & Kumar (2006), while Singh *et al.* (2010) studied SA in constrained optimisation. SA can also be used in hybrid algorithm optimisation (Liu *et al.*, 2011). A typical process engineering application is considered by Ruiz-Torres *et al.* (2011) and Sankararao & Kyoo Yoo (2011), while Yu *et al.* (2010) apply SA to the Location Routing Problem (LRP).

2.4.2 Tabu search

Tabu Search (TS) was developed by Fred Glover (Glover, 1986; Glover & Laguna, 1997) who is also credited with coining the term "metaheuristic". TS acts as a local

search procedure and iteratively allows a move to a good neighbouring solution. Moves are allowed even if the neighbour seems unfavourable relative to the current optimisation state. Reverse moves are forbidden to avoid cycling, and these moves are stored in a tabu list. If a given move satisfies a sufficient aspiration criterion, the tabu status may be overridden. An intermediate list stores good solutions that are used to intensify the search, while a long-term list helps to diversify the search. The tabu list has a finite size. If the list is populated with tabu solutions, and a new tabu solution must be recorded, the first ("oldest") solution in the list is removed and the new solution added. TS works well in discrete problems, but has difficulty moving to neighbours when the search space is continuous.

TS has been extended to MOO by the *Multi-objective Tabu Search* (MOTS), and uses a utopian reference point. Each objective function improvement is measured in terms of this point and recorded in the tabu list. This list is later used to update the search direction. Maintaining diversity is a problem since the algorithm is supposed to perform a neighbourhood search, but combining it with another algorithm (for example an EA) to form a hybrid seems a natural decision. Of course, the computation penalty must be considered.

TS has been widely applied in MOO (Ghisu *et al.*, 2011; Jaeggi *et al.*, 2008). For details on the TS, see Coello Coello *et al.* (2007:557–572), Weise (2009) and Gendreau & Potvin (2010).

2.4.3 Ant systems

Ant Systems (AS) optimisation was developed by Marco Dorigo (Dorigo, 1992; Dorigo & Stützle, 2004) for the travelling salesman problem. It is based on the activities of real ants seeking food: an ant wanders from the nest and leaves a pheromone trail, and then traces it back to the nest. If food is found, the trail is traversed again while depositing more pheromone – this in turn attracts other ants and the route to the food is well established. The pheromone also evaporates, and so a non-rewarding route eventually ceases to exist. In optimisation, artificial ants move about in a network and deposit pheromone on each link between two nodes. Eventually they tend to follow the same route, which becomes more and more marked with pheromone, while evaporation is allowed to prevent premature convergence. This route is then considered optimal.

AS optimisation was extended to the multi-objective domain by Mariano & Morales (1999) and is called the *Multi-objective Ant-Q* algorithm (MOAQ). In this

algorithm, there is a family of ants for each objective, and each family attempts to optimise its objective. Families exchange solution information and adapt their own objective functions accordingly. An archive of non-dominated solutions is maintained.

AS optimisation has been widely applied in, for example, flow shop scheduling (Yagmahan & Yenisey, 2010), batch planning of hot rolling processes in a steel plant (Shixin Liu, 2010), supply chain design (Moncayo-Martinez & Zhang, 2011) and material development (Hudson *et al.*, 2011). A survey on AS optimisation was conducted by Blum (2005) while a special journal issue was dedicated to the topic (Doerner *et al.*, 2009). The interested reader is referred to Coello Coello *et al.* (2007:572–582), Weise (2009), Gendreau & Potvin (2010) and the AS web site at `http://iridia.ulb.ac.be/~mdorigo/ACO/about.html` (cited on 10 August 2012) for further information.

2.4.4 Particle swarm optimisation

Particle Swarm Optimisation (PSO) was developed by Kennedy & Eberhart (1995) and simulates the movement of a population of birds searching for food. The behaviour of each individual is affected by the locally best or the globally best individual. Initially, each particle in a finite population ("birds") is assigned a random position and velocity, the position being a decision variable value. The particles have some freedom to move around independently, and are allowed to move to new positions. When good positions (food availability) are found, these are communicated to other particles, which tend to move to these positions. The algorithm proceeds stepwise, and the velocity and position of each particle are updated during each step. Each particle keeps its best position in memory, and the overall best is also known to all particles. These values are used to update the velocity of each particle.

PSO is conceptually simple, also simple to implement and has a high convergence rate. In MOO applications, diversity control is difficult, but turbulence operators, which are analogous to mutation, are used to address this problem. Goh *et al.* (2010b) improved the multi-objective PSO (MOPSO) through the *Competitive and Cooperative Co-evolutionary Multi-objective Particle Swarm Optimisation* (CCPSO) algorithm, which addresses the tendency of premature convergence.

Multi-objective PSO has been applied in various domains such as, for example, in project selection (Rabbani *et al.*, 2010) and in the design of green sand moulds

(Surekha *et al.*, 2011). Ali *et al.* (2011) applied MOPSO in an ad hoc network to provide an energy-efficient solution and reduce the network traffic by optimising the number of clusters, as well as energy dissipation in nodes. Also see the Particle Swarm Website at http://www.swarmintelligence.org/ (cited on 10 August 2012), Coello Coello *et al.* (2007:584–594), Olsson (2011) and Panigrahi *et al.* (2011) for further information.

2.4.5 Hill-climbing techniques

Hill climbing is an old optimisation method for single-objective functions. A single solution is initially created, followed by an offspring. If the offspring is better than the parent, it is accepted as the new parent, otherwise it is rejected. The method often converges prematurely and is thus easily trapped in the region of a local optimum. It has been extended to MOO by operating on a set of solutions instead of a single value as in the case of single-objective optimisation (Weise, 2009). Some mechanisms of EAs are used in this technique, for example when selecting the best solutions to determine the parents of the current step. Suggestions to overcome problems include: 1) maintaining a tabu list to prevent premature convergence, 2) sometimes rejecting the new solution, as is done by simulated annealing, and 3) randomly restarting the algorithm after a predetermined number of steps (Weise, 2009).

2.4.6 Distributed reinforcement learning

The concept of Q-learning (Watkins & Dayan, 1992) has been extended to MOO by Mariano & Morales (2000) which they call *Multi-objective Distributed Q-learning* (MDQL). To each objective of a problem, a family of agents is assigned, which interact in a common environment of states and actions and cooperate towards a common goal. The agents provide solutions, and a map is built based on the update of a value function defined for the optimisation problem. An update occurs when an agent visits a state and selects an action, thus leaving a trace for other agents. An agent decides its next move based on the traces left by other agents. The solutions of one family of agents are compared with those of other families via a negotiation mechanism. This mechanism produces members for the approximate Pareto set, and those states which provided such solutions are rewarded.

A drawback of the MDQL is that it requires a discrete decision variable space, which limits its application. It maintains the convergence properties of Q-learning

and uses a relatively easy penalty approach to handle constraints.

Mariano and Morales used MDQL to solve problems relating to water-distribution systems. Also see Coello Coello *et al.* (2007:582–584).

2.4.7 Differential evolution

Differential Evolution (DE) was developed by Storn & Price (1997) for continuous optimisation. It is an evolutionary algorithm and shares many similarities with traditional EAs (Coello Coello *et al.*, 2007). It does not use binary encoding and the parameters are updated through mutation using the distribution of solutions in the given current population. At least eight DE variants are available for single-objective optimisation in the literature.

Also, many variants exist for MOO: the *Non-dominated Sorting Differential Evolution* (NSDE) algorithm of Iorio & Li (2004) is a simple modification of the NSGA-II because only the operators of the NSGA-II are replaced by DE operators. This algorithm can solve rotated problems. Robič & Filipič (2005) developed the notion of *Differential Evolution for Multi-objective Optimisation* (DEMO), which combines DE with Pareto ranking and crowding distance sorting. Pedersen (2010) studied meta-optimisation in which algorithm parameters are varied to improve the algorithm performance. In this study, he used DE and PSO for his evaluations. Lee *et al.* (2011) adapted the DE algorithm to improve a surface grinding process by controlling the variables such as wheel speed, workpiece speed, depth of dressing, and lead of dressing, while adhering to various constraints. The objectives were to minimise production cost, maximise production rate and produce high-quality surface finish.

Qin *et al.* (2010) developed a DE algorithm to assist with reservoir flood control operation at the Three Gorges Project in the Yangtze River, China. Reservoir flood control attempts to minimise flood peaks by utilising the flood storage capacity of reservoirs. Several similar applications of DE in reservoir management can be found in the literature (Reddy & Kumar, 2007).

DE is powerful but perhaps too fast in terms of convergence. Also see Coello Coello *et al.* (2007:594–604), an earlier survey by Mezura-Montes *et al.* (2008) and an account of recent advances in DE by Neri & Tirronen (2010).

2.4.8 Artificial immune systems

The immune system of an organism protects it against detrimental biological threats, e.g. foreign elements like viruses and pathogens. It forms antibodies against the foreign pathogenes and tries to eliminate these to ensure survival of the organism. The immune system has memory and can retrieve previous knowledge of defensive actions. The basic optimisation algorithm based on the mechanism of the immune system is attributed to Bersini & Varela (1991). A population of antigens and one of antibodies are formed. These are matched and a fitness value is assigned to each antibody that is a good match for an antigen, where the higher fitness values imply good antibodies. A conventional genetic algorithm is used to create more antibodies which, in the end, should outmatch the antigens.

Artificial immune systems require simple algorithms and can be used to maintain diversity in the genetic algorithms for multimodal optimisation problems. Coello Coello & Corts (2005) present the *Multi-objective Immune System Algorithm* (MISA) and compare it to the microGA, PAES and NSGA-II, using the error ratio, spacing and inverted generation distance indicators (see Subsection 2.3.4). Campelo *et al.* (2007) gives an overview of artificial immune systems for MOO, while Bernardino & Barbosa (2009) discuss artificial immune systems on a wider basis, which includes the case of single-objective optimisation. Applications of artificial immune systems are reported by Luh & Chueh (2004), who applied the *Constrained Multi-objective Immune Algorithm* (CMOIA) to six test functions and two well-known truss sizing optimisation problems. Tavakkoli-Moghaddam *et al.* (2007) developed a hybrid algorithm, called *Hybrid Multi-objective Immune Algorithm* (HMOIA), which uses the principles of artificial immune systems, but uses a Tabu Search to construct the initial set of antibodies. Flow shop scheduling problems are studied in which the weighted mean completion time and weighted mean tardiness are minimised. Also refer to Coello Coello *et al.* (2007:604–612), and Gendreau & Potvin (2010) for further information and applications of artificial immune systems.

2.4.9 Evolution strategy

Evolution Strategy (ES) was created by Rechenberg, Schwefel and co-workers (Beyer & Schwefel, 2002). The basic ES operates on a population of two members only. The first is a parent and the second a mutant of the parent. If the mutant is "better" than the parent, it becomes the parent of the next generation, otherwise it

is rejected. This approach in which the population has only one member, is similar to hill-climbing described in Subsection 2.4.5, but variants of the ES allow for more than two members in a population. For example, in the $(n+1)$-ES variant, the population consists of $n+1$ members, and one member is drawn randomly. The member is reproduced using the existing population and its offspring, and the least fit individual is removed after each iteration.

The *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) is a powerful version of ES for optimisation in non-linear non-convex problems in the continuous domain. It is efficient in large problem spaces and when the search landscape is rugged, but second-order derivative methods can be better in conjunction with for example convex quadratic functions. The algorithm controls both the probabilities of successful solution candidates and search steps.

The CMA-ES has been extended to MOO by Igel *et al.* (2007). The general algorithm is called *Multi-objective Covariance Matrix Adaptation Evolution Strategy* (MO-CMA-ES), of which two variants are presented. The first, c-MO-CMA, uses crowding distance as second-level sorting criterion, while s-MO-CMA uses the contributing hypervolume. A variety of benchmark problems from the literature are empirically evaluated and the results compared with those uncovered by the NSGA-II, with s-MO-CMA performing superior in almost all cases.

Considering practical applications, Paly *et al.* (2010) studied MOO to estimate crop production functions. Such a function typically shows the expected yield of a certain crop, for example maize, in relation to the irrigation depth. The latter is concerned with the amount of water supplied to the crop, and since water is a scarce commodity, its use must be efficient and minimised. The authors compared four algorithms, namely NSGA-II, NSDE (a rotation invariant multi-objective version of differential evolution), DEMO (see Subsection 2.4.7), and MO-CMA-ES. Three scenarios were studied which differ in terms of crop type (maize and potato) and climate. The conclusion was that MO-CMA-ES is the best performing algorithm, with DEMO also performing well.

In engine calibration, experimenters seek to find an optimal tuning of engine parameters to be used in engine control. Parameter values include fuel injection pressure, mass of air flow and boost pressure. At a given operating point and engine speed, NO_x emissions, as well as CO_2, HC and CO emissions, must be minimised while the engine noise must be below a certain level. Langouët *et al.* (2011) studied this real-world problem by means of the MO-CMA-ES and claimed

that worthy information on antagonistic engine responses were obtained, which include optimal engine maps.

2.4.10 Memetic algorithms

The *Memetic Algorithm* (MA) was introduced by Moscato (1989). The term "meme" is defined as "an element of culture that may be considered to be passed on by non-genetic means" (English Oxford Dictionary). In MAs, population-based searches for solutions are complemented by non-genetic local optimisations. Human beings in their daily interaction exchange information which affects the recipients of such information, and the concept of the meme is to model the transfer of one unit of information in a population. The information is not genetically inherited but transferred by imitation.

Lakshmi & Rao (2010) present the *Shuffled Frog-leaping Algorithm* (SFLA), which imitates the memetic evolution of a group of frogs when they search for a location with maximum food supply. The authors use this to optimise the lay-up sequence of laminate composite structures. The convergence rate is improved via the customised neighbourhood search algorithm and an adaptive search factor. The SFLA outperforms the NSGA-II, PAES and microGA.

Frutos *et al.* (2010) developed an MA to address the flexible job-shop scheduling problem by using the NSGA-II as basis for a population search, and simulated annealing for the local search. They minimise the total makespan and the operating cost.

Chen & Chyu (2010) study the question of how to assign a limited number of resources to a large number of potential projects in terms of funding. The return on investments is maximised while investment is minimised in a case study involving six projects and 120 units of capital. Their MA consists of a GA combined with a local search which transfers small numbers of units of capital between pairs of projects.

MAs are also used in conjunction with artificial neural networks. Almeida & Ludermir (2010) propose a combination of ESs, PSO and concepts from GAs to better estimate the initial weights, number of hidden nodes and layers, training algorithm rates and transfer functions. These are usually selected by a manual process of trial-and-error which can be detrimental to the final solution obtained. In their experiments, they applied an MA to standard supervised classification

problems and report that artificial neural networks with a lower number of nodes were obtained, but execution time was a disadvantage.

Burke *et al.* (2010) applied an MA to develop more robust airline flight schedules, that is schedules which are less likely to be delayed. Objectives are improved through flight retiming and aircraft rerouting, subject to a fixed fleet assignment. They conducted a real-world study at KLM Royal Dutch Airlines and estimated the operational performance of the improved schedules with a large-scale simulation study. Chiam *et al.* (2009) developed an MA using a PSO algorithm as a local optimiser of an EA. The MA is applied in the financial domain to do portfolio optimisation and time series forecasting.

In dynamic or robust MOO, the objectives change over time. Isaacs *et al.* (2008) solve such problems by embedding a sequential quadratic programming (SQP) solver in an algorithm that uses artificial neural networks. The MA is illustrated using two test functions developed by Farina *et al.* (2004) and a real-time problem. In the latter, smaller artificial neural networks are identified online to assist with flight control of a fixed-wing, six-degree-of-freedom unmanned aerial vehicle (UAV). Training the neural network faster makes the control system more adaptable to the dynamic behaviour of the UAV during flight.

2.4.11 Firefly algorithm

The firefly algorithm is a population-based method developed by Yang (2010), and is very similar to PSO. The algorithm emulates the brightness of firefly flashes, with the idea that brighter flashes attract other fireflies, which eventually converge towards an optimum. It has been adapted for MOO by Apostolopoulos & Vlachos (2011). They studied the economic emission load dispatch problem in power plants in which the decision task is to allocate loads to the power generators such that power demands are met, while total fuel cost and emissions are minimised. The algorithm seemed to perform well in this constrained problem.

This concludes the discussion of some of the popular MOO algorithms. In the next section, hyperheuristics will be briefly explained.

2.5 Hyperheuristics

A hyperheuristic attempts to combine the strengths of several (meta)heuristics to solve a problem, thus allowing the building of optimisers which can solve classes of problems. This requires them to be generic. Whereas the metaheuristic searches a

solution space, the hyperheuristic searches a set of (meta)heuristics (the "search space") to find the appropriate solution method (or sequence of methods) for a given problem. The formal definition by Gendreau & Potvin (2010:452) is useful in this context: *A hyper-heuristic is an automated methodology for selecting or generating heuristics to solve hard computational search problems.* Hyperheuristics often rely on machine learning techniques to select and combine heuristics.

An example of a hyperheuristic is the AMALGAM algorithm (Vrugt & Robinson, 2007), which uses four algorithms in its framework, namely NSGA-II, Adaptive Metropolis, PSO and DE. A population of N solutions is maintained, and the offspring are created proportionally to the success of the contributing algorithms during the optimisation process.

A practical application of the AMALGAM algorithm has been demonstrated in the field of Water Distribution System Design Optimisation (WDSDO) by Raad *et al.* (2011). Miranda *et al.* (2010) demonstrated an application in MOO using the 2D guillotine strip-packing problem. They optimised the usage of the raw material as well as the production process by minimising the cutting length as well as the number of cuts executed by the cutting machines. The latter is important, as it affects the lifespan of the equipment and the cost of the production process.

2.6 General applications of MOO

Multi-objective optimisation problems and their solutions are widely reported in the literature. Baesler & Sepúlveda (2001) improved the design of a cancer treatment centre based on four objectives, while Li *et al.* (2009) used an MOO method in an environmentally conscious design of chemical processes and products. Kleijnen & Wan (2007) studied optimisation of simulated systems by comparing some optimisation methods. These include a brute-force approach, modified *Response Surface Methodology* (RSM), *Perturbation Analysis* (PA) and *Feasible Directions* (FD).

In an aerodynamic application, Szőllős *et al.* (2009) optimised the airfoil design of a standard-class glider. NC-tool paths can be determined via simulation, and Kersting & Zabel (2009) developed an optimisation approach which is included in a tool path simulation model for a five-axis milling process. A hybrid MOEA works on the tilting and rotational angles of the milling tool to obtain collision free tool paths.

Coello Coello *et al.* (2007) also list a large number of MOO application areas in Chapter 7 of their book, among others the following:

- Environmental, naval and hydraulic engineering
- Electrical and electronics engineering
- Telecommunications and network optimisation
- Robotics and control engineering
- Structural and mechanical engineering
- Civil and construction engineering
- Transport engineering
- Aeronautical engineering
- Geography
- Chemistry
- Physics
- Medicine
- Ecology
- Computer science and computer engineering
- Design and manufacture
- Scheduling
- Management
- Grouping and packing
- Finance
- Classification and prediction.

Zhou *et al.* (2011) list similar application areas, as well as other areas like data mining, bio-informatics, artificial neural networks and fuzzy systems.

In the next sections, applications specific to Industrial Engineering and Process Engineering are briefly listed.

2.7 MOO applications in Industrial Engineering

There are many examples of MOO applications in the domain of Industrial Engineering. A few are discussed here, followed by a summary of publications that combine both applications and the domain.

Tight design tolerances result in good components and good fits at assembly, but cost money. Sivakumar *et al.* (2011) apply the NSGA-II and MOPSO to find good tolerances and manufacturing processes for an overrunning clutch assembly and a three-arm knuckle joint assembly. They consider three objectives (minimum tolerance stack-up, minimum manufacturing cost and minimum quality loss function), three constraints and five decision variables.

Chica *et al.* (2011) propose that MOGAs are ineffective in MOO of time and space assembly line balancing, and develop an algorithm, called advanced *Time and Space Assembly Line Balancing NSGA-II* (advanced TSALBP-NSGA-II), which takes into account the characteristics specific to this family of problems. Objectives are typically the cycle time, the number of stations, and/or the area of these stations. They studied nine standard problems as well as a tenth problem corresponding to the assembly process of the Nissan Pathfinder engine, which is assembled at the Nissan industrial plant in Barcelona, Spain. Their conclusion is that MOO algorithms such as advanced TSALBP-NSGA-II can perform well in the context of difficult problems if the algorithm parameters are well chosen.

Workers, especially those who do physical work, may suffer from *Musculoskeletal Disorders* (MSDs). Manual operations are optimised in the design stage to avoid or decrease the risk of MSD via human modelling techniques from ergonomics and occupational biomechanics. Ma *et al.* (2009) propose a new posture prediction and analysis method for predicting the optimal posture under both non-fatigue and fatigue conditions. The objectives are the minimisation of fatigue (stress) and discomfort, and the constraints (kinematical and biomechanical) define the possible design space. A posture can be evaluated and designed for manual handling operations. They apply the method to a drilling operation of an aircraft fuselage. In this case, an operator holding a drill-pipe combination with a mass of 7 kg has to drill up to 2 000 holes for rivets, each hole requiring 49 N of force.

A human lifting simulation model is studied by Xiang *et al.* (2010). They attempt to predict the dynamic lifting motion and determine the contributions of each performance measure using a 55-degree-of-freedom digital human model. The

performance measures are the dynamic effort and stability, which were combined in a weighted objective function. MOO was used to find the best weights and to determine what governs human behaviour during lifting tasks.

MOO has been widely applied to inventory management (Mahapatra & Maiti, 2005; Roy & Maiti, 1998; Tsou, 2008, 2009), while MOO applications in economy and finance are described in Mishra *et al.* (2011).

The author did a survey of journal articles including the term "multi-objective optimization" in the journal *Computers & Industrial Engineering*, which is arguably the scholarly journal containing the majority of articles on computer applications associated with Industrial Engineering. Fifty two recent publications had been identified by 31 July 2012. The publications were categorised according to topic; when more than one topic appeared in the title, for example "Scheduling of flow shops", the first topic ("Scheduling") was taken for classification, instead of "flow shops". Only the volumes and issue numbers are shown in the summary in Table 2.2. The application areas are diverse with a strong focus on scheduling problems.

Next, the focus turns to MOO applications in Process Engineering.

2.8 MOO applications in Process Engineering

Multi-objective optimisation has been applied in a diverse range of problems in Process Engineering. Bashkar *et al.* (2001) apply the NSGA of Srinivas & Deb (1995) to a polymerisation reactor system in which the objectives are to minimise the (undesired) acid and vinyl end group concentrations, while requiring to produce a polymer with a desired *degree of polymerisation* (DP). The latter is a constraint, and it is further required that the concentration of the di-ethylene end group in the product be within a certain range. Several decision variables were identified, including the rotation speed of the agitator, reactor pressure, temperature, catalyst concentration and the time the reaction mass is allowed to reside in the reactor. The settings of these variables influence the quality of the final product (e.g. stiffness, strength) in often conflicting ways. They found that the NSGA fails to give correct solutions (measured against a pre-validated model), unless the algorithm is applied several times with fewer decision variables.

In Anderson *et al.* (2005), a generic waste incineration plant is studied using the MOGA of Fonseca & Fleming (1998). The objectives are the maximisation of throughput and the minimisation of environmental damaging products like

Topic	Volume and Number
Line balancing	60(3), 62(1)
Flexible manufacturing systems	23(1–4)
Flow shops	30(4)
Curriculum development	31(1–2)
Non-linear goal programming	31(3–4)
Product design	33(1–2), 60(4)
Classification	35(3–4)
Product planning in QFD	44(1)
Antenna placement	44(2)
Supply chain design	50(1–2), 52(1), 55(3), 56(4), 58(4)
Logistics	50(3)
Spatial design	54(4)
Travelling salesman problem	55(2), 56(3), 59(2)
Product mix	56(3), 61(3)
MOEA refinement	56(4)
Aggregate production planning	56(4)
Project management	57(4)
Scheduling	37(1–2), 48(2), 51(3), 54(3–4) 55(2), 56(4), 59(4), 61(3), 63(1)
Quality	60(1)
Facility layout design	62(4)
Production planning	62(2)
Machining	62(2)
Passenger screening	62(4)
Reliability	62(1), 63(1)

Table 2.2: Publications pertaining to MOO in *Computers & Industrial Engineering*.

NO_x and dioxins while keeping the operational constraints (temperature, oxygen concentration) within limits. Although several decision variables can be identified in this plant type, they considered the waste feed rate and the residence time as decision variables. The plant was modelled using a radial basis function network that yielded the performance of the plant under different operational settings. It was concluded that the use of the MOGA allowed for a robust plant-wide optimisation procedure, and that the inclusion (or exclusion) of constraints and objectives results in different solution regions.

In a study by Tarafder *et al.* (2005) of the design and operation of an industrial styrene monomer manufacturing process, the NSGA-II (Deb *et al.*, 2002) is applied. Two objectives, namely styrene flow rate and styrene selectivity, are maximised

in single-bed, steam-injected and double-bed reactors. In the second part of their study, a double-bed reactor is considered while maximising the styrene flow rate and selectivity and minimising the total heat duty of the manufacturing process. Up to 10 decision variables are identified, depending on the reactor type. These include pressure, steam to reactant (ethylbenzene) molar ratio, reactor length to diameter ratio, temperature, feed rates and superheated steam fraction. Pareto fronts were obtained that may be explained qualitatively, while useful counter-intuitive results are also reported in terms of the reactor volume and the heat duty required by the reactors.

For more MOO applications in Process Engineering, also see Tarafder *et al.* (2007) (finding the best of the Pareto set), Gao & Engell (2005) (set-point optimisation of batch chromatography), and Montazer-Rahmati & Binaee (2010) (hydrogen plant optimisation).

2.9 Robust MOO

In MOO, the problem (including its decision variables, objectives and constraints) is usually considered given one or more algorithms may be used to solve the problem. The problem may involve test functions or be practical for which an existing algorithm is adapted or a new algorithm is developed. These problems are usually deterministic, but noisy optimisation is also performed, as will be described in specific parts of this dissertation. An emerging research field is *Robust Multi-objective Optimisation* (RMOO), which is different from noisy optimisation. This field is briefly considered in this section.

In noisy optimisation (Goh & Tan, 2007), uncertainty or variation is inherent to the objective functions, mainly because they are estimated and contain a degree of statistical error. In RMOO problems, optimisation is performed while the problem experiences perturbations. For example, a measuring process has inherent variation, while the demand at each visiting point in a vehicle routing problem may vary (Goh *et al.*, 2010a). These authors identify four types of uncertainties which affect the optimisation process: 1) noisy fitness functions, 2) uncertainty of design variables or the environment, 3) approximation errors and 4) time-varying or dynamic fitness functions.

Goh *et al.* (2010a) develop test functions for case 2 above (uncertainty of design variables or the environment) and Gupta & Deb (2005) propose constraint-handling

strategies for RMOO. Techniques to address RMOO include a Polynomila Chaos expansion by Poles & Lovison (2009), which speeds up the optimisation process compared to a Monte-Carlo approach. Cromvik *et al.* (2011) hypothesised that each decision maker has a single hidden objective in mind, which they characterise by a family of utility functions. They also propose a computational procedure for estimating robustness.

Voss *et al.* (2011) propose new noise-handling strategies by measures of uncertainty to estimate the Pareto dominance. Cinnella & Hercus (2010) study airfoil profiles for transonic inviscid flows of dense gases with uncertainties induced by upstream thermodynamic conditions. Such conditions exist in for example industrial processes where energy is recovered from waste heat. Turbine design (airfoil geometries) can be improved by their combination of a MOGA and the probabilistic collocation method. Power dispatching is subjected to uncertainty in demand. In this regard, Zhihuan *et al.* (2010) used *Multi-objective Optimal Reactive Power Dispatch* (MORPD) to address uncertain load perturbations during system operations by adapting the NSGA-II in order to find robust Pareto solutions. These robust solutions are stable in the presence of load perturbations.

2.10 Summary: Chapter 2

An overview of the scholarly books, chapters in books, journal articles, reports and conference proceedings in the discipline of multi-objective optimisation and related topics was presented in this chapter. The objective was not to provide a comprehensive or complete survey, but to give the reader an idea of the various approaches typically used in MOO, some MOO applications and new trends in MOO. There was a strong focus on evolutionary algorithms, as these may be considered as "the point where it all started", but other metaheuristics were also discussed, including others based on biological and non-biological principles. General applications of these and relevant sources were presented, as well as applications in the narrower context of Industrial Engineering and Process Engineering, respectively.

It seems as if the major trend in MOO during the last decade has been to 1) think of a novel approach, 2) formulate such an approach, 3) code it, 4) evaluate it with respect to test problems, 5) compare it to results obtained by (an)other algorithm(s) and conclude that it outperforms the(se) algorithm(s). A similar observation was made by Huband *et al.* (2006) in terms of testing: testing was

apparently driven by the desire to develop algorithms that could do optimisation with as few objective evaluations as possible. This is also an observation made by Coello Coello (2009), which he calls "Efficiency". Many algorithms are evaluated against the NSGA-II of Deb *et al.* (2002), and it would seem that there is still no better benchmark algorithm available after nearly 10 years.

A certain suite of test functions for algorithm evaluation is widely used, but the functions proposed by Huband *et al.* (2006), Igel *et al.* (2007) and Li & Zhang (2009) may also be considered when evaluating an algorithm. All these test problems are deterministic, while standard stochastic problems are needed for testing to cover all the requirements listed above (Goh *et al.*, 2010a). The quality indicators typically employed are also limited to a standard few, and perhaps a more comprehensive standard suite should be developed.

The research field of MOO is active over a diverse front of problem solutions and will present academics and practising scientists and engineers many more opportunities for research in the next decade.

In the next chapter the cross-entropy method for optimisation, on which this research is based, will be presented.

CHAPTER 3

THE CROSS-ENTROPY METHOD FOR OPTIMISATION

In the previous chapter a scholarly overview of topics related to MOO was given. The research in this dissertation is based on the *cross-entropy method* (CEM) which is presented in this chapter, specifically from an optimisation viewpoint. Its theoretical origin is discussed first, followed by sections on the theory of its application in continuous optimisation and discrete optimisation.

The main algorithm for optimisation via the CEM is illustrated using four continuous, single-objective problems. The theoretical discussions are based on the book by Rubinstein & Kroese (2004). A practical application of the CEM to airport apron layout design is presented as a single-objective optimisation problem of a dynamic, stochastic problem.

3.1 The CEM for optimisation

The essence of the theory supporting the CEM for optimisation is briefly outlined in this section. For more detail the reader is referred to Rubinstein & Kroese (2004), the CEM website (http://www.cemethod.org, cited on 10 August 2012) and Kroese & Rubinstein (2005), the latter being a complete journal issue devoted to the CEM. The CEM for optimisation has its foundation in *Importance Sampling* and the *Kullback-Leibler distance*. These aspects are discussed first (Rubinstein & Kroese, 2004).

Let $\mathbf{X} = (X_1, \ldots, X_n)$ be a random vector assuming values from some space $\mathcal{X}$, and let f be some real function on $\mathcal{X}$. Suppose one wants to determine the probability that $f(\mathbf{X})$ is greater than or equal to a real number γ under a family

of probability density functions $h(\cdot;\mathbf{u})$ on $\mathcal{X}$, with $\mathbf{u}$ the parameter vector. This probability is

$$l = \mathbb{P}_{\mathbf{u}}(f(\mathbf{X}) \geq \gamma) = \mathbb{E}_{\mathbf{u}} I_{\{f(\mathbf{X})\geq\gamma\}}. \tag{3.1}$$

The $f(\mathbf{X}) \geq \gamma$ is called a *rare event* if l is very small, and it can be efficiently estimated using importance sampling. To do so, a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ is taken from a different density g on $\mathcal{X}$, and l is estimated using the likelihood ratio estimator (Rubinstein & Kroese, 2004)

$$\hat{l} = \frac{1}{N}\sum_{i=1}^{N} I_{\{f(\mathbf{X}_i)\geq\gamma\}} \frac{h(\mathbf{X}_i;\mathbf{u})}{g(\mathbf{X}_i)}. \tag{3.2}$$

Now use of the change of measure with density

$$g^*(\mathbf{x}) = \frac{I_{\{f(\mathbf{x})\geq\gamma\}} h(\mathbf{x};\mathbf{u})}{l} \tag{3.3}$$

yields the probability

$$l = \frac{I_{\{f(\mathbf{X}_i)\geq\gamma\}} h(\mathbf{X}_i;\mathbf{u})}{g^*(\mathbf{X}_i)}. \tag{3.4}$$

The value of g^* depends on the unknown l, but g^* can be approximated within the family of densities $\{h(\cdot;\mathbf{v})\}$ with reference parameter $\mathbf{v}$ such that the distance between g^* and $h(\cdot;\mathbf{v})$ is minimal. A measure of this distance is the *Kullback-Leibler distance* or cross-entropy (CE)

$$\begin{aligned} \mathcal{D}(g,h) &= \mathbb{E}_g \ln \frac{g(\mathbf{X})}{h(\mathbf{X})} && (3.5)\\ &= \int g(\mathbf{x}) \ln g(\mathbf{x}) \mathrm{d}\mathbf{x} - \int g(\mathbf{x}) \ln h(\mathbf{x}) \mathrm{d}\mathbf{x}, && (3.6) \end{aligned}$$

between g and h. To minimise the Kullback-Leibler distance between g and g^* in (3.3), $h(\cdot;\mathbf{v})$, $\mathbf{v}$ is chosen such that $-\int g^*(\mathbf{x}) \ln h(\mathbf{x};\mathbf{v}) \mathrm{d}\mathbf{x}$ is minimised. This can be achieved by solving the maximisation problem

$$\max_{\mathbf{v}} \int g^*(\mathbf{x}) \ln h(\mathbf{x};\mathbf{v}) \mathrm{d}\mathbf{x}. \tag{3.7}$$

When g^* in (3.3) is substituted into (3.7), the maximisation program

$$\max_{\mathbf{v}} \int \frac{I_{\{f(\mathbf{x})\geq\gamma\}} h(\mathbf{x};\mathbf{u})}{l} \ln h(\mathbf{x};\mathbf{v}) \mathrm{d}\mathbf{x} \tag{3.8}$$

is obtained, which is equivalent to the program

$$\max_{\mathbf{v}} D(\mathbf{v}) = \max_{\mathbf{v}} \mathbb{E}_{\mathbf{u}} I_{\{f(\mathbf{X})\geq\gamma\}} \ln h(\mathbf{X};\mathbf{v}). \tag{3.9}$$

The result is similar for the discrete case. With reference to the above, the CEM for optimisation can now be stated, with the continuous and discrete cases presented separately.

3.1.1 The CEM for continuous optimisation

Suppose one wishes to find the maximum of some performance function $f(\mathbf{x})$ over all states $\mathbf{x}$ in some set $\mathcal{X}$. Let the maximum be γ^*, then

$$\gamma^* = \max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}). \tag{3.10}$$

The deterministic problem is randomised by defining a family of *probability density functions* (pdfs) $\{h(\cdot;\mathbf{v}), \mathbf{v}\in\mathcal{V}_p\}$ on the set $\mathcal{X}$. The *stochastic problem associated* with (3.10) is the estimation problem

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

The $\mathbf{X}$ is a random vector with probability density function $h(\cdot;\mathbf{u})$ for some $\mathbf{u}\in\mathcal{V}_p$. When estimating l, $\{f(\mathbf{X})\geq\gamma\}$ can be considered a rare event, and l can be estimated by making adaptive changes to the probability density function using the Kullback-Leibler cross-entropy. A sequence of probability density functions $h(\cdot;\mathbf{u}), h(\cdot;\mathbf{v}_1), h(\cdot;\mathbf{v}_2), \ldots$ is thus created which is steered in the direction of the theoretical optimal density. One generates a sequence of tuples $\{(\hat{\gamma}_t, \hat{\mathbf{v}}_t)\}$ that converges to the optimal tuple $(\gamma^*, \mathbf{v}^*)$, and setting $\mathbf{v}_0 = \mathbf{u}$, the procedure is as follows (Rubinstein & Kroese, 2004):

1. **Adaptive updating of γ_t.** For a fixed $\mathbf{v}_{t-1}$, let γ_t be the $(1-\varrho)$-quantile of $f(\mathbf{X})$ under $\mathbf{v}_{t-1}$. That is, $\mathbb{P}_{\mathbf{v}_{t-1}}(f(\mathbf{X})\geq\gamma_t)\geq\varrho$, with ϱ typically chosen as $\varrho = 10^{-2}$. Now estimate γ_t by drawing a random sample $\mathbf{X}_1,\ldots,\mathbf{X}_N$ from $h(\cdot;\mathbf{v}_{t-1})$ and determine the sample $(1-\varrho)$-quantile of the performances

$$\hat{\gamma}_t = f_{(\lceil(1-\varrho)N\rceil)}. \tag{3.12}$$

2. **Adaptive updating of $\mathbf{v}_t$.** For γ_t and $\mathbf{v}_{t-1}$ derive $\mathbf{v}_t$ by solving the program

$$\max_{\mathbf{v}} D(\mathbf{v}) = \max_{\mathbf{v}} \mathbb{E}_{\mathbf{v}_{t-1}} I_{\{f(\mathbf{X})\geq\gamma_t\}} \ln h(\mathbf{X};\mathbf{v}). \tag{3.13}$$

The value of $\max_{\mathbf{v}} D(\mathbf{v})$ in (3.13) can be estimated by means of the stochastic program

$$\max_{\mathbf{v}} \hat{D}(\mathbf{v}) = \max_{\mathbf{v}} \frac{1}{N} \sum_{i=1}^{N} I_{\{f(\mathbf{X}_i) \geq \hat{\gamma}_t\}} \ln h(\mathbf{X}_i; \mathbf{v}). \tag{3.14}$$

The parameter vector $\hat{\mathbf{v}}$ can be updated using a smoothing function

$$\hat{\mathbf{v}}_t = \alpha \tilde{\mathbf{v}}_t + (1 - \alpha)\hat{\mathbf{v}}_{t-1}, \tag{3.15}$$

where $\tilde{\mathbf{v}}_t$ is obtained from (3.14) and α is a smoothing constant typically in the range 0.6–0.9. Based on the above, the main CE Algorithm for Optimisation by Rubinstein & Kroese (2004) is shown as Algorithm 1.

Algorithm 1 Main CE Algorithm: continuous optimisation

1: Choose some $\hat{\mathbf{v}}_0$ for the density $h(\cdot; \mathbf{v})$. Set $t = 1$.
2: Generate a sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from the density $h(\cdot; \hat{\mathbf{v}}_{t-1})$ and compute the sample $(1 - \varrho)$-quantile $\hat{\gamma}_t$ of the performances according to (3.12).
3: Use the same sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ and solve the stochastic program in (3.14). This solution is $\tilde{\mathbf{v}}_t$.
4: Smooth the vector $\tilde{\mathbf{v}}_t$ using the expression in (3.15).
5: If, for some $t \geq \delta$, say $\delta = 5$, $\hat{\gamma}_t = \hat{\gamma}_{t-1} = \ldots = \hat{\gamma}_{t-\delta}$, then stop; otherwise set $t \leftarrow t + 1$ and return to Step 2.

Rubinstein & Kroese (2004) proved that the CE optimal density is often the atomic density at $\mathbf{x}^*$.

The discrete case of the CEM for optimisation is discussed next.

3.1.2 The CEM for discrete optimisation

The discrete optimisation case of the CEM is analogous to the continuous case and is briefly described here. Suppose the maximum of Υ over $\mathcal{X}$ is γ^*, then similar to (3.10),

$$\Upsilon(\mathbf{x}^*) = \gamma^* = \max_{\mathbf{x} \in \mathcal{X}} \Upsilon(\mathbf{x}). \tag{3.16}$$

The CEM requires that an estimation problem be associated with the optimisation problem of (3.16). To do so, one defines a collection of indicator functions $\{I_{\{\Upsilon(\mathbf{x}) \geq \gamma\}}\}$ on $\mathcal{X}$ for different values of the threshold $\gamma \in \mathbb{R}$. Let $\{f(\cdot, \mathbf{v}), \mathbf{v} \in \mathcal{V}_p\}$ be a family of *discrete probability mass functions* (pmfs) on $\mathcal{X}$ that are parameterised by a real-valued vector $\mathbf{v}$.

To solve the problem associated with (3.16), assume $\mathbf{u} \in \mathcal{V}_p$ and estimate the probability

$$l = \mathbb{P}_{\mathbf{u}}\{\Upsilon(\mathbf{X}) \geq \gamma\} = \sum_{\mathbf{x}} I_{\{\Upsilon(\mathbf{x})\geq\gamma\}} f(\mathbf{x};\mathbf{u}) = E_{\mathbf{u}} I_{\{\Upsilon(\mathbf{X})\geq\gamma\}} \tag{3.17}$$

with $f(\mathbf{x};\mathbf{u})$ being the pmf on $\mathcal{X}$ and γ some chosen level. Suppose now γ is equal to γ^*, then $l = f(\mathbf{x}^*;\mathbf{u})$, which is a very small probability. It can be estimated using importance sampling by taking a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from a different pmf g and estimating l via

$$\hat{l} = \frac{1}{N}\sum_{k=1}^{N} I_{\{\Upsilon(\mathbf{X}_k)\geq\gamma\}} \frac{f(\mathbf{X}_k,\mathbf{u})}{g(\mathbf{X}_k)} \tag{3.18}$$

which is the unbiased *importance sampling estimator* of l. The optimal way to estimate l is to use the change of measure with a different pmf

$$g^*(\mathbf{x}) := \frac{I_{\{\Upsilon(\mathbf{x})\geq\gamma\}} f(\mathbf{x};\mathbf{u})}{l}. \tag{3.19}$$

Since this optimal probability mass function is generally difficult to obtain and depends on the unknown l, one chooses g such that the cross-entropy or Kullback-Leibler distance between g and g^* is minimal. The Kullback-Leibler distance between two probability mass functions g and h is defined as

$$\begin{aligned} \mathcal{D}(g,h) &= \mathbb{E}_g\left[\log\frac{g(\mathbf{X})}{h(\mathbf{X})}\right] \\ &= \sum_{\mathbf{x}} g(\mathbf{x}) \log\frac{g(\mathbf{x})}{h(\mathbf{x})} \\ &= \sum_{\mathbf{x}} g(\mathbf{x}) \log g(\mathbf{x}) - \sum_{\mathbf{x}} g(\mathbf{x}) \log h(\mathbf{x}). \end{aligned} \tag{3.20}$$

Since $I_{\{\Upsilon(\mathbf{x})\geq\gamma\}}$ is non-negative, and the probability mass function f is parameterised by a finite dimensional vector $\mathbf{v}$, $f(\mathbf{x}) = f(\mathbf{x};\mathbf{v})$, $g(\mathbf{x}) = f(\mathbf{x};\mathbf{v})$ for some reference parameter $\mathbf{v}$ (Kroese, 2010). To estimate l (in (3.18)), one chooses $\mathbf{v}$ such that $\mathcal{D}(g^*, f(\cdot;\tilde{\mathbf{v}}))$ is minimal. That means $\mathbb{E}_{\mathbf{v}} I_{\{\Upsilon(\mathbf{X})\geq\gamma\}} \log f(\mathbf{X};\tilde{\mathbf{v}})$ should be maximal.

Alon *et al.* (2005) showed that for discrete random vectors $\mathbf{X}$ the components of $\tilde{\mathbf{v}}$ will always be of the form

$$\frac{\mathbb{E}_{\mathbf{v}} I_{\{S\Upsilon(\mathbf{X})\geq\gamma\}} I_{\{\mathbf{X}\in A\}}}{\mathbb{E}_{\mathbf{v}} I_{\{\Upsilon(\mathbf{X})\geq\gamma\}} I_{\{\mathbf{X}\in B\}}}, \tag{3.21}$$

where $A \subset B \subset \mathcal{X}$. This number can be estimated by taking a random sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from the pmf $f(\cdot, \mathbf{v})$ and evaluating

$$\frac{\sum_{k=1}^{N} I_{\{\Upsilon(\mathbf{X}_k) \geq \gamma\}} I_{\{\mathbf{X}_k \in A\}}}{\sum_{k=1}^{N} I_{\{\Upsilon(\mathbf{X}_k) \geq \gamma\}} I_{\{\mathbf{X}_k \in B\}}}. \tag{3.22}$$

One can use distributions p_j in the discrete optimisation problem to draw observations for random vectors $\mathbf{X}_i = (X_{i1}, \ldots, X_{in_d})$, for $j = 1, \ldots, n_d$ elements in the decision vector. The estimator for p_j is (Alon *et al.*, 2005)

$$\hat{p}_j = \frac{\sum_{i=1}^{N} I_{\{\hat{\Upsilon}(\mathbf{X}_i) \geq \gamma\}} I_{\{X_{ij}=j\}}}{\sum_{i=1}^{N} I_{\{\hat{\Upsilon}(\mathbf{X}_i) \geq \gamma\}}}, \tag{3.23}$$

which has the same form as the expression in (3.22). The elements $\hat{p}_j$ in (3.23) together form the probability vector $\hat{P}_t$, and $P_0 = 0.5$.

The smoothing update rule is similar to that in (3.15), and is

$$\hat{P}_t = \alpha \tilde{P}_t + (1 - \alpha)\hat{P}_{t-1}. \tag{3.24}$$

The probabilities in P will approach zero or one after a sufficient number of iterations. The optimisation algorithm for the discrete case is shown in Algorithm 2 for N row vectors and n_d elements in the decision vector $\mathbf{X}$.

Algorithm 2 Main CE Algorithm: discrete optimisation

1: Assign the elements of $\hat{P}_0$ the value 0.5. Set $t = 1$.
2: Generate a sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ using P_{t-1}, and compute the sample quantile $(1 - \varrho)$-quantile $\hat{\gamma}_t$ of the performance function according to (3.12).
3: Using the same sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$, update $\hat{p}_j$ with the expression in (3.23).
4: Smooth $\hat{P}_t$ with (3.24).
5: If, for some $t \geq \delta$, say $\delta = 5$, $\hat{\gamma}_t = \hat{\gamma}_{t-1} = \ldots = \hat{\gamma}_{t-\delta}$, then stop; otherwise set $t \leftarrow t + 1$ and return to Step 2.

This concludes the theoretical overview of the CEM and its formulation in an optimisation context. Next, it is demonstrated using concrete examples.

3.2 The CEM and single-objective optimisation

In this section, the CEM for optimisation is illustrated using four continuous, single-objective functions defined on finite spaces. These are De Jong's first function, the Rosenbrock function, the Shekel function and the Rastrigin function. Algorithm 1 was implemented in Matlab® 2007b, and the parameters for all four cases were: $N = 100$, $\alpha = 0.8$, $\varrho = 0.2$ and $\gamma = 10^{-6}$.

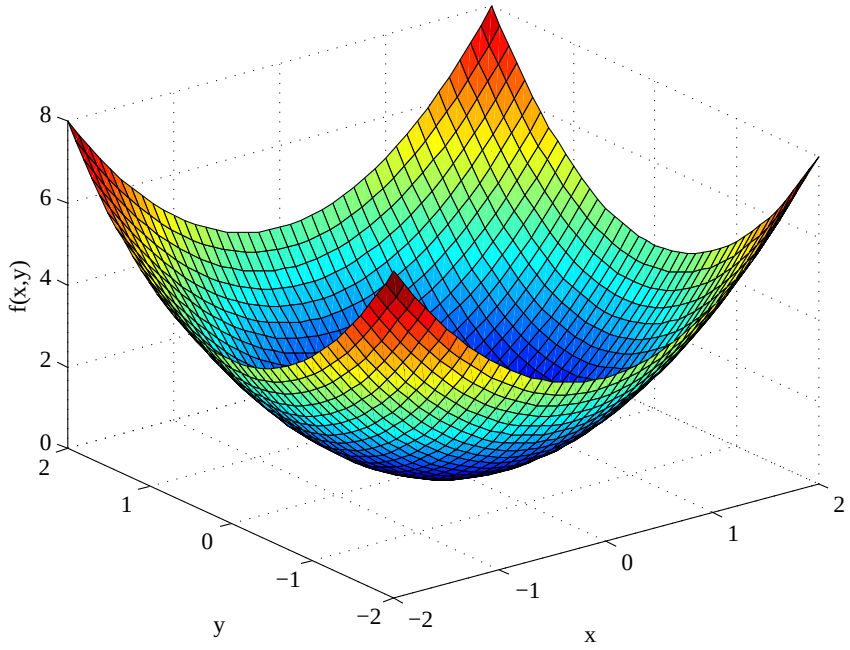

Figure 3.1: De Jong's first function.

3.2.1 De Jong's first function

The function shown in Figure 3.1 is defined by $f(x,y) = x^2 + y^2$, $-2 \leq x, y \leq 2$, and is known as De Jong's first function. It has its absolute minimum at $(0,0)$. The initial parameter vector $\hat{\mathbf{v}}_0$ is assigned random values on the range $(-2,2)$. Sampling was performed on this definition range using truncated normal distributions with the parameter vector $\mathbf{v} = (\mu_x, \sigma_x, \mu_y, \sigma_y)$, and estimated with $\hat{\mathbf{v}}_\mathbf{t} = (\overline{x}_t, \hat{\sigma}_x, \overline{y}_t, \hat{\sigma}_y)$. The algorithm found the minimum 1.0088×10^{-12} at $(-0.2515 \times 10^{-6}, -0.3080 \times 10^{-6})$ after 26 iterations. De Jong's first function is very simple. Three more intricate functions with deceptive local optima are presented next.

3.2.2 The Rosenbrock function

The Rosenbrock function is an accepted benchmark for optimisation algorithms. It must be minimised and one variant is defined as

$$f(\mathbf{x}) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{3.25}$$

with D variables $x_1, \ldots, x_D$, where $-2 \leq x_i \leq 2$ for all $i = 1, \ldots, D$. A plot for $D = 2$ is shown in Figure 3.2. The exact minimum is a vector of ones, that is $\mathbf{x}^* = (1, 1, 1, \ldots, 1)$, with $f(\mathbf{x}^*) = 0$, while a local minimum exists at $(-1, 1, 1, \ldots, 1)$

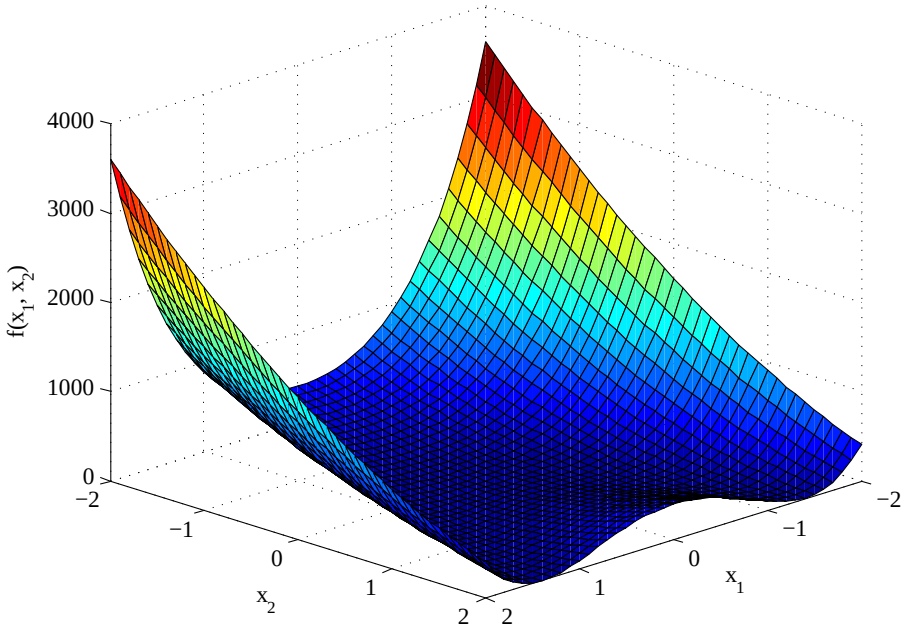

Figure 3.2: The Rosenbrock function with $D = 2$ and $-2 \leq x_i \leq 2$.

for $4 \leq D \leq 7$. The case of $D = 6$ required 4 000 iterations and yielded the vector $\hat{\mathbf{x}}^* = (0.9999, 0.9998, 0.9997, 0.9995, 0.9990, 0.9979)$, with $f(\hat{\mathbf{x}}^*) = 1.378 \times 10^{-5}$. The development of μ_i for this case $(D = 6)$ is shown in Figure 3.3.

The implementation of the algorithm in Matlab® 2007b is based on the code listing in Appendix A.5 in Rubinstein & Kroese (2004). They used a penalisation value if the algorithm samples solutions out of bounds, but in this study the sampling was performed using truncated normal distributions on the variable definition ranges.

Note in Figure 3.3 how all parameters converge to their limiting values.

3.2.3 The Shekel function

The Shekel function

$$f(\mathbf{x}) = -\sum_{i=1}^{m_s} \frac{1}{c_i + \sum_{j=1}^{D} (x_j - a_{ij})^2} \tag{3.26}$$

is another widely accepted benchmark for evaluating optimisation algorithms. It has to be minimised and depends on D variables $x_1, \ldots, x_D$ with $0 \leq x_i \leq 10$, m_s being the number of local minima and can take, by definition, values of 5, 7 or 10.

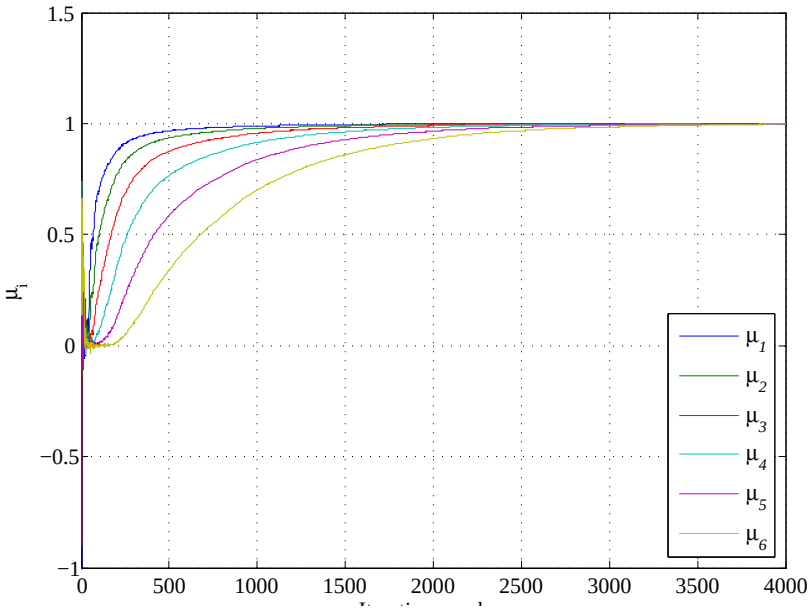

Figure 3.3: Rosenbrock function optimisation: progress of the μ_i.

If $D = 4$ and $m_s = 10$, then

$$a_{ij} = \begin{bmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 5.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 5.0 & 1.0 & 2.0 & 3.6 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 & 2.0 & 3.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 & 9.0 & 3.0 & 1.0 & 2.0 & 3.6 \end{bmatrix}^{\mathrm{T}}$$

and

$$c_i = \begin{bmatrix} 0.1 & 0.2 & 0.2 & 0.4 & 0.4 & 0.6 & 0.3 & 0.7 & 0.5 & 0.5 \end{bmatrix}^{\mathrm{T}}.$$

The values of a_{ij} and c_i are determined via an algorithm for each problem instance. A plot of $-f(\mathbf{x})$ for the case $m_s = 10$ and $D = 2$ is shown in Figure 3.4 (the negative of the function is plotted for clarity). The absolute minimum of $f(\mathbf{x})$ is at $(4, 4)$ with $f(4, 4) = -11.0298$.

The CEM was applied to the case $D = 4$ and $m_s = 10$. The progress of the $\mathbf{v}$-vector of means and standard deviations is shown in Figure 3.5. The values of the σ_i are hard to distinguish because they are very close, but they all approach zero as required by the CEM.

The final solution was reached after 36 iterations with the estimated optimal vector $\hat{\mathbf{x}}^* = (4.0007, 4.0006, 3.9997, 3.9995)$ and $f(\hat{\mathbf{x}}^*) = -10.5364$.

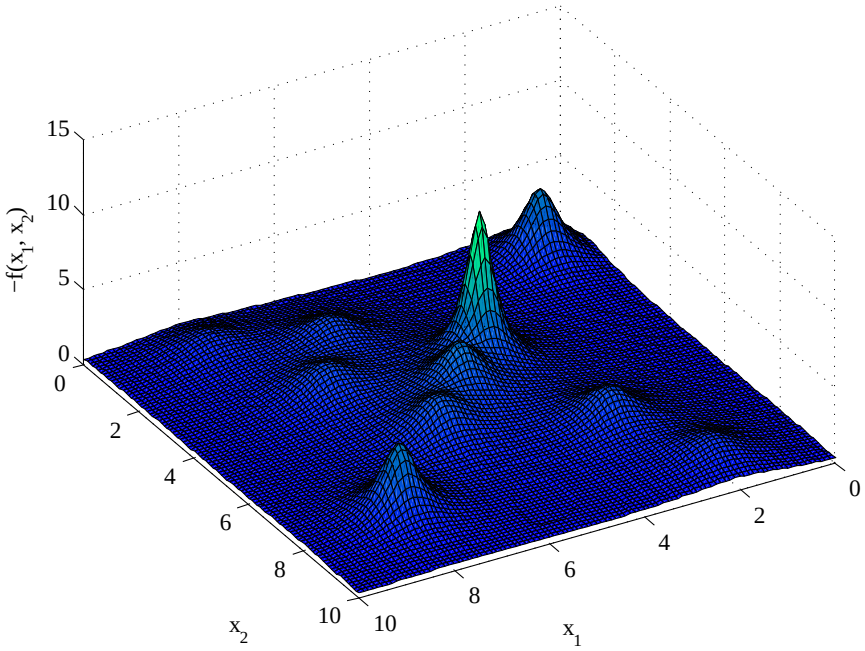

Figure 3.4: Negative Shekel function with 10 peaks.

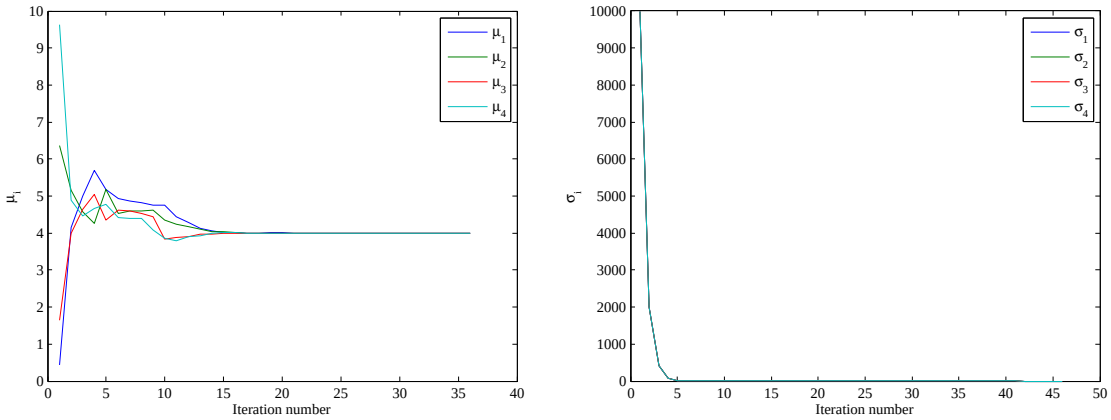

Figure 3.5: Shekel function optimisation: progress of the **v** vector.

3.2.4 The Rastrigin function

The Rastrigin function is the last function for which the single-objective CEM is demonstrated, and is chosen because it has many more local optima than the Shekel function. It is the function

$$f(\mathbf{x}) = 10D + \sum_{i=1}^{D}[x_i^2 - 10(\cos(2\pi x_i))], \tag{3.27}$$

which has to be minimised. A plot for the case of $D = 2$ decision variables, where $-5.12 \leq x_i \leq 5.12$ for $i = 1, 2$, is shown in Figure 3.6. The absolute minimum is at $(0, 0)$ with $f(0, 0) = 0$.

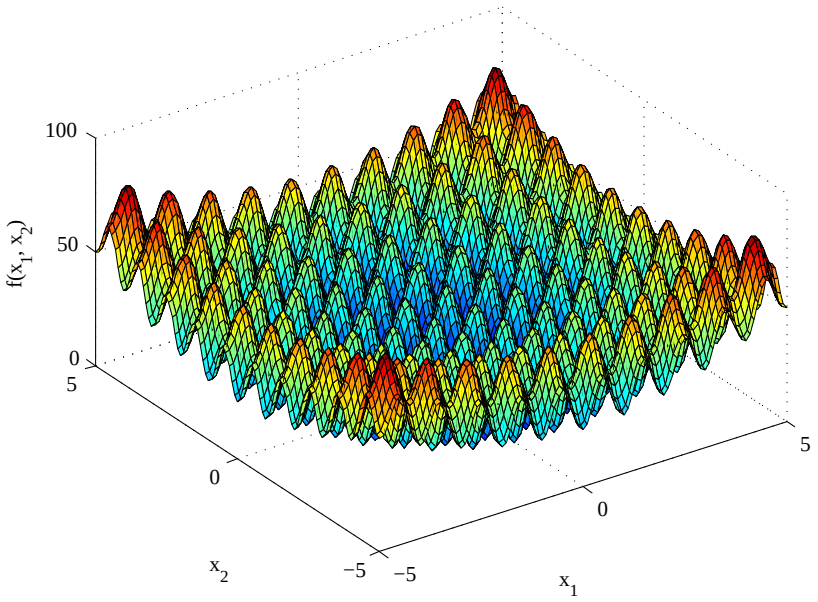

Figure 3.6: Rastrigin function with $D = 2$.

The CEM was applied to the case $D = 6$. The progress of the $\mathbf{v}$-vector of means and standard deviations is shown in Figure 3.7.

The final solution was reached after 307 iterations with the estimated optimal vector $\hat{\mathbf{x}}^* = (-2.047 \times 10^{-7},\ -8.48 \times 10^{-7},\ 13.52 \times 10^{-7},\ 18.34 \times 10^{-7},\ 4.74 \times 10^{-7},\ -3.61 \times 10^{-7})$ and $f(\hat{\mathbf{x}}^*) = 2.4349 \times 10^{-8}$.

3.3 The CEM in other research and applications

In the previous sections the research which inspired the CEM was mainly cited. In this section, other research using the CEM is briefly described. Lü *et al.* (2008)

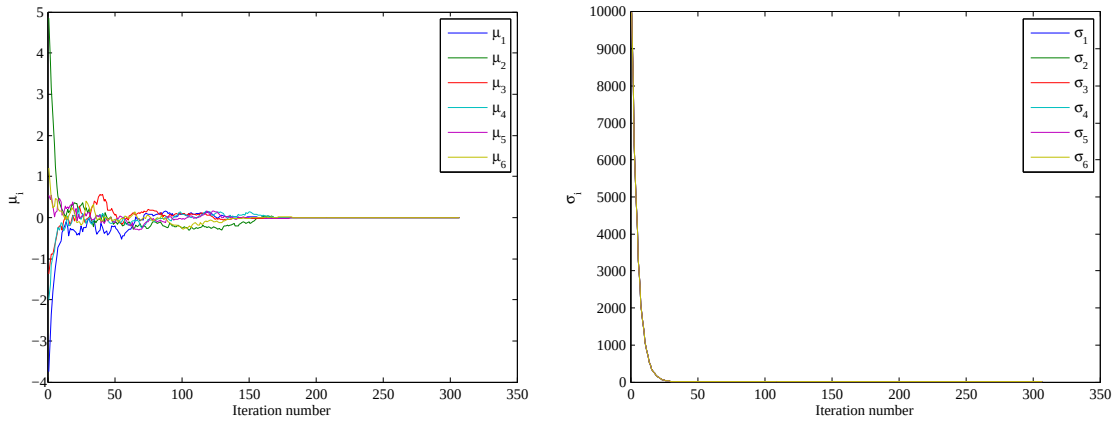

Figure 3.7: Rastrigin function optimisation: progress of the $\mathbf{v}$ vector.

developed a parallel leader-based algorithm using the CEM to solve the maximum clique problem. They tested the algorithm on 25 selected benchmark problems and established that their algorithm found most solutions significantly faster than other algorithms they included in their experiment. Evans *et al.* (2007) presented a general method for designing parallel cross-entropy algorithms to be used on *multiple instruction multiple data* (MIMD) machines and *message passing interface* (MPI) library routines. Application is on a discrete problem (max-cut problem) and a continuous problem (the Rosenbrock function). In Cohen *et al.* (2005), application of the cross-entropy method to project management is illustrated. The loading of a finite capacity, stochastic and dynamic project system is achieved by finding the number of projects (CONPIP, constant number of projects in progress) in the system to minimise the projects' average total stay time. The project system is stochastic since processing times of resources and arrival rates of project types, among others, are taken as stochastic. The CEM found good project loading parameters, even in noisy environments, and the authors suggest that the CEM be applied to the popular critical chain multiproject management methodology.

The CEM has also been applied to the vehicle routing problem and some of its variants, machine learning, the quadratic assignment problem and in scheduling (Rubinstein & Kroese, 2004). More examples of CEM applications can be found on the CEM web site at `http://www.cemethod.org` (cited on 10 August 2012).

A single-objective optimisation using the CEM in a South African study has been performed by Leonard (2011) under the supervision of the author. In that study, different apron layouts for a new development at Lanseria Airport in Gauteng,

South Africa, are assessed. It is projected that the largest airport in South Africa, the O.R. Tambo International Airport, also in Gauteng, will not be able to handle future air traffic. Lanseria is currently small and underutilised. Future expansion of this airport is a logical option, since it is also relatively close to the main centres in the Gauteng province. However, as there were no data available for current Lanseria operations, flight data for O.R. Tambo was used in that study.

Four different apron layouts were considered in that study. The assignment of arriving aircraft to a specific position (a "gate" in air control parlance) on the apron was the main driver of the study. The objective was to minimise passenger transport distance between the aircraft on the apron and the terminal building. This transport distance can be covered either on foot (the passengers have to walk), or by mechanical means. The gate assignment has an effect on the transport distance for both arriving and departing passengers, as well as aircraft movement time from the runway to designated parking areas or gates, and vice versa. Leonard (2011) used a real-life flight schedule of O.R. Tambo to simulate the arrivals and departures of flights with true passenger counts. She also simulated the aircraft movement on the apron while measuring passenger transport distances. This process considered small, medium and large aircraft, the size measure being based on the wingspan of the aircraft. Also, several operational rules had to be obeyed, for example two aircraft cannot pass each other when moving on concourses on the apron.

Leonard (2011) applied the CEM and computer simulation in a novel way by determining gate assignments that will minimise passenger transport distance. The simulation model followed a myopic approach and also assigned a number of future flights. At periodic points in time, when some flights had arrived and departed, hence belonging to the past, the assignment was revised to consider future arrivals. This resulted in effectively having a rolling planning window that moves through the flight schedule while performing good gate assignments. This approach also allowed for gate reassignment when flight delays occurred, making it a useful decision support tool. A research article of that study has been submitted for publication (Leonard & Bekker, 2012).

3.4 Summary: Chapter 3

The theoretical foundation and development of the cross-entropy method for optimisation was presented in this chapter, with focus on the continuous and discrete optimisation paradigms. The main algorithm of the CEM for optimisation was presented in Algorithm 1 (continuous case) and Algorithm 2 (discrete case). Both are arguably simple procedures.

The continuous optimisation algorithm was illustrated using four continuous, single-objective problems, namely De Jong's first function, the Rosenbrock function, the Shekel function and the Rastrigin function. The latter three functions are accepted benchmarks. They are scalable and the cross-entropy method found the optimum of each problem requiring a finite number of iterations, with the Rosenbrock function requiring the largest number of iterations (4 000).

The material presented in this chapter provides an understanding of the application of the CEM to single-objective optimisation. In the next chapter part of the research objective of this study is addressed, namely to develop an algorithm for MOO with the CEM as basis.

CHAPTER 4

MULTI-OBJECTIVE OPTIMISATION WITH THE CROSS-ENTROPY METHOD

The theoretical foundation of the *cross-entropy method* (CEM) for optimisation was presented in the previous chapter. With that background available, the proposed algorithm for *multi-objective optimisation* (MOO) using the cross-entropy method is presented in this chapter. The algorithm design is called the *Multi-objective Optimisation using the Cross-entropy Method* (*MOO CEM algorithm*) for short. Its required data structure and search mechanism are presented together with its exploration and exploitation ability, preservation of diversity, the ranking method and the preference type. The proposed algorithm is applied to benchmark problems, and values of quality performance indicators are presented by means of graphical displays of the true and known Pareto fronts. These benchmark problems are continuous and deterministic, and an application of the MOO CEM algorithm to a discrete, deterministic problem is also presented. Application of the proposed algorithm to these problems represents the first step in the development process of the MOO CEM algorithm.

4.1 The proposed MOO using the CEM

The multi-objective optimisation method using the cross-entropy method (MOO CEM) and its associated algorithm were published by the author in Bekker & Aldrich (2010). The proposed algorithm is based on Algorithm 1 outlined in Chapter 3, and is explained by narrative and pseudo-code.

The algorithm requires a working matrix consisting of N rows and $D + K + 1$ columns, where N is an arbitrary number of solutions (as in Algorithm 1), D is

the number of decision variables (DVs) and K is the number of objectives. Sample values of the first DV are stored in column 1, the second DV in column 2, and so on up to column D. The objective function values for objective 1 are stored in column $D+1$, for objective 2 in column $D+2$, and for objective K in column $D+K$. The last column is used to store the rank value ρ of each solution. The structure is shown in Table 4.1.

Decision variables				Objectives				Rank
X_{11}	X_{12}	$\dots$	X_{1D}	f_{11}	f_{12}	$\dots$	f_{1K}	ρ_1
$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
X_{N1}	X_{N2}	$\dots$	X_{ND}	f_{N1}	f_{N2}	$\dots$	f_{NK}	ρ_N

Table 4.1: Structure of the working matrix.

To form a sample vector $\mathbf{X}_i$ from the density $h_i(\cdot;\hat{\mathbf{v}}_{t-1})$, a truncated normal distribution is used for each DV. An example of a truncated normal distribution is shown in Figure 4.1. For the D DVs defined over ranges $[l_i, L_i]$, l_i is the lower limit and L_i the upper limit of DV $x_i, 1 \le i \le D$. The truncated normal distribution ϕ_i, defined in the range $[l_i, L_i]$ with mean μ_i and variance σ_i^2, is given by

$$\phi_i(x) = \begin{cases} 0, & x < l_i \\ \frac{h_n(x)}{\int_{l_i}^{L_i} h_n(x)\,\mathrm{d}x}, & l_i \le x \le L_i \\ 0, & x > L_i. \end{cases} \tag{4.1}$$

The function $h_n(x)$ is the normal probability density function defined on $-\infty < x < \infty$.

Using truncated distributions makes it easy to contain the search. As required by the CEM, an arbitrarily large value for σ_i is initially assigned, using $\sigma_i = 10\cdot(L_i - l_i)$. The first D columns of the working matrix are populated with sample values from each applicable truncated normal distribution.

Next, each of the objective functions is evaluated using the set of row vectors $X_{1i}, \dots, X_{Ni}$ of Table 4.1. This yields two or more performance vectors $f_j(\mathbf{X})$ with $1 \le j \le K$ as opposed to the single vector $f(\mathbf{X})$ in the original CEM. The $(1-\varrho)$-quantile (γ) cannot be estimated, because ranking one objective function will not necessarily yield a good estimate of γ for the other objective function(s).

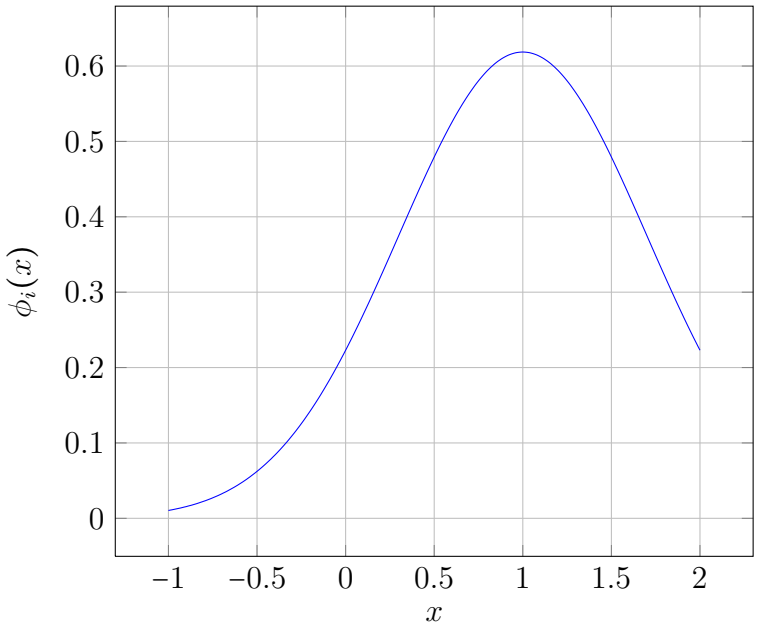

Figure 4.1: Truncated normal distribution on $-1 \leq x \leq 2$, $\mu = 1$, $\sigma = 1$.

The best combinations of objective function values are found by performing a Pareto ranking, using Algorithm 3 (Goldberg, 1989). The working matrix containing N rows and $D + K + 1$ columns is provided for the algorithm, and the columns numbered $D + i - 1$ $(1 < i \leq K)$ are sorted consecutively. After sorting column $i, D + 1 \leq i \leq D + K - 1$, the $(i + 1)$-th column is ranked. The ranking value of a solution indicates the number of solutions in the population which dominate that solution. This number is stored in column $D + K + 1$ of the working matrix. A solution with a ranking value of zero is a non-dominated solution. When all the solutions have been ranked, those with a ranking value not exceeding a specified threshold value ρ_E are appended to an elite vector called *Elite*, which represents the current (weakly) non-dominated set.

The values of the decision variables in the elite vector provided by Algorithm 3 are used to construct a histogram for each decision variable. The histograms provide guiding information for the MOO CEM algorithm and are maintained while the algorithm is searching for non-dominated solutions.

The histogram concept is implemented as follows: for a DV x_i that is defined in the range $[l_i, L_i]$, the lower limit of the first class is set equal to l_i, and the upper limit of the last class is set equal to L_i. Next, the upper boundary of the first class is set equal to the minimum value of the DV x_i in the elite vector,

Algorithm 3 Pareto ranking algorithm (Minimisation)

1: Input: working matrix $\mathbf{W}$ with N rows and $D+K+1$ columns, and user-selected threshold ρ_E.
2: $j \leftarrow D+1$.
3: Sort the working matrix $\mathbf{W}$ with the values in column j in *descending* order.
4: $r_p \leftarrow 1$.
5: $r_q \leftarrow r_p$.
6: If $\mathbf{W}(r_p, j+1) \geq \mathbf{W}(r_q+1, j+1)$, increment the rank value ρ_{r_p} in $\mathbf{W}(r_p, D+K+1)$.
7: $r_q \leftarrow r_q + 1$.
8: If $\mathbf{W}(r_p, D+K+1) < \rho_E$ and $r_q < N$, return to Step 6.
9: $r_p \leftarrow r_p + 1$.
10: If $r_p < N$, return to Step 5.
11: $j \leftarrow j+1$.
12: If $j < D+K-1$, return to Step 3, otherwise return the rows in $\mathbf{W}$ with rank value not exceeding ρ_E as the weakly or non-dominated vector **Elite**.

i.e. $\min(Elite(\cdot, i))$. The lower limit of the last class is equal to the maximum value of the DV in the elite vector, namely $\max(Elite(\cdot, i))$, and the upper limit is set equal to L_i. A number of equally sized classes are formed between these two boundaries using $(\max(Elite(\cdot, i)) - \min(Elite(\cdot, i)))/r$ if r of these classes are formed, resulting in a total number of $r+2$ classes (see Figure 4.2).

The class limits for the histogram of DV x_i are recorded in a vector $\mathbf{C}_i = \{c_{i1}, c_{i2}, \ldots, c_{i(r+2)}, c_{i((r+2)+1)}\}$, with $c_{i1} = l_i$ and $c_{i((r+2)+1)} = L_i$. Note that $\mathbf{C}_i$ contains $r+3$ elements because the histogram has $r+2$ classes, and that the class widths of the first class ($[c_{i1}, c_{i2}]$) and the last class ($[c_{i(r+2)}, c_{i((r+2)+1)}]$) may differ from each other and from the widths of the r classes.

The elite vector has the same columns as the working matrix shown in Table 4.1, and the values in column i, $1 \leq i \leq D$ are used to determine frequency values for the DV x_i. The DV values are classified according to the following rule: X_{ij} belongs to the class $[c_{i\kappa}, c_{i(\kappa+1)})$ if $c_{i\kappa} \leq X_{ij} < c_{i(\kappa+1)}$, $1 \leq \kappa \leq r+2$. The histogram frequency values are recorded in a vector $\mathbf{R}_i = \{\tau_{i1}, \tau_{i2}, \ldots, \tau_{i(r+1)}, \tau_{i(r+2)}\}$, where τ_{i1} is equal to the frequency count of decision variable x_i in the range $[c_{i1}, c_{i2})$, τ_{i2} represents the count in the range $[c_{i2}, c_{i3})$, and so on.

In preparation for the next iteration of the algorithm, the new population of possible solutions is formed proportionally according to the class frequencies for each DV: Suppose the elite vector contains E_r rows and that there are $\tau_{i\kappa}$ occurrences in class $[c_{i\kappa}, c_{i(\kappa+1)})$ for a given DV x_i. Then $\lfloor N\tau_{i\kappa}/E_r \rfloor$ values are created from

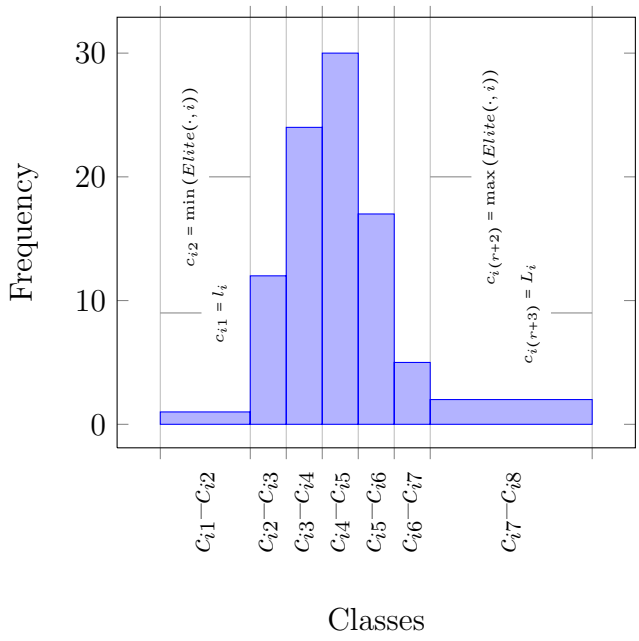

Figure 4.2: Example of a histogram for the DV x_i and $r = 5$.

this class range for this variable (the population size is N and $1 \leq \kappa \leq r + 2$). When the proportional numbers do not add up to N due to the rounding down of the proportion calculation, the small difference is arbitrarily added to the last class.

When generating observations from a class range $[c_{i\kappa}, c_{i(\kappa+1)}]$, temporary values $\mu'_{i\kappa}$ and $\sigma'_{i\kappa}$ are used. These values are associated with the specific histogram class ranges; for the class $[c_{i\kappa}, c_{i(\kappa+1)})$ corresponding to the DV x_i, the parameter estimators are $\mu'_{i\kappa} = c_{i\kappa} + U(c_{i(\kappa+1)} - c_{i\kappa})$, whereas $\sigma'_{i\kappa} = (c_{i(\kappa+1)} - c_{i\kappa})$, $1 \leq \kappa \leq r + 2$ and U is a uniformly distributed random number.

To prevent premature convergence, the histogram frequencies are adjusted during each iteration t using a preset probability of typically $p_h = 0.1$ to $p_h = 0.3$. To do so, the maximum frequency over all classes is determined for a given DV. The frequency in each class is then subtracted from this value, resulting in an inverted histogram, as shown in Figure 4.3. This ensures that search ranges that were given small proportions of population candidate allocations receive higher proportions of allocations, while search ranges with high proportions of population allocations receive fewer allocations after frequency inversion. The algorithm readjusts the frequencies according to the rankings returned by the candidates so that a class which does not contribute to the elite vector effectively becomes eliminated as the search progresses.

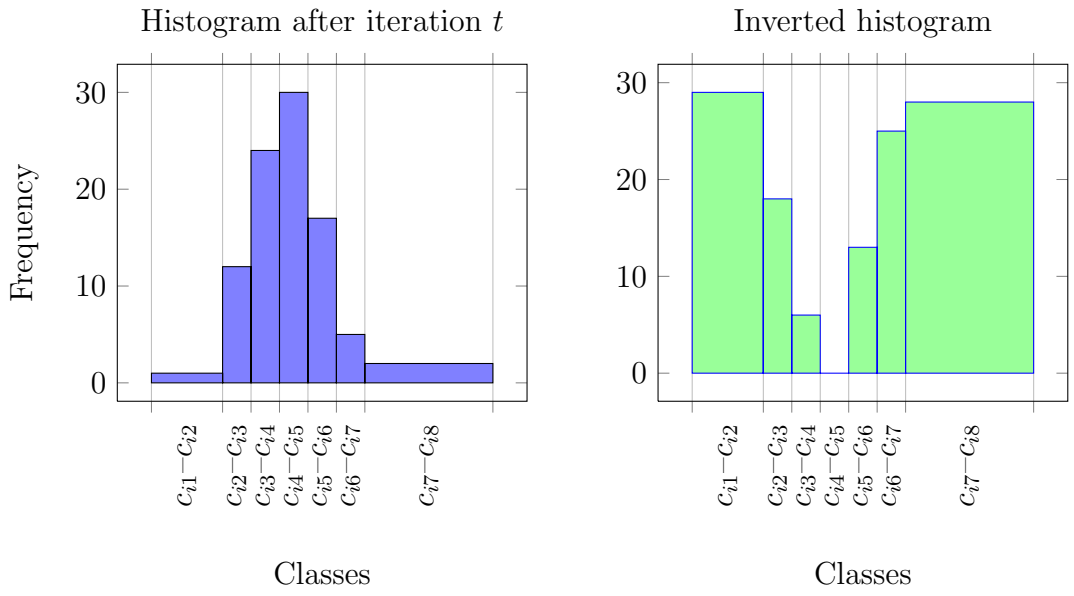

Figure 4.3: The effect of adjusting histogram frequencies for the DV x_i.

The histogram approach allows for the accommodation of discontinuous search spaces. However, here independent samples are drawn for the DVs, while one expects them to be correlated, because certain combinations of DV values yield the non-dominated objective function values. Increasing the number of classes as the search progresses makes it possible to maintain good combinations of DV values because the resolution of the decision variable spaces becomes finer. The number of classes should of course not grow too large, because then the algorithm will become inefficient. Constructing the histogram sets adds to the computational burden, which complies with the "No free lunch for optimisation" theorem (Wolpert & Macready, 1997) for single-objective optimisation and extended to MOO by Corne & Knowles (2003).

Since one deals with more than one objective and to ensure exploitation, the process described above is repeated several times as an outer loop of the algorithm. After each loop, the elite vector is ranked again and the number of classes of the histograms is incremented. The algorithm is presented in pseudo-code form as Algorithm 4.

The parameter vectors (μ_i, σ_i) are smoothed by means of (3.15) and the values in the DV columns of the elite vector. For example, the $\sigma_{1,t}$-value of the first DV

Algorithm 4 MOO CEM Algorithm

1: Set *Elite*= $\varnothing$, $t = 1$, $k = 1$.
2: Initialise variable vectors $\mathbf{X}_i = \varnothing$, $1 \leq i \leq D$, and compute initial objective values.
3: For each decision variable x_i, $1 \leq i \leq D$, initialise a histogram class vector $\mathbf{C}_i = \{c_{i1}, c_{i2}, \ldots, c_{i(r+2)}, c_{i((r+2)+1)}\}$ and histogram frequency vector $\mathbf{R}_i = \{\tau_{i1}, \tau_{i2}, \ldots, \tau_{i(r+1)}, \tau_{i(r+2)}\}$.
4: Set $i = 1$.
5: Set $\kappa = 0$.
6: Increment κ.
7: **for** each frequency element $\tau_{i\kappa}$ in $\mathbf{R}_i$ **do**
8: Generate a class-based $\tilde{\mathbf{v}}'$ in the range $[c_{i\kappa}, c_{i(\kappa+1)})$, $1 \leq \kappa \leq r+2$.
9: Generate a subsample $\mathbf{Y}$ according to the pdf $\phi_i(\mathbf{x}_i, \tilde{\mathbf{v}}')$
10: with $\mathbf{x}_i \in [c_{i\kappa},\ c_{i(\kappa+1)})$ and $|\mathbf{Y}| = \tau_{i\kappa}$, $1 \leq \kappa \leq r+2$.
11: Append $\mathbf{Y}$ to $\mathbf{X}_i$.
12: **end for**
13: If $\kappa < r+2$ return to Step 6.
14: Invert the histogram counts with probability p_h.
15: Increment i.
16: If $i \leq D$, return to Step 5.
17: Compute the NK objective function values using $\mathbf{X}_i$, $1 \leq i \leq D$.
18: Rank the objective function values using the Pareto ranking of Algorithm 3 with a relaxed $\rho_E = 2$ to obtain an updated elite vector **Elite**.
19: Form new histogram class vectors $\mathbf{C}_i$ and histogram frequency vectors $\mathbf{R}_i$ based on **Elite**, $1 \leq i \leq D$.
20: Use the values in **Elite** and compute $\tilde{\mathbf{v}}_{it}$ for all i, $1 \leq i \leq D$.
21: Smooth the vectors $\tilde{\mathbf{v}}_{it}$ for all i, $1 \leq i \leq D$, using (3.15).
22: If all $\sigma_{it} > \epsilon_c$ or less than the allowable number of evaluations has been done, increment t and reiterate from Step 4.
23: Rank the elite vector **Elite** using the Pareto ranking of Algorithm 3 with $\rho_E = 1$.
24: Increment k.
25: If k is smaller than the allowable number of loops, return to Step 2.
26: Rank the elite vector **Elite** using the Pareto ranking of Algorithm 3 with $\rho_E = 0$ to obtain the final elite vector.

is updated as follows

$$\hat{\sigma}_{1,t} = \alpha\tilde{\sigma}_{1,t} + (1-\alpha)\hat{\sigma}_{1,t-1} \tag{4.2}$$

after iteration t. This process is continued until the σ_i-value of each decision variable has decreased below a common threshold, ϵ_c. On algorithm termination, the elite vector should contain the solutions on the approximate front, as well as the associated DV values.

To ensure exploration and exploitation of the search, the initial ranking threshold is relaxed and a value of $\rho_E = 2$ is selected. This means that solutions having a ranking of zero to two are included in the initial elite vector, while the true dominating set will have a ranking value of zero for all solutions. When a new loop starts and a new population is formed, the elite vector is trimmed and the threshold is set to one. When the algorithm terminates, the existing elite vector is refined a last time. The threshold then used is zero, which means that all solutions selected are non-dominated.

4.2 Assessing the MOO CEM with deterministic, continuous benchmark problems

The research objective is to apply the proposed MOO CEM to solving multi-objective stochastic problems, but it is firstly applied to standard MOO test problems with known Pareto fronts. The test functions listed in Table 2.1 were evaluated after implementing the algorithm in Matlab® 2007b. These functions are all deterministic and continuous, although the decision spaces and solution spaces may be discontinuous. The number of evaluations for each problem was limited to 10 000. To put this number in context, one must consider the number of evaluations required by other algorithms on the same test problems. Zitzler *et al.* (2000) performed a comparison of eight algorithms on six test functions and used 25 000 evaluations in each test, while Shukla & Deb (2007) used 20 000 to 100 000 evaluations in their research of generating methods. Coello Coello (2009) pointed out that a current research trend in MOO is to find algorithms that can achieve good results with few objective function evaluations.

The quality performance of the proposed algorithm can be evaluated using the quality indicators listed in Subsection 2.3.4. Many accepted quality indicators of a unary nature are available (see Zhou *et al.*, 2011). The following indicators were used in conjunction with the test problems of Table 2.1:

MOP	Generation Distance GD	Spacing SP	Max. Pareto front error ME	Convergence CV	Execution time (s)	Size of Elite
MOP1	0.0000	0.2023	0.0038	0.0068	< 5	3 485
MOP2	0.0000	0.0006	0.0128	0.0112	< 25	1 397
MOP3	0.0013	0.0340	0.1751	0.0004	< 25	861
MOP4	0.0020	0.0157	0.2903	0.0957	< 25	559
MOP6	0.0001	0.0002	0.0328	0.0005	< 25	1 039
ZDT1	0.0012	0.0039	0.0235	0.0716	< 20	236
ZDT2	0.0012	0.0027	0.0386	0.2545	< 20	232
ZDT3	0.0039	0.0024	0.1300	0.0289	< 20	104

Table 4.2: Quality indicator values obtained for the test problems of Table 2.1.

1. Generation distance (GD; see (2.7)).
2. Spacing (SP; see (2.8)).
3. Maximum Pareto front error (ME; see (2.10)).
4. Convergence (CV; see (2.12)).

The indicator values obtained from the tests with the proposed MOO CEM algorithm are shown in Table 4.2.

Note that the execution time is listed for information only; it is not considered as a quality indicator. The tests were performed on an IBM laptop with two Intel Core i5 cores and a 2GB memory. The true Pareto fronts and the approximate fronts generated by the MOO CEM algorithm for the various problems are shown in Figures 4.4 to 4.7.

To better understand the behaviour of the MOO CEM algorithm, the trends of the CE vector $\mathbf{v}$ for each problem were also collected during algorithm assessment. Some of these are shown in Figure 4.8 (MOP1), Figure 4.9 (MOP4) and Figure 4.10 (ZDT1). The mean(s) and standard deviation(s) shown are those of the values of the DV(s) in the *Elite* set, calculated after each termination of the secondary loop in Algorithm 4 (Step 22).

With reference to Figure 4.8, the trend of μ_1 and σ_1 for MOP1 can be explained as follows: the mean and standard deviation are assigned arbitrary values from the MOP1 definition range $-10^5 \leq x_1 \leq 10^5$. Then the MOO CEM algorithm drives the standard deviation towards a non-zero finite value, while the mean

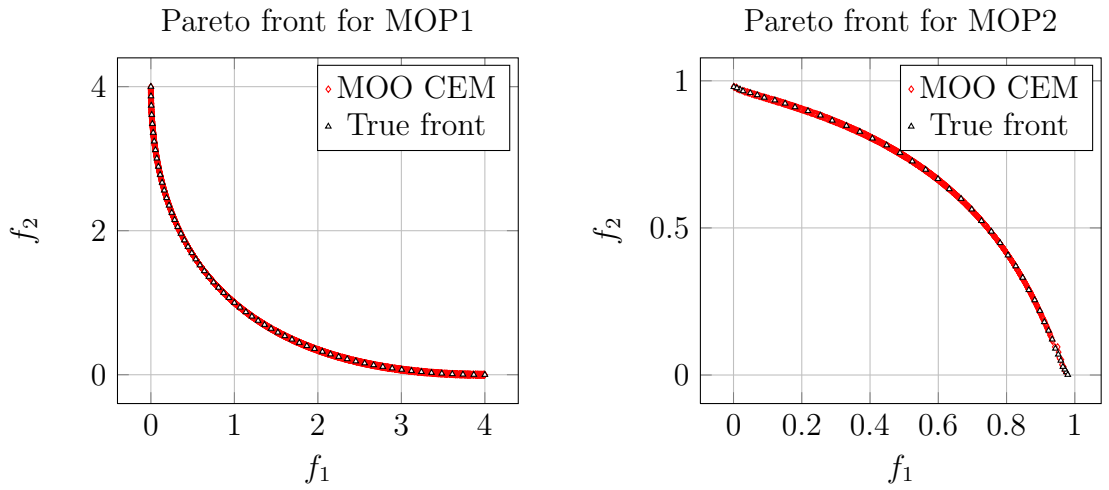

Figure 4.4: Approximate fronts for MOP1 and MOP2 obtained by the MOO CEM.

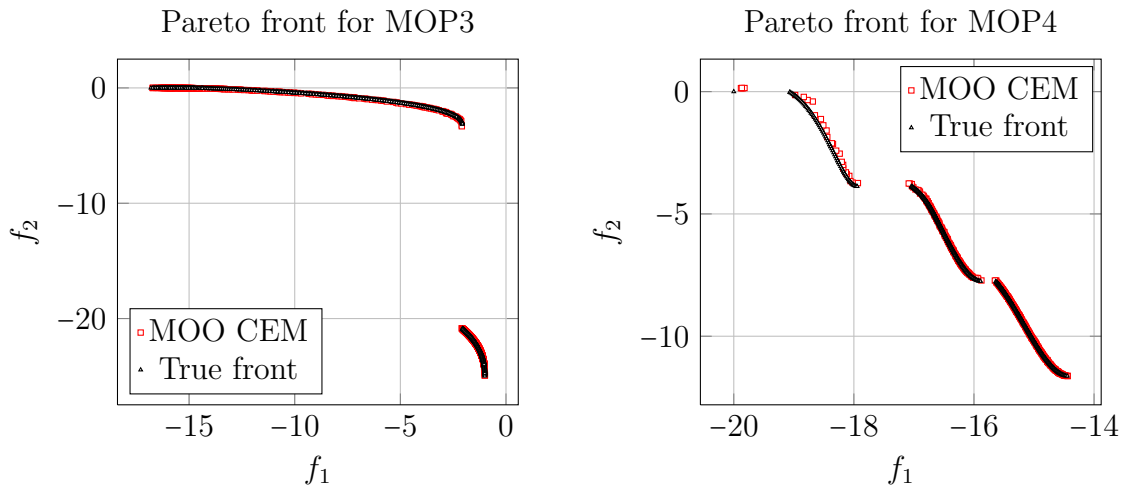

Figure 4.5: Approximate fronts for MOP3 and MOP4 obtained by the MOO CEM.

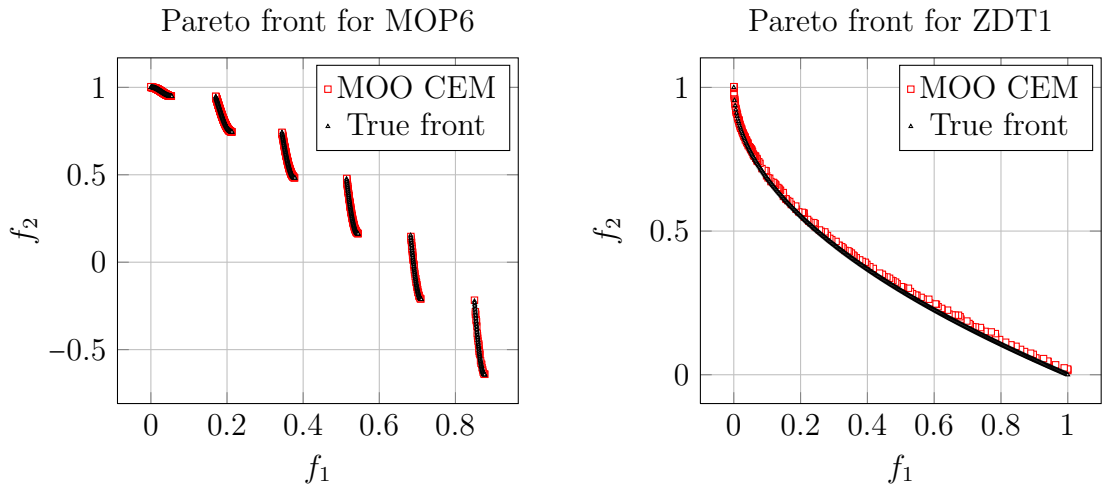

Figure 4.6: Approximate fronts for MOP6 and ZDT1 obtained by the MOO CEM.

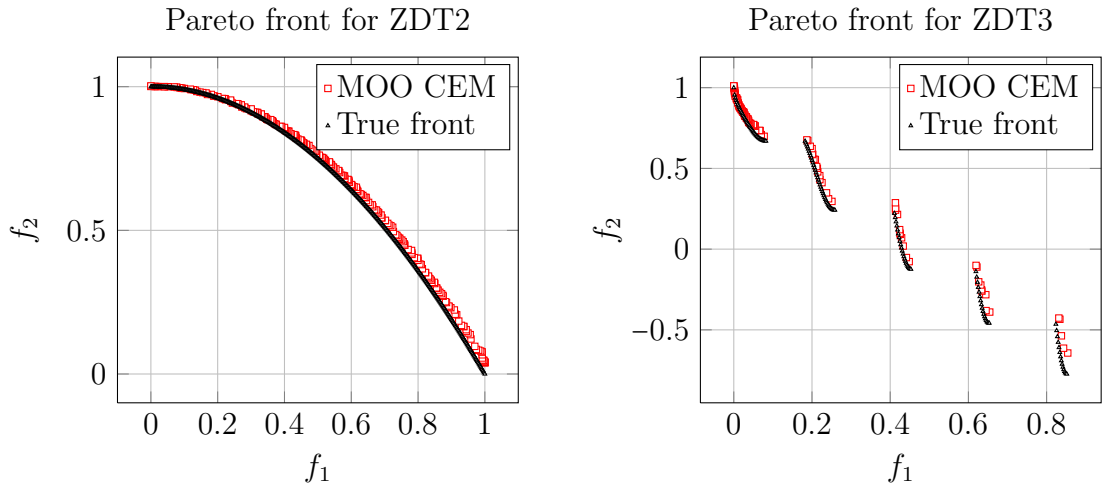

Figure 4.7: Approximate fronts for ZDT2 and ZDT3 obtained by the MOO CEM.

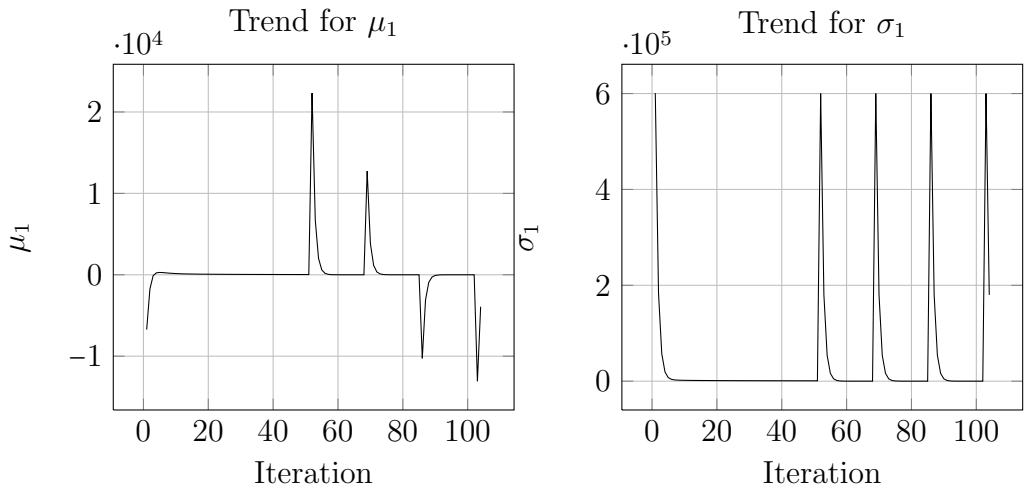

Figure 4.8: Trends of the CE vector $\mathbf{v}$ for MOP1.

converges towards the *range* of optimal values. Once this convergence is achieved, the algorithm resets while preserving the current *Elite* set, as explained previously. On resetting, the mean and standard deviation are assigned new values. For this particular run of the algorithm (shown in Figure 4.8), resetting occurred at iterations 52, 67, 82 and 97. Note that both graphs show these events. It is known that the Pareto-optimal values of MOP1 lie in the range $[0, 2]$, and the mean μ_1 converges to values in this range.

A subset of the raw data of MOP1, on which Figure 4.8 is based, is shown in Table 4.3 to clarify the convergence of the mean and standard deviation. Sets of data were extracted where convergence occurred, and the search redirected. A horizontal line separates each objective function value group, with the last row in each set showing the iteration number and the newly assigned values for μ_1 and σ_1.

The trend for MOP4 is shown in Figure 4.9, for $-5 \leq x_i \leq 5,\ i = 1, 2, 3$. The true DV space is $x_i \in [-1, 0]$, and the estimated means vary, but approach that region. The number of spikes in the graph is more than that in the graph for MOP1, because MOP1 has only one DV. MOP4 has three decision variables and the search has to change direction on more occasions.

The trend for ZDT1 is shown in Figure 4.10. Here, for clarity, only x_1, x_2 and x_{30} are shown. It is known that $x_1 \in [0, 1]$, while $x_i = 0$ for $i = 2, \ldots, 30$. The graph shows an oscillating behaviour for x_1, since it can assume any value in $[0, 1]$, while x_2 and x_{30} tend to approach zero, which is the true value for these variables. The

Iteration	μ_1	σ_1	Iteration	μ_1	σ_1
50	−19.208	455	76	0.634	12.398
51	−18.854	451	77	0.894	4.130
52	−3 757.173	600 000	78	0.972	1.650
59	0.189	131.839	79	0.996	0.906
60	0.769	39.983	80	1.003	0.682
61	0.944	12.424	81	1.005	0.615
62	0.995	4.155	82	26 720.437	600 000
63	1.010	1.674	93	1.055	1.644
64	1.015	0.928	94	1.021	0.899
65	1.016	0.704	95	1.011	0.676
66	1.016	0.636	96	1.008	0.609
67	−18 861.149	600 000	97	26 610.635	600 000

Table 4.3: Specific values of $\mathbf{v}$ for MOP1.

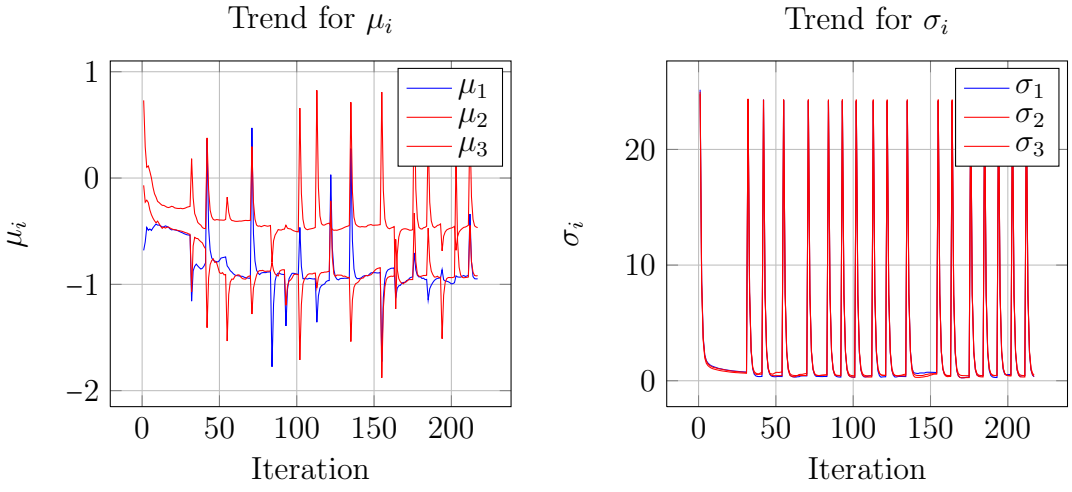

Figure 4.9: Trends of the CE vector $\mathbf{v}$ for MOP4.

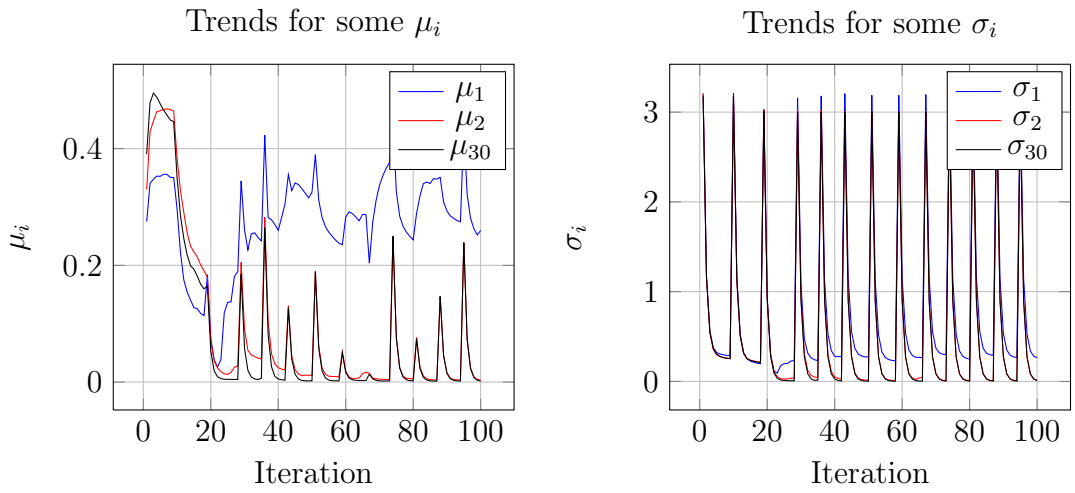

Figure 4.10: Trends of part of the CE vector $\mathbf{v}$ for ZDT1.

number of spikes in the graph again shows changes of search direction, and this behaviour can be attributed to Steps 2 and 25 of Algorithm 4.

After testing the algorithm on continuous, deterministic problems, the following was noted:

1. A large value for N is detrimental to the aim of the algorithm, because it increases the number of objective function evaluations without contributing significantly to the solution quality. It is therefore recommended to use $30 \leq N \leq 100$.

2. The elite vector can grow very fast, resulting in slowdown of the algorithm due to the required ranking per iteration. Note that the complexity of the ranking algorithm used is $O(KN^2)$. (Faster algorithms do exist, see for example Qu & Suganthan, 2009 and Fang *et al.*, 2008).

3. Since good mutual solutions are collected via the modified CEM, the value of ϵ_c does not need to be as small (10^{-5}, for example) as in single-objective problems solved by Rubinstein & Kroese (2004). However, some test functions returned better quality solutions with smaller values for ϵ_c, while others still maintained good quality solutions with larger values for ϵ_c. During the research for this chapter, values for $\epsilon_c \in [0.1, 1]$ were found to be sufficient.

4. The probability of inverting the decision variable histogram should be in the range 0.1 to 0.3.

4.3 Assessing the MOO CEM with a discrete benchmark problem

In the previous section, the merits of the proposed MOO CEM algorithm were assessed by applying it to deterministic, continuous benchmark problems. In this section, the algorithm is applied to a discrete, deterministic problem, and the *vehicle routing problem* (VRP) was chosen for this purpose, since it is an NP-hard problem.

There is an abundance of publications in the literature on the VRP and its variants, and detail will not be repeated here. For a comprehensive reference, see Toth & Vigo (2002), and Laporte (2007) for an overview of the classical VRP. In essence, a number of vehicles are routed from a central depot to geographically dispersed visiting nodes or customers. The vehicle performs some service at each node, and each vehicle has limited capacity which constrains the number of nodes that can be visited per trip. Each vehicle must return to the depot, and Dantzig & Ramser (1959) considered the "truck dispatching problem" a generalisation of the travelling salesperson problem (TSP). Toth & Vigo (2002) summarise the constraints as follows:

1. Each route visits the depot vertex.
2. Each customer vertex is visited by exactly one route (within the specified time window).
3. The total demand of the customers visited by a route does not exceed the capacity of the separate vehicles.

A variant of the VRP is the *VRP with time windows* (VRPTW), which has a further variant, namely soft time windows, which is denoted as VRPSTW. In the VRPTW, a vehicle may only arrive during a specified time window at a certain node. The VRPSTW, on the other hand, allows that vehicles can arrive after the time window has closed, although this is often associated with a penalty cost for late arrival. The MOO CEM algorithm was applied to the VRPSTW by Hauman & Bekker (2012) and is briefly presented here.

4.3.1 MOO and the VRP

In MOO of the VRP, researchers typically attempt to minimise the number of vehicles and the total travel distances, and there are many MOO variants of the VRP available in the literature. Jozefowiez *et al.* (2008) provided a review of MOO VRPs and group the methods used to solve these multi-objective problems into scalar methods, Pareto methods and a third category in which different objectives are considered separately.

Evolutionary algorithms are used in most cases of Pareto methods. A hybrid multi-objective evolutionary algorithm is proposed by Tan *et al.* (2006) while Ombuki *et al.* (2006) use a MOGA. Geiger (2008) states that the relaxation of the time window restriction (VRPSTW) allows for a more practical multi-objective formulation and investigates the influence of this relaxation and other problem characteristics on genetic operators in evolutionary algorithms.

Lau *et al.* (2009) developed a MOGA that uses fuzzy logic to adjust the cross-over rate and mutation rate dynamically, and they consider travel distance and travel time as objectives. These objectives are not always positively correlated. Recently Garcia-Najera & Bullinaria (2011) proposed an improved MOEA which uses a similarity measure to enhance the diversity of solutions. When compared to general evolutionary methods, such as the popular NSGA-II (Deb *et al.*, 2002), this method shows improvements — in particular in preserving high diversity before settling on a solution.

4.3.2 Benchmark problems for the VRP

Solomon (1987) developed six sets of benchmark problems that have since been used in comparing different VRP solution methods. The Solomon benchmark set has been extended and used in most of the literature on multi-objective vehicle routing, but recently, Castro-Gutierrez *et al.* (2011) found that classic test instances, such as the Solomon set, are not entirely suitable for MOO, since objectives such as the number of vehicles and the total travel distance were, in fact, found to be in harmony. Castro-Gutierrez *et al.* (2011) provided a set of problem instances for multi-objective test cases, and identified five objectives to be used in VRP analyses: the number of vehicles ($Z1$), the total travel distance ($Z2$), the makespan (travel time of longest route) ($Z3$), the total waiting time when vehicles arrive before the time window ($Z4$) and the total delay time when vehicles arrive after the time

window ($Z5$). Optimisation of some combinations of these objectives are presented in this section, but the VRPSTW is first defined with reference to these objectives.

4.3.3 The VRP with soft time windows

To define the VRPSTW, consider a fleet of vehicles $\mathcal{V}$, a set of customers $\mathcal{C}$ and a directed graph $\mathcal{G}$. $\mathcal{N}$ is the set of vertices, $0, 1, ..., n+1$, where 0 is the starting point at the depot, and $n+1$ the depot returning point (Kallehauge *et al.*, 2005). Define the decision variable

$$x_{ij\nu} = \begin{cases} 1, & \text{if vehicle } \nu \text{ drives directly from vertex } i \text{ to vertex } j, \\ 0, & \text{otherwise.} \end{cases}$$

The capacity of each vehicle is C_v, the demand of customer i is D_i, the cost (or distance) c_{ij} and time t_{ij} associated with each arc (i, j) where $i \neq j$. Customer i specifies the time window $[a_i, b_i]$, and in the case of hard time windows, the vehicle must arrive at the customer before time b_i. In the case of soft time windows, a delay time is logged. Arriving before the time window starts incurs a waiting time until time a_i when the service can start. The variable $s_{i\nu}$ denotes the time at which vehicle ν starts service at customer i. This is defined for every vehicle ν and customer i, unless vehicle ν does not service customer i. It is assumed that the time window of the depot always starts at zero ($s_{0\nu} = 0$), and the time back at the depot (although no service is required) is defined as $s_{(n+1)\nu}$.

The objectives in the mathematical model of Kallehauge *et al.* (2005) for the multi-objective VRPSTW are to

$$\text{Minimise } \nu \tag{4.3a}$$

$$\text{Minimise } \sum_{\nu \in \mathcal{V}} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} c_{ij} x_{ij\nu} \tag{4.3b}$$

$$\text{Minimise } \left(\max_{\nu} \left(s_{(n+1)\nu} \right) \right) \tag{4.3c}$$

$$\text{Minimise } \sum_{\nu \in \mathcal{V}} t_w^{\nu} \tag{4.3d}$$

$$\text{Minimise } \sum_{\nu \in \mathcal{V}} t_d^{\nu} \tag{4.3e}$$

subject to

$$\sum_{\nu\in\mathcal{V}}\sum_{j\in\mathcal{N}} x_{ij\nu} = 1 \qquad \forall i \in \mathcal{C}, \tag{4.4a}$$

$$\sum_{i\in\mathcal{C}} D_i \sum_{j\in\mathcal{N}} x_{ij\nu} \leq C_v \qquad \forall \nu \in \mathcal{V}, \tag{4.4b}$$

$$\sum_{j\in\mathcal{N}} x_{oj\nu} = 1 \qquad \forall \nu \in \mathcal{V}, \tag{4.4c}$$

$$\sum_{i\in\mathcal{N}} x_{ih\nu} - \sum_{j\in\mathcal{N}} x_{hj\nu} = 0 \qquad \forall h \in \mathcal{C}, \forall \nu \in \mathcal{V}, \tag{4.4d}$$

$$\sum_{i\in\mathcal{N}} x_{i,n+1,\nu} = 1 \qquad \forall \nu \in \mathcal{V}, \tag{4.4e}$$

$$x_{ij\nu}(s_{i\nu} + t_{ij} - s_{j\nu}) \leq 0 \qquad \forall i, j \in \mathcal{N}, \forall \nu \in \mathcal{V}, \tag{4.4f}$$

$$t_w^\nu = \sum_{i\in s_{i\nu}<a_i} (a_i - s_{i\nu}) \qquad \forall \nu \in \mathcal{V}, \tag{4.4g}$$

$$t_d^\nu = \sum_{i\in s_{i\nu}>b_i} (s_{i\nu} - b_i) \qquad \forall \nu \in \mathcal{V}, \tag{4.4h}$$

$$x_{ij\nu} \in \{0, 1\} \qquad \forall i, j \in \mathcal{N}, \forall \nu \in \mathcal{V}. \tag{4.4i}$$

Five conflicting objectives are defined in (4.3a) to (4.3e) and optimised in pairs. The five objectives are shown in Table 4.4 with the labels defined by Castro-Gutierrez *et al.* (2011).

Objective	Label	Expression
Number of vehicles	Z1	(4.3a)
Total travel distance	Z2	(4.3b)
Makespan of tasks	Z3	(4.3c)
Total vehicle waiting time	Z4	(4.3d)
Total vehicle delay time	Z5	(4.3e)

Table 4.4: Objectives of the VRPSTW (Castro-Gutierrez *et al.*, 2011).

The model of Kallehauge *et al.* (2005) has been adapted for soft time windows as follows: t_w^ν is the total time a vehicle waits for a time window to start on a route, and is determined by (4.4g), while (4.4h) calculates the total delay time t_d^ν of customers on a route waiting for vehicles that arrive after the close of a time window. The other constraints follow the original model: the constraint in (4.4a) ensures that each customer is visited exactly once, the capacity constraint is (4.4b), and (4.4i) is the integrality constraint. The constraints (4.4c), (4.4d) and (4.4e) ensure that each vehicle leaves the depot, arrives at a customer and then proceeds

to the next customer, and that all vehicles end at the depot. This model with the soft time windows is used as platform for further analysis.

4.3.4 The VRP and the CEM

The model formulation for solving the VRP case with soft time windows with the CEM is considered in this section.

The discussion refers to the discrete case of CEM optimisation and is based on (3.22) and (3.23). Single-objective optimisation of the VRP with the CEM requires a transition probability matrix P with elements p_{ij}, where p_{ij} is the probability to travel to node j when at node i. De Boer *et al.* (2005) showed that these probabilities are updated as

$$p_{ij} = \frac{\sum_{l=1}^{N} I_{\{S(\boldsymbol{X}_l)\leq\gamma\}} I_{\{\boldsymbol{X}_l \in \mathcal{X}_{ij}\}}}{\sum_{l=1}^{N} I_{\{S(\boldsymbol{X}_l)\leq\gamma\}}}. \tag{4.5}$$

In (4.5), there are N possible tours, and $S(\boldsymbol{X}_l)$ is a minimisation performance measure, such as the distance travelled. $\mathcal{X}_{ij}$ is the set of vectors in which the transition from i to j is made ($x_{ij} = 1$). For MOO of the VRPSTW, the condition $S(\boldsymbol{X}_l) \leq \gamma$ is changed to consider the ranking of solutions of the population, in which case (4.5) becomes

$$p_{ij} = \frac{\sum_{q=1}^{N} I_{\{\rho_q=0\}} I_{\{x_{ij}=1\}}}{\sum_{q=1}^{N} I_{\{\rho_q=0\}}}. \tag{4.6}$$

The ranking value of $\rho_q = 0$ ensures that the transition probabilities are updated according to the best solutions in the population of a given iteration. The structure of the optimisation model is shown in Figure 4.11 and the route construction is achieved by means of Algorithm 5.

The algorithm applied to the VRPSTW is given in pseudo-code form as Algorithm 6, and is based on Algorithm 1 and (4.6). The probability matrix P is smoothed as usual, that is $P_t \leftarrow \alpha P_t + (1-\alpha)P_{t-1}$.

The algorithms and model structure in Figure 4.11 were applied to the benchmark problems proposed by Castro-Gutierrez *et al.* (2011), and the objectives $(Z1, Z3)$, $(Z1, Z5)$, $(Z2, Z3)$, $(Z2, Z5)$, $(Z4, Z3)$ and $(Z4, Z5)$ were studied. Each problem is distinguished with a specific label, for example, "50_d0_tw1" denotes the benchmark problem with 50 customers, a demand/capacity profile number 0 and time window profile 1. There are three capacity and/or demand constraints

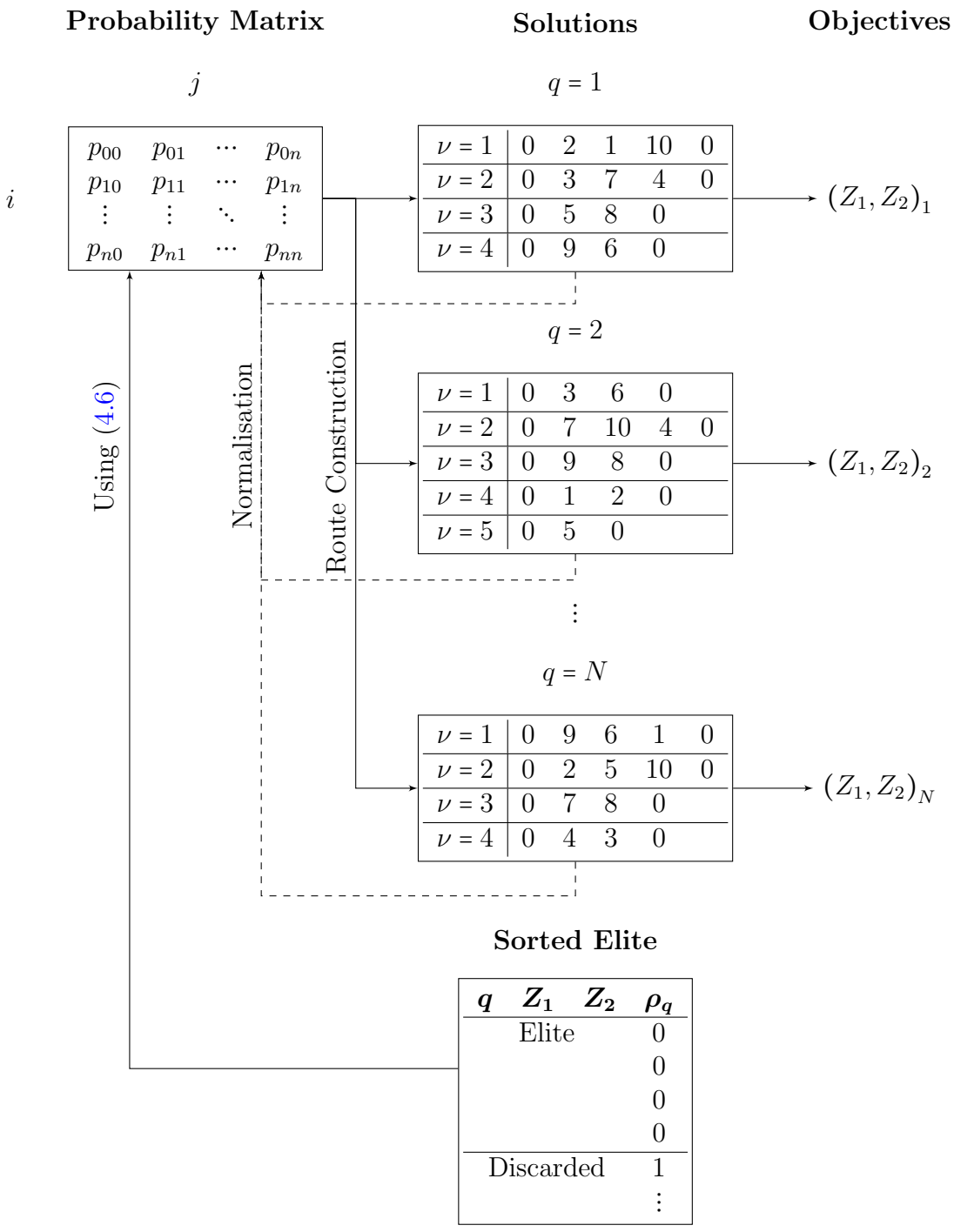

Figure 4.11: The structure of the VRP optimisation model (Hauman & Bekker, 2012).

Algorithm 5 Construction of routes for the VRPSTW

Let *Depot* be customer $i+1$ (next customer).
while all customers are not visited **do**
    Previous Customer $i \leftarrow$ Customer $i+1$.
    for all customers not yet visited **do**
        $\text{feas}_1 \leftarrow$ (capacity of truck).
        $\text{feas}_2 \leftarrow$ (time window (in the case of hard time windows)).
        $\text{feas}_3 \leftarrow$ (cut-off time back at the depot).
        Feasible $\leftarrow \text{feas}_1$ **and** feas_2 **and** feas_3.
        if Not Feasible **then**
            Change probability to be visited from current customer to 0.
        end if
    end for
    if No customers are feasible **then**
        The next customer is *Depot*.
        Increment routes: $\nu \leftarrow \nu + 1$.
        return to 2.
    else
        Calculate the new weighted row of the probability matrix.
        Sample a customer from this row.
        Add this customer to the route as customer $(i+1)$.
    end if
end while

(0–2, 2 being the tightest) and five different time window profiles (0–4, 4 being the tightest). The cases studied were 50_d(0-2)_tw(0-4) and 250_d2_tw(1-4). Some of the results thus obtained are presented next.

4.3.5 Results: experimenting with the VRP

Tests were conducted with parameters set to $N = 2\,000$, $\alpha = 0.9$, $\tau_m = 25$ and $N_m = 10$. The results are shown as paired plots for every pair of conflicting objectives; see, for example, Figure 4.12 and Figure 4.13. In Figure 4.12, the progression of the approximation front is shown through 10 iterations, and in the adjacent plot on the right (Figure 4.13), the final approximation front is shown. The solution set for a particular solution point on the approximation front is shown in subsequent tables with the routes for the set of vehicles ($\mathcal{V}_\nu$). Table 4.5 and Table 4.6 list the reference numbers of the customers in the order that they were visited by each vehicle. Castro-Gutierrez *et al.* (2011) specified these reference numbers. In Table 4.5, for example, it is shown that six vehicles were used, and

Algorithm 6 MOO VRPSTW with the CEM

Let P be the transition matrix of probabilities, N_m the maximum number of loops, τ_m the maximum number of evaluations per loop and $\boldsymbol{\mu}$ the vector of means of the objectives in the elite array.
$L_c \leftarrow 0$.
repeat
    $t \leftarrow 0$.
    Initialise P_t and $\boldsymbol{\mu}_t$.
    repeat
        $t \leftarrow t + 1$.
        Construct routes using Algorithm 5 and P_t.
        Evaluate routes.
        Rank the solutions using the threshold $t_h = 2$.
        Build the elite set from the top ranked solutions.
        Update $\boldsymbol{\mu_t}$.
        Update P_t using (4.6).
        Smooth $P_t \leftarrow \alpha P_t + (1 - \alpha) P_{t-1}$.
    until $|\boldsymbol{\mu}_t - \boldsymbol{\mu}_{t-1}| \leq \epsilon$ or $t \geq \tau_m$.
    Rank with $t_h = 1$ and update the elite set.
    $L_c \leftarrow L_c + 1$.
until $L_c > N_m$.
Rank with $t_h = 0$ and update the elite set.
return the elite set.

vehicle number 1 (V1) visited eight customers. Also, there are exactly 50 non-zero labels in the table, and the label sets represented by each column are mutually exclusive.

The approximation front in Figure 4.13 has two good solutions which are labelled "A_1" and "A_2". The decision maker can choose to have six vehicles and a total makespan of 27 660 hours ("A_1"), or pay for a seventh vehicle which will reduces the makespan to 27 480 hours ("A_2").

In Figure 4.15, one good solution is found, labelled "B". This indicates that five vehicles will make the delay time zero, and that at least five vehicles are required.

A map of the routes in Table 4.5 is shown in Figure 4.16.

Solutions for other combinations of objectives are included in Appendix B.

In this section, the application of the proposed MOO CEM algorithm to a discrete, deterministic problem was described. The VRPSTW was chosen as test platform, and benchmark problems with known conflicting objectives were chosen from a recent publication (Castro-Gutierrez *et al.*, 2011). Although no reference

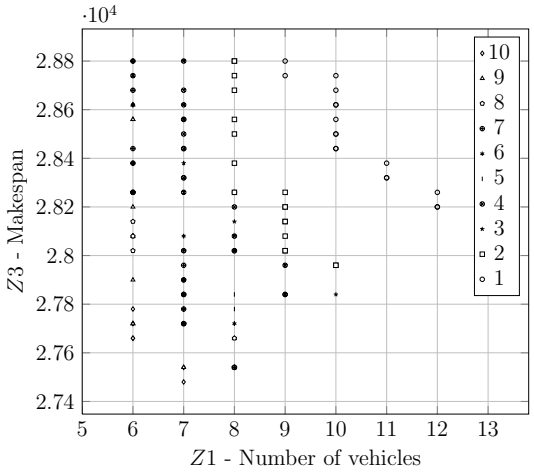

Figure 4.12: Front progression of 50_d1_tw4 for $Z1$ vs $Z3$.

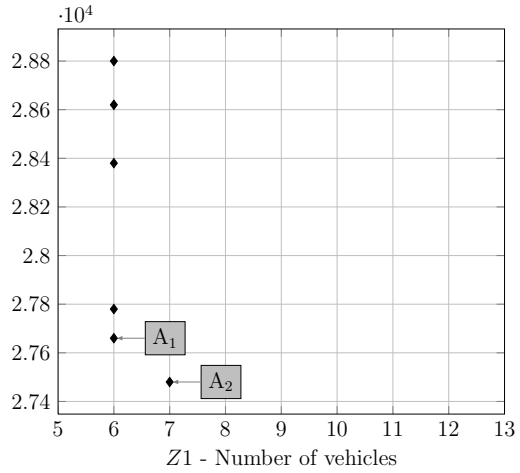

Figure 4.13: Final approximate front of 50_d1_tw4 for $Z1$ vs $Z3$.

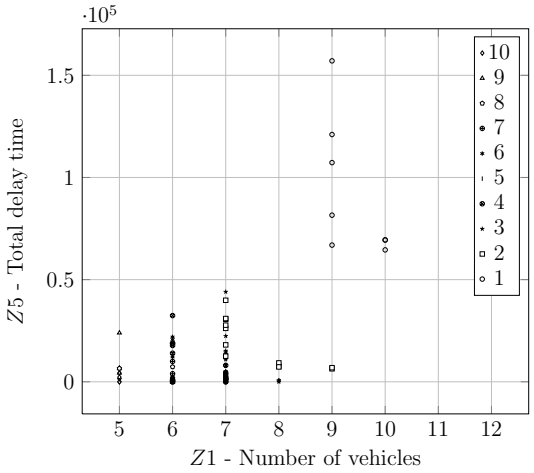

Figure 4.14: Front progression of 50_d1_tw4 for $Z1$ vs $Z5$.

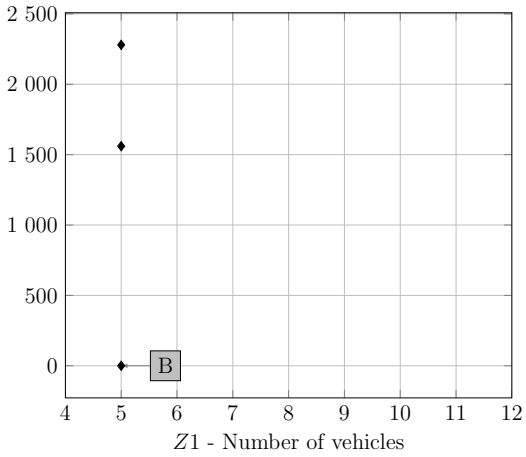

Figure 4.15: Final approximate front of 50_d1_tw4 for $Z1$ vs $Z5$.

solutions were available for comparison, the approximate fronts obtained make sense. In some cases there is only one solution; for example, in problem 50_d1 _t4, two cases are noted:

1. The pair $(Z1, Z5)$ or "Number of vehicles" vs "Total delay time" has three solution points. The objective $Z1$ is discrete, and in Figure 4.14, it may be seen that the delay time $Z5$ is zero for different numbers of vehicles, but only one value makes sense to choose, as was shown in Figure 4.15. Here the delay time is zero for the minimum number of vehicles (five).

V1	V2	V3	V4	V5	V6
0	0	0	0	0	0
649	1 725	2 000	1 173	2 104	1 870
1 384	856	430 148	430 030	1 714	2 107
430 761	1 888	1 897	1 703	430 378	1 777
2 044	1 362	1 813	1 781	948	1 203
2 121	1 678	1 463	669	430 625	1 389
2 149	2 073	486	2 003	0	1 875
430 804	1 588	1 509	0		974
482	1 686	661			1 235
0	2 152	907			2 138
	106	430 471			0
	0	1 721			
		2 007			
		0			

Table 4.5: Routes of solution A_1 in Figure 4.13, 50_d1_tw4 ($Z1$ vs $Z3$).

V1	V2	V3	V4	V5
0	0	0	0	0
2 104	1 813	1 173	649	1 870
856	1 725	1 384	2 107	1 777
1 888	1 897	1 714	907	1 203
430 148	1 678	430 378	2 073	1 703
1 463	430 030	2 044	486	1 389
1 588	974	1 875	1 509	669
106	430 761	1 235	1 686	482
661	1 781	2 121	2 152	0
430 471	2 149	2 138	1 362	
2 000	430 804	0	1 721	
430 625	2 003		0	
2 007	0			
948				
0				

Table 4.6: Routes of solution B in Figure 4.15, 50_d1_tw4 ($Z1$ vs $Z5$).

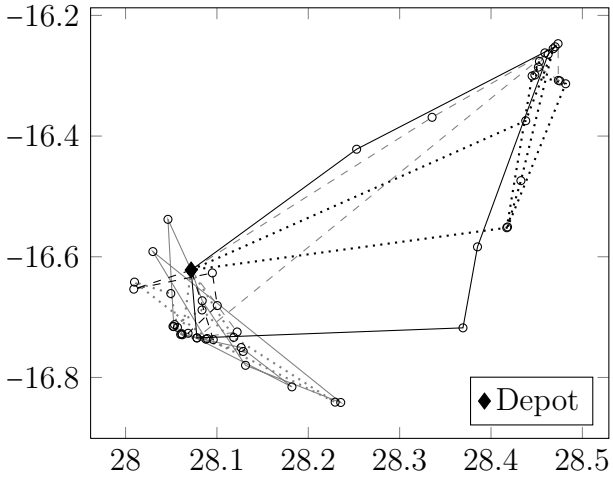

Figure 4.16: Map of routes of solution A_1, 50_d1_tw4 for $Z1$ vs $Z3$.

2. The pair $(Z4, Z5)$ or "Total waiting time" vs "Total delay time " has one solution point, where both are zero (Figure B.8 on p. B-4).

Other sets of objective pairs present more solutions; for example, the pair $(Z2, Z5)$ represents distance and time, and different distances usually require different times to travel by all the vehicles. These observations build confidence and the author can assume that the MOO CEM algorithm is able to solve MOO problems of discrete, deterministic nature. In this section, results for problems

with 50 visiting points were presented, and results for problems with 250 visiting points are included in Appendix B.

4.4 Summary: Chapter 4

In conclusion, the properties of the MOO CEM algorithm are stated. The MOO CEM algorithm

1. is population-based,
2. uses a statistical basis (it is thus not biologically inspired),
3. employs elitism to preserve weakly-dominated and eventually non-dominated solutions during its search,
4. employs a ranking scheme of objective function values to find non-dominated solutions,
5. is according to Operations Research solution classification, an *a posteriori* technique, the decision maker thus searches for solutions before making decisions; see Coello Coello *et al.* (2007:31,54),
6. achieves *proximity* to the Pareto front via the convergence mechanism of the CE method, and
7. ensures *diversity* of the search via the histogram implementation and probability-based inversion, as explained in Section 4.1.

The proposed MOO CEM algorithm was presented in this chapter, and was applied to some benchmark evaluations as a means of its quality performance testing. The algorithm was applied to deterministic problems of continuous (Table 2.1) and discrete (the VRPSTW) nature, and the empirical evidence presented shows that there is merit in applying the MOO CEM algorithm to more complex, practical problems, since it requires a relatively low number of evaluations to generate solution sets approximating the true Pareto set. The experiments using complex, dynamic and stochastic optimisation problems are presented in the next chapter. These problems include textbook problems as well as real-world problems.

CHAPTER 5

MULTI-OBJECTIVE OPTIMISATION APPLICATIONS OF THE MOO CEM ALGORITHM

In the previous chapter the MOO CEM algorithm was introduced and its potential assessed using a subset of benchmark problems. Empirical evidence and quantitative quality indicators suggest the potential usefulness of the algorithm, especially in terms of its relatively low computing budget. Since the research hypothesis focuses on dynamic stochastic problems, applying the MOO CEM algorithm to problems of that class is required.

In this chapter, a variety of problems and problem variants are presented. First, the classical (s, S) inventory problem is presented as a dynamic, stochastic MOO problem, and the MOO CEM algorithm is applied. Then a well-researched problem, the buffer allocation problem is studied as an MOO problem. Analyses of several variants of the linear buffer allocation problem as well as the network buffer allocation problem are performed. Three practical problems are also considered: the first problem deals with the design of a reconfigurable manufacturing system, of which the throughput rate and work-in-progress are optimised considering different layouts and operational policies. The second problem deals with determining values for the design and operational variables of a polymer extrusion unit. In the third problem, CO gas management at a South African heavy minerals smelter is considered. The essential processes at the smelter are simulated and an accompanying MOO problem is studied.

5.1 An inventory problem

Many real-world problems are dynamic and complex, with variation in behaviour. Operational Research practitioners often apply computer simulation to address problems of this nature. When studying such stochastic processes, small-sample theory must be applied when dealing with observations generated by a simulation model. Typically, the expected values for the objectives are estimated via point and interval estimators. It is thus required to run several pseudo-independent replications for the same input set and different random numbers in order to obtain an acceptable confidence interval for each objective. For details on simulation model output analysis, see Law & Kelton (2000:505).

Sometimes a simulation replication requires a considerable amount of time to execute. If several replications must be made, it takes even longer and becomes computationally expensive (see also Bettonvil *et al.*, 2008). If a simulation model is used to evaluate objective functions for the purpose of MOO, then several replications at various combinations of decision variable levels must be made. Attempting to find the Pareto front of a simulated process may become very hard due to the computational burden. One needs to assess whether or not the proposed MOO CEM algorithm can be used in such cases while requiring a small number of objective function evaluations.

The first case that will be examined is a simple inventory problem, which is a dynamic, stochastic problem based on a modification of the well-known (s, S) inventory problem. For a detailed description of the problem, see Bashyam & Fu (1998). The problem was adjusted to make it dynamic and stochastic, as follows: Consider a system in which a single, discrete commodity is sold to customers who arrive at the selling point according to a Poisson process, so that the interarrival times are exponentially distributed with mean $\beta = 0.5$ hours. Assume that the demand of customer i is distributed $\lfloor 20 \cdot beta(2, 1) \rfloor$ and that the order lead time is $U(1, 3)$ hours. The manager of this process will wait until the inventory is consumed below the reorder point s, and then reorder a quantity S. A lead time until delivery follows, during which customers still demand the commodity. When the inventory reaches zero and the replenishment has not arrived, a stockout period follows during which customers cannot be served. All demands during that period are considered lost sales, which must be avoided from a profit point of view.

Symbol	Description
i	Customer number at time t.
I_t	Inventory level at time t when customer c arrives.
I_0	Starting inventory (at time 0).
s	Reorder point.
S	Reorder quantity.
S_L	Service level.
S_i	Number of units that cannot be supplied to customer i.
D_i	Number of units demanded by customer i.
N_C	Total number of customers arriving in period $[0, T]$.
N_o	Number of reorders placed during period $[0, T]$.
$\overline{I}$	Average inventory level during period $[0, T]$.

Table 5.1: Notation for the (s, S) inventory problem.

On the other hand, carrying inventory incurs a cost. Holding cost is taken as ZAR10/unit/unit time, and the administration fee of a reorder is ZAR100.

The notation in Table 5.1 applies to this problem.

Customer i arrives at time t and demands D_i units of the inventory which is at level I_t. Over a fixed time period $[0, T]$, there will be N_C customers to be served while N_o reorders need to be done. The number of reorders and the average inventory $\overline{I}$ contribute to the total cost. The service level is

$$S_L = \frac{\sum_{i=1}^{N_C} D_i - \sum_{i=1}^{N_C} |S_i|}{\sum_{i=1}^{N_C} D_i} \cdot 100\% \tag{5.1}$$

and stockout follows from

$$S_i = \begin{cases} 0, & I_t \geq D_i \\ I_t - D_i, & I_t < D_i. \end{cases} \tag{5.2}$$

An infinite holding area and a reliable supplier are assumed, i.e. each time an order is placed, the correct number of units is received after the lead time has elapsed. Also, no backlog is allowed: if $D_i > I_t$ and $I_t > 0$, the customer takes I_t units and after that I_t becomes 0. If $I_t = 0$ when a customer arrives, I_t remains 0, but the stockout is adjusted according to (5.2). When the replenishment quantity arrives, I_t is adjusted according to $I_t \leftarrow I_t + S$. The typical inventory consumption and replenishment process is shown in Figure 5.1.

The decision variables in this problem are s and S, and the performance measures (objectives) are the average inventory cost over a fixed period $[0, T]$ (consisting of the inventory carry cost and the reorder cost) and the service level.

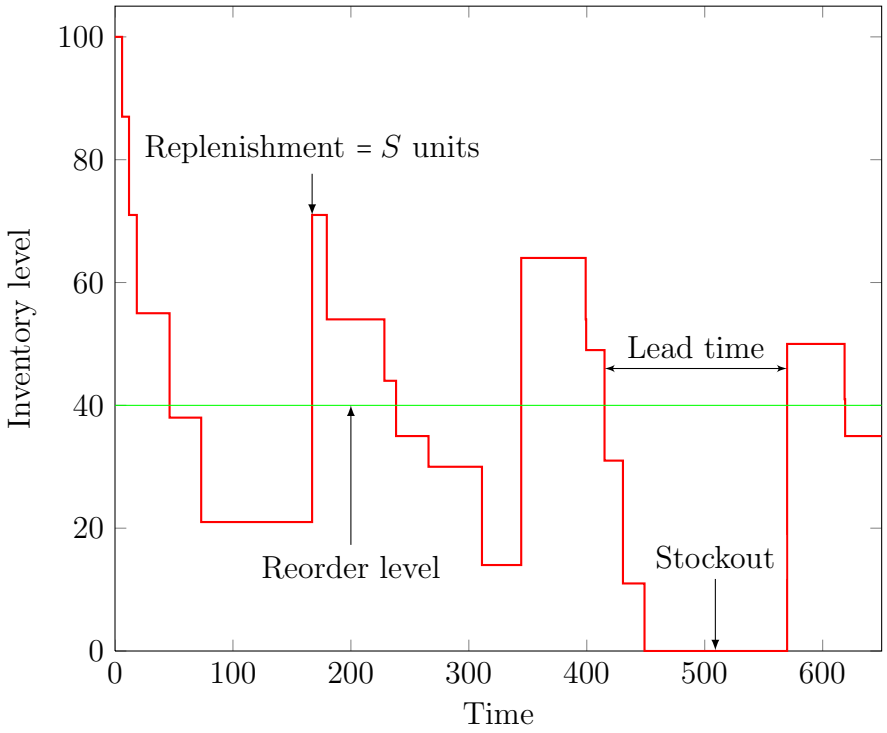

Figure 5.1: Some characteristics of the generalised (s, S) inventory process.

The MOO question is thus: for what values of s and S will the inventory cost be at a *minimum* while the service level is at *maximum*? The manager of this process will want to service all customers while carrying as little inventory as possible, and to reorder on as few occasions as possible. Note that the objective values are measured in different units.

A simulation model of this process was implemented in the simulation package Arena (Kelton *et al.*, 2007), while Algorithm 4 was implemented in VBA code in Microsoft® Excel. The algorithm samples possible values for s and S. Then the VBA code calls the Arena simulation model for evaluation of the objective functions via point estimators. Thus each combination of (s, S) in the population is evaluated via simulation and returns a combination of service level and inventory cost values. Each combination of (s, S) is evaluated using five simulation replications (observations). This may be too few when considering proper small-sample analysis, but the objective is to minimise the simulation runtime. It is also more important to obtain an approximate Pareto front showing a trend than having near-accurate estimations of the objectives. Once a front is obtained, the objectives can be evaluated more thoroughly using the values of the elite decision set. Note that the

decision variables were arbitrarily limited as follows: $0 < s \leq 500$ and $0 < S \leq 500$, giving a discrete solution space of 250 000 possibilities.

The simulated inventory system operates continuously for 2 000 minutes after a warm-up of 100 minutes and starts with an initial inventory of 100 units. To obtain a reference Pareto front, a near-exhaustive enumeration was performed with 50 simulation replications per (s, S) combination. Both s and S were adjusted in steps of five units from 5 to 500, thus exploring a solution space of 10 000 possibilities. The resulting front (containing 51 points) is considered the reference Pareto front $\mathcal{P}_T$ for this problem and is shown in Figure 5.2, together with the approximation front (41 points) obtained using the proposed MOO CEM and the same simulation model. The parameters of the MOO CEM algorithm were taken as follows: $N = 30$, $\epsilon_c = 2.5$, the probability of resetting the histograms $p_h = 0.2$ and the number of main loops was 10. The random number generator started at the same seed number for each (s, S) evaluation.

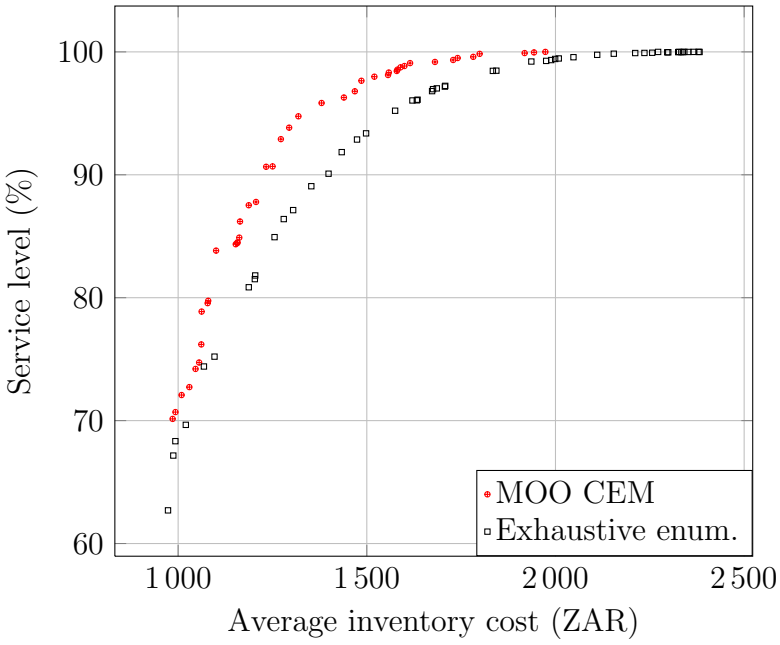

Figure 5.2: Pareto fronts for the (s, S) inventory process.

The approximation front obtained via the CEM required 1 140 evaluations (5 700 replications), which is far fewer than the 10 000 evaluations (50 000 replications) used for the near-exhaustive search. This approximation front seems to be "better" than the reference Pareto front obtained via the near-exhaustive search, but this is

due to the statistical estimation error. The shapes of the fronts are similar and the approximation front estimated via the MOO CEM algorithm will afford the decision maker the same information as will the reference Pareto front.

This concludes the study of a basic dynamic, stochastic optimisation problem using the MOO CEM algorithm in conjunction with computer simulation. In the subsequent section, a much harder problem of the same type will be studied.

5.2 The buffer allocation problem

In this section MOO of finite buffer queuing networks by means of the MOO CEM algorithm is presented. This application was chosen because it is a classical, NP-hard problem that is well researched with reference solutions available. The problem is hereafter called the *buffer allocation problem* (BAP).

5.2.1 Background on the BAP

Finite queuing networks are associated with many practical systems through which discrete or continuous flow occurs, such as manufacturing systems and telecommunication networks. These networks often exhibit flow variation or asynchronous part movement; hence the need for buffer space in the network. One of the network design priorities is to maximise the network throughput or throughput rate, which increases with more buffer space (Cruz *et al.*, 2010). However, buffer space may be costly for several reasons in commercial projects and costs must be minimised. This gives rise to the BAP which is usually formulated as a stochastic, non-linear, integer mathematical programming problem and which is computationally hard to solve (Cruz *et al.*, 2008). The problem has several variations, including problems with reliable or unreliable servers (for example, manufacturing machines), the type of distribution assigned to service time, repair times and time-to-failure, synchronous or asynchronous part movement, and the network topology. Papadopoulos & Heavey (1996) classified queuing network models for production and transfer lines, while Cruz *et al.* (2008) identified four traditional methodological approaches to the BAP: simulation methods, metaheuristics, dynamic programming and search methods. In some studies the processing times were assumed to be deterministic, while time-to-failure and repair times were exponentially distributed. Researchers also adopt different performance measures (objectives) and consider single or multi-objective problem variants, while decision variables include buffer sizes and server processing rates.

The BAP instances adopted in this study have series and general topologies with the sum of buffers constrained to a predetermined fixed number, while modifications to some of these instances present new problems, including the case where the number of buffers satisfies an inequality constraint. The problem is similar to those considered in studies by Vouros & Papadopoulos (1998) and Rubinstein & Kroese (2004), and is briefly described here: A production line consists of a series of m machines with $m-1$ buffers. Discrete parts are processed by each of these machines in sequence, and each discrete part takes up one buffer space or occupies a machine. There are n such spaces available, while the spaces in front of the first and beyond the last machine are considered infinite. There are at least two versions of the BAP, namely:

1. The sum of the allocated buffer spaces must be equal to a selected value of n, i.e. if B_i is the buffer space between machine i and $i+1$, $(1 \leq i \leq m-1)$, then $\sum_{i=1}^{m-1} B_i = n$. A buffer may be assigned zero spaces, which means that a part occupying machine i, $(1 \leq i \leq m-1)$ can only proceed to machine $i+1$ if the latter is operational and unoccupied. Under these conditions, there are thus $\binom{m+n-2}{m-2}$ possible buffer size allocations.

2. Determine the minimum number of buffers and their allocation to optimise one or more objectives.

Experiments with both these versions are presented in this section.

Generally, the machines have exponentially distributed processing and repair times with mean rates μ_i and r_i, respectively. The machine failures of the models in this study are *operation dependent failures* (ODF), which are more realistic than time-based failures (Yang *et al.*, 2000). A machine thus fails after a number of operations have been completed, and these operation counts are dictated by Poisson distributions with rates β_i. Note that the ODF approach results in longer times between failures when finite buffers are required. Suppose the time between failures for Machine 1 is exponentially distributed with rate β_1 and the repair rate is r_1. Then the expected number of failures for this machine for a simulation run length of T is $T/(1/\beta_1 + 1/r_1)$ when infinite buffers are used. When the buffer sizes are limited, this number decreases and the time between failures increases.

An upstream machine can become blocked when its successor has failed, while a downstream machine can eventually become starved if its predecessor has failed.

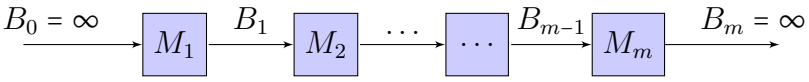

Figure 5.3: A typical series of machines $M_1, \ldots, M_m$ with finite buffers $B_1, \ldots, B_{m-1}$ in a queuing network.

The basic performance measures of such a system are the throughput rate and the *work-in-progress* (WIP). A network with a series topology is shown schematically in Figure 5.3.

Researchers have studied the BAP for several decades: Hillier & So (1991) developed an exact analytical model that shows how the lengths of machine uptimes and downtimes, as well as buffer sizes, affect line throughput. Gershwin & Schor (2000) developed efficient algorithms for buffer space allocation by formulating a primal problem, which minimises total buffer space for a required production rate, and a dual problem, which maximises the throughput rate subject to a total buffer space constraint. Their processing times are deterministic, making the algorithms appropriate for automated systems. A method for obtaining exact solutions for the throughput rate of a stochastic BAP was proposed by Heavey *et al.* (1993) and several (small) reference models were developed by Vouros & Papadopoulos (1998) using this method. Cruz *et al.* (2008) developed a buffer allocation method based on Lagrangian relaxation and applied it to queuing networks configured in various topologies, i.e. linear and non-linear.

Lutz *et al.* (1998) applied tabu search and simulation to the BAP, and used uniform distributions for processing times on failure-free machines. Their test instance has six machines and they considered up to 17 spaces for the five buffers. Dolgui *et al.* (2002) applied a GA to a BAP with exponential failure and repair rates but deterministic processing times. They maximised a cost function for the amortisation time of the production line, the revenue generated, the inventory cost due to the buffer sizes and the acquisition investment for a given line configuration. Papadopoulos & Vidalis (2001) developed a heuristic they call "PaVi", which finds good initial solutions by using information about the production line under consideration. Massim *et al.* (2010) combined an artificial immune system optimisation algorithm with a decomposition method to allocate buffers optimally. They noted that maximum line throughput does not guarantee maximum profit, and they maximised line economic profit. Shi & Gershwin (2009) maximised profit

of production lines by considering buffer cost and average inventory cost. They used a nonlinear programming approach to solve both short and long production lines.

In the context of the BAP, a multi-objective decision-making approach using quasi-concave and quasi-convex utility functions has been proposed by Malakooti (1991). The three non-commensurate objectives are 1) number of workstations (which is often assumed in other problem formulations), 2) cycle time and 3) short-term operating costs. The processing times in this case are deterministic. Cruz *et al.* (2010) studied buffer allocation and throughput trade-offs in $M|G|1|K$ queuing networks using the NSGA-II of Deb *et al.* (2002) and developed a Pareto front for a 16-node queuing network. The Pareto front is similar to the "line-specific output curve" developed by Buzacott in 1967 (Buzacott in Lu *et al.*, 2009).

The objectives of the BAP are the throughput rate $T_R(\mathbf{x})$ and WIP, denoted by $W_P(\mathbf{x})$. The values of the objectives are estimated by means of simulation models of the BAPs, as will be explained later. The problem formulation for the first version studied is

$$\text{Minimise } S_B(\mathbf{x}) := [-T_R(\mathbf{x}), W_P(\mathbf{x})] \tag{5.3}$$

$$\text{subject to } \mathbf{x} \in \mathcal{X},$$

$$\sum_{i=1}^{m-1} x_i = n, \tag{5.4}$$

where $\mathcal{X}$ is the set of all possible valid combinations of $\mathbf{x}$.

The proposed algorithm for MOO using the MOO CEM algorithm is stated as Algorithm 7, and is applied in this section to instances of the BAP using discrete-event simulation for evaluation of decision sets. In these problems, the decision maker specifies n niches, and requires that they all be used. However, the WIP is still minimised to show the decision maker the behaviour of the problem, facilitating the possibility that he/she will decide to trade throughput rate for fewer WIP. In Section 5.2.4, the equality constraint (5.4) is changed and the WIP for "large" buffers is estimated.

Convergence is tested for using the standard deviations of the objective function values in the Elite set. As soon as consecutive values of each of the standard deviations do not differ by more than a preset deviation ϵ_c, say $\epsilon_c = 10^{-2}$, the algorithm terminates. The decision maker therefore has to specify the population

Algorithm 7 MOO CEM multi-objective buffer allocation algorithm

1: Initialise $P_0 = 1/(n+1)$, $t = 0$ and elite set **Elite** = $\varnothing$.
2: **repeat**
3: Generate a population of N solutions $\mathbf{x}_1, \ldots, \mathbf{x}_N$ using P_t.
4: Evaluate each vector $\mathbf{x}_k$ in the population using discrete-event simulation and return estimations $\hat{T}_R(\mathbf{x}_k)$ and $\hat{W}_P(\mathbf{x}_k)$, $1 \leq k \leq N$.
5: Append **Elite** to the current population and use Algorithm 3 to rank the new set.
6: Set **Elite** = $\varnothing$ and move members in the population having rank 0, to **Elite**.
7: Develop the elements $\hat{p}_j$ of P_{t+1} using (3.23):
$\hat{p}_j \leftarrow \frac{\sum_{i=1}^{N} I_{\{\hat{\Upsilon}(\mathbf{x}_i) \geq \gamma\}} I_{\{X_{ij}=j\}}}{\sum_{i=1}^{N} I_{\{\hat{\Upsilon}(\mathbf{x}_i) \geq \gamma\}}}$, $j = 0, \ldots, n$.
8: Update P_{t+1} by means of (3.24): $\hat{P}_t \leftarrow \alpha \tilde{P}_t + (1-\alpha)\hat{P}_{t-1}$.
9: Increment t.
10: **until** P_t has converged.
11: Return **Elite**.

size N, the total buffer size n for a given number of machines m, the smoothing parameter α, and the termination value of ϵ_c.

During algorithm execution, previous candidate solutions may be selected again. To save simulation time, an archive of previously evaluated candidates and their solutions are maintained. Before a candidate solution is evaluated by means of the simulation model, it was first tested for membership of the archive. If it is a member, its evaluation is retrieved from the archive, otherwise the simulation model is called for evaluation, and when completed, the result is added to the archive. The archive search requires additional storage space and since a simple sequential search of the non-ordered archive list is used, the complexity is $O(N_a)$ and the expected number of searches is $N_a/2$, where N_a is the archive size. Initially the archive is empty, but it grows with the algorithm iterations. The maximum archive size observed was less than 5 000 solutions.

The ranking in Algorithm 7 is based on the Pareto ranking algorithm of Goldberg (1989) and was presented in Section 4.1 as Algorithm 3.

Next, the validation of the simulation-optimisation model for the BAP is discussed.

n	Optimum buffer allocation	Model buffer allocation	$\overline{\lambda}_n$	λ^*
1	(0,1,0,0)	(0,1,0,0)	0.5222	0.5213
2	(1,1,0,0)	(1,1,0,0)	0.5520	0.5514
3	(1,1,1,0)	(1,1,1,0)	0.5858	0.5824
4	(1,2,1,0)	(1,2,1,0)	0.6063	0.6027
5	(2,2,1,0)	(2,2,1,0)	0.6243	0.6213
6	(2,2,1,1)	(2,2,1,1)	0.6483	0.6422
7	(2,2,2,1)	(2,2,2,1)	0.6663	0.6585
8	(3,2,2,1)	(3,2,2,1)	0.6822	0.6744
9	(3,3,2,1)	(3,3,2,1)	0.6975	0.6894
10	(3,3,3,1)	(3,3,3,1)	0.7097	0.7005

Table 5.2: Simulation validation values for reference instances (Set 1).

5.2.2 BAP: Simulation-optimisation model validation

The discrete-event simulation models of the queuing networks were implemented in the Arena simulation package (Kelton *et al.*, 2007) while the optimisation algorithm was coded in Matlab® R2007b. These two components were integrated to form the simulation-optimisation model and applied to known problems in the literature.

The model was validated by optimising the single objective (throughput rate) of known instances studied by Vouros & Papadopoulos (1998) and later by Dolgui *et al.* (2002) and Rubinstein & Kroese (2004). Two cases of network configurations are presented (Set 1 and Set 2).

The first case (Set 1) follows from Rubinstein & Kroese (2004) for $m = 5$ machines and various values of n. The throughput rate estimation $\overline{\lambda}_n$ for the case $m = 5$, $n = 1, \ldots, 10$, exponentially distributed processing times with rates $\mu_1 = 1$, $\mu_2 = 1.1$, $\mu_3 = 1.2$, $\mu_4 = 1.3$, $\mu_5 = 1.4$, failure rates $\beta_i = 1/19$ and repair rates $r_i = 0.5$ are shown in Table 5.2, where λ^* is the exact throughput rate determined by Vouros & Papadopoulos (1998). The simulated estimations were obtained via the replication-deletion output analysis approach (Law & Kelton, 2000). This was achieved by executing 250 pseudo-independent replications, each 11 000 simulation time units long, with the first 1 000 simulation time units as warm-up, and calculating the mean of the throughput rate as well as its 95% confidence interval.

The second set of validation values (Set 2) is shown in Table 5.3 for $m = 5$ and $n = 1, \ldots, 10$, Erlang$_2$-distributed processing times with rates $\mu_1 = 1$, $\mu_2 = 1.1$,

n	Optimum buffer allocation	Model buffer allocation	$\overline{\lambda}_n$	λ^*
1	(0,1,0,0)	(0,1,0,0)	0.5845	0.5968
2	(1,1,0,0)	(1,1,0,0)	0.6186	0.6338
3	(1,1,1,0)	(1,1,1,0)	0.6539	0.6673
4	(2,1,1,0)	(2,1,1,0)	0.6735	0.6808
5	(2,2,1,0)	(2,2,1,0)	0.6940	0.6996
6	(2,2,1,1)	(2,2,1,1)	0.7181	0.7195
7	(2,2,2,1)	(2,2,2,1)	0.7341	0.7341
8	(3,2,2,1)	(3,2,2,1)	0.7499	0.7501
9	(3,3,2,1)	(3,3,2,1)	0.7648	0.7620
10	(4,3,2,1)	(4,3,2,1)	0.7755	0.7740

Table 5.3: Simulation validation values for reference instances (Set 2).

$\mu_3 = 1.2$, $\mu_4 = 1.3$, $\mu_5 = 1.4$, failure rates $\beta_i = 1/19$ and repair rates $r_i = 0.5$. Again, the λ^* are the exact values from Vouros & Papadopoulos (1998).

Having validated the simulation-optimisation model using the results of known single-objective instances, the next step was to consider the same problems (Set 1 and Set 2), but with two optimisation objectives: minimising WIP while maximising the throughput rate. A third problem set (Set 3) with $m = 10$ is added, and finally, a more complex network with $m = 16$ is considered (Set 4). The results of the BAP analysis are presented in the next section.

5.2.3 Finding buffer allocations with an equality constraint

The experimental results obtained are presented in three subsections: in the first the output characteristics of the MOO CEM for each BAP are shown (Subsection 5.2.3.1). The number of evaluations and the size of the Elite set compared to the known Pareto set are of importance — ideally, the Pareto set should be replicated in the Elite set. Secondly, to quantify the performance quality of the MOO CEM algorithm when searching for Pareto sets in BAPs, numerical values for some MOO algorithm quality indicators are presented (Subsection 5.2.3.2). Lastly, the buffer size allocation trends are presented and discussed (Subsection 5.2.3.3).

Note that the allowed buffer size per problem, i.e. the value of n, was arbitrarily chosen, but the values do cover at least three categories: a "few" niches (10) to "many" niches (40–60). The cross-entropy smoothing parameter was taken as $\alpha = 0.6$ while the population size was chosen arbitrarily (10–30), and was increased

		Exponential			Erlang$_2$		
Buffer size	Problem size	Number of evaluations[1]	Size Elite	Size PF	Number of evaluations[1]	Size Elite	Size PF
$n = 10$	286	192	18	31	255	23	33
$n = 20$	1 771	210	22	42	575	32	42
$n = 40$	12 341	625	57	70	625	57	70

[1] To obtain Elite set.

Table 5.4: Simulated MOO CEM results for some reference instances (Set 1 and 2).

as the problem size increased. The simulation-optimisation model was executed on an IBM Lenovo laptop with an Intel Core i5 CPU running at 2.40GHz.

5.2.3.1 The BAP with equality constraint: results of approximation sets found

The values of algorithm performance characteristics for various configurations of the test instances are shown in Table 5.4, Table 5.5 and Table 5.6, while the Pareto set and Elite set for $m = 5$ machines and $n = 10$ niches, exponential processing times are shown graphically in Figure 5.4. Similar plots for the remainder of the BAP instances are shown in Appendix A for exponential and Erlang$_2$-distributed processing times. The smallest instance in Set 1 and Set 2 was enumerated, its Pareto front being a *true* front. The remaining instances have too many solutions for enumeration. It is therefore only possible to construct *approximate* Pareto fronts, which was done via five independent runs of the MOO CEM algorithm to build archives of solutions. The *Elite*-column in the tables contains the estimated Pareto front determined by another independent run of the algorithm. In the cases where enumeration was not possible, the archived solutions are included on the graphs and labelled "Subset solutions."

The third set of test instances (Set 3) used $m = 10$ machines and various values of n based on Rubinstein & Kroese (2004). The results are shown in Table 5.5. The processing times are exponential with rates $\mu_1 = 8$, $\mu_2 = 8$, $\mu_3 = 11$, $\mu_4 = 14$, $\mu_5 = 14$, $\mu_6 = 11$, $\mu_7 = 8$, $\mu_8 = 8$, $\mu_9 = 6$, $\mu_{10} = 6$. The failure rates are $\beta_i = 1/19$ and the repair rates are $r_i = 0.5$.

The final network considered (shown in Figure 5.5) has 16 nodes (machines) with branches and was proposed by Smith & Cruz (2005).

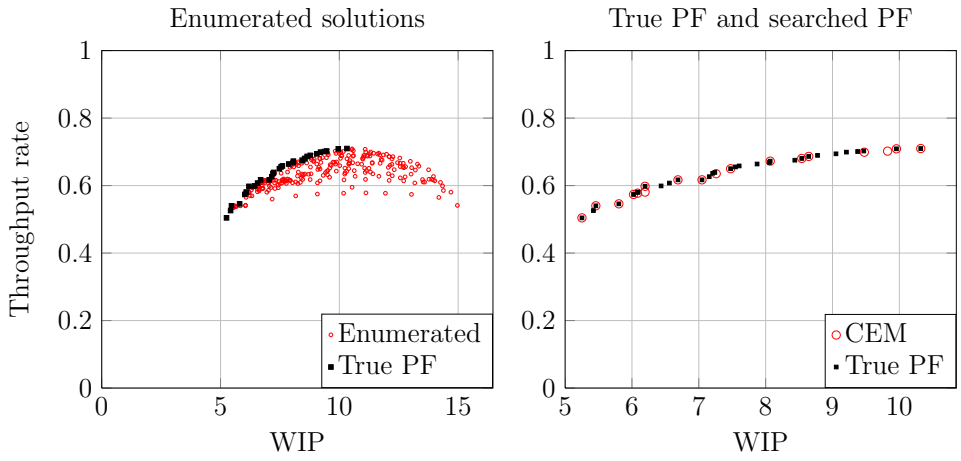

Figure 5.4: Graphic results for $m = 5$ machines and $n = 10$ niches, exponential processing times.

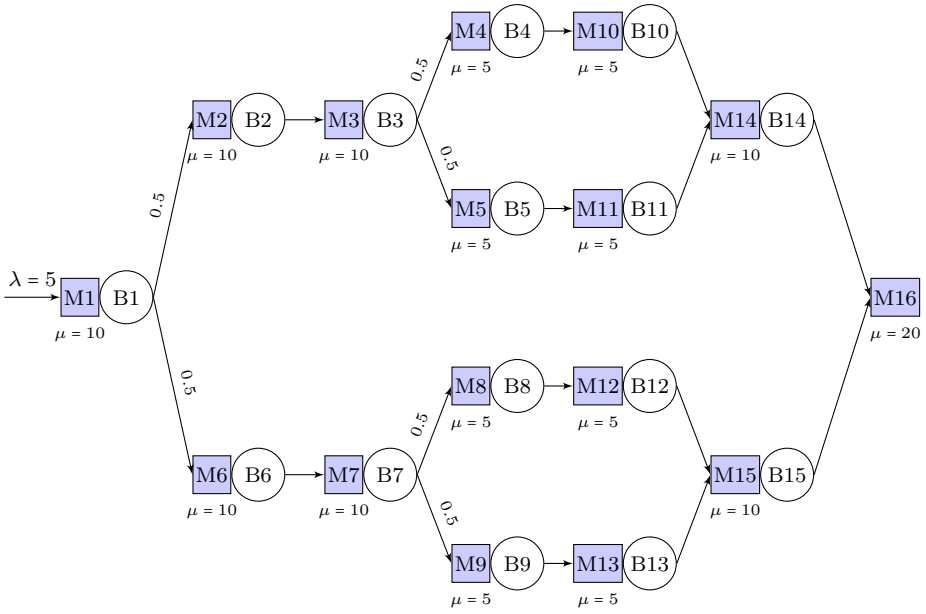

Figure 5.5: Sixteen-node network (Smith & Cruz, 2005).

Buffer size	Instance size	Number of evaluations[1]	Size Elite	Size PF
$n = 10$	43 758	210	18	25
$n = 20$	$> 3 \times 10^6$	690	18	18
$n = 30$	$> 48 \times 10^6$	450	22	25
$n = 40$	$> 377 \times 10^6$	750	15	21

[1] To obtain Elite set.

Table 5.5: Simulated MOO CEM results for instances with $m = 10$ and exponential processing times (Set 3).

Buffer size	Instance size	Number of evaluations[1]	Size Elite	Size PF
$n = 10$	$> 1.9 \times 10^6$	350	23	25
$n = 20$	$> 1.3 \times 10^9$	400	20	38
$n = 30$	$> 114 \times 10^9$	780	39	40
$n = 40$	$> 3 \times 10^{12}$	1 200	40	40
$n = 50$	$> 47 \times 10^{12}$	625	38	43
$n = 60$	$> 456 \times 10^{12}$	480	30	30

[1] To obtain Elite set.

Table 5.6: Simulated MOO CEM results for the 16-node instance.

The optimisation results for various buffer allocations for this instance are shown in Table 5.6 (Set 4).

Plots of approximate fronts for this instance are included in Appendix A. There is no proof that these approximate Pareto sets contain the complete sets of true solutions, but the graphic shapes of the fronts found in the experiments make sense and are in line with the findings of others, for example Cruz *et al.* (2010).

Furthermore, the number of evaluations required are fewer than the numbers reported by other researchers. Cruz *et al.* (2010) applied the well-known NSGA-II of Deb *et al.* (2002) to the BAP and used a population size of 80 with the maximum number of generations equal to 1 000, thus allowing up to 80 000 evaluations. For the 16-node instance (Set 4), their minimum average number of generations in the various experiments was approximately 250, which required 2 000 evaluations. It is claimed that the Pareto fronts of multi-objective problems that have many solutions can be estimated via fewer evaluations when using the MOO CEM algorithm with the BAP.

Buffer allocation multi-objective instance	Short name	GD	SP	ME	CV
Exponential, $m = 5, n = 10$	BAP1	0.0363	0.1921	0.4081	0.1956
$n = 20$	BAP2	0.0654	0.3174	0.5880	0.2737
$n = 40$	BAP3	0.0282	0.2713	0.9040	0.1010
Erlang$_2$, $m = 5, n = 10$	BAP4	0.0411	0.1567	0.5316	0.0172
$n = 20$	BAP5	0.0211	0.2214	0.3122	0.0858
$n = 40$	BAP6	0.1828	0.2505	2.9114	0.0868
Exponential, $m = 10, n = 10$	BAP7	0.0302	0.1590	0.3522	0.1046
$n = 20$	BAP8	0.0000	0.2382	0.0000	0.0000
$n = 30$	BAP9	0.0649	0.3392	0.9121	0.0458
$n = 40$	BAP10	0.3395	0.9137	3.4852	0.0296
Sixteen nodes, $n = 10$	BAP11	0.0463	0.2155	0.4377	0.0734
$n = 20$	BAP12	0.1072	0.7367	1.5846	0.1074
$n = 30$	BAP13	0.0375	0.3530	1.0033	0.0157
$n = 40$	BAP14	0.0000	0.7401	0.0017	0.0005
$n = 50$	BAP15	0.0345	0.4345	0.7302	0.0128
$n = 60$	BAP16	0.0013	1.4934	0.0402	0.0005

Table 5.7: Values for quality indicators of the MOO CEM algorithm applied to instances of the BAP.

Next, numerical values for some quality indicators of the solutions to the BAP instances are presented.

5.2.3.2 The BAP with equality constraint: results of approximation set quality indicators

The same quality indicators introduced in Subsection 2.3.4 are used to quantify the performance of the algorithm, namely *Generation distance* (GD), *Spacing* (SP), *Maximum Pareto front error* (ME), and *Convergence* (CV). The experimental values obtained are shown in Table 5.7.

The GD indicator requires the true Pareto front for comparison. This is impractical to obtain when large solution spaces prevail; therefore the solution archives mentioned before were ranked and it was assumed that the approximate fronts following from the rankings were the true fronts. Good indicator values approach zero, and the SP indicator operates on the known Pareto set only.

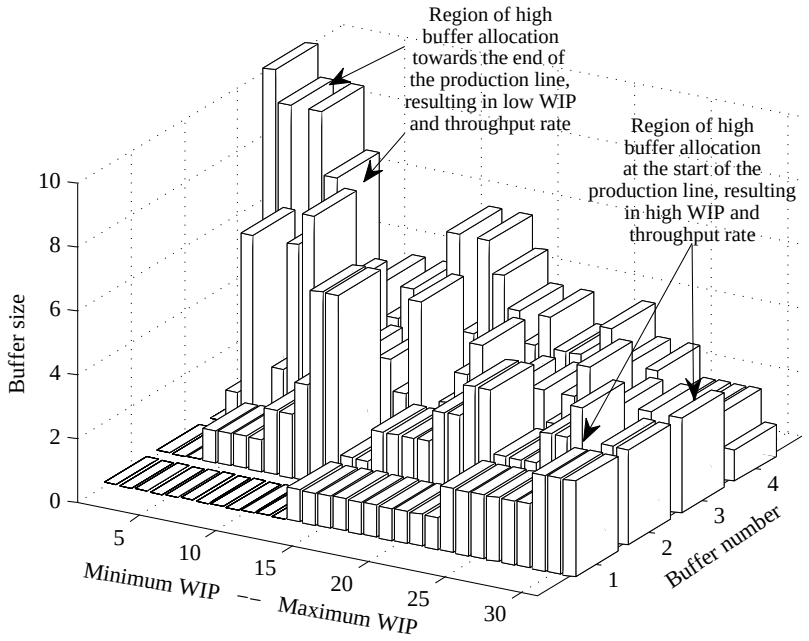

Figure 5.6: Buffer allocations for $m = 5$, $n = 10$, exponential processing times.

5.2.3.3 The BAP with equality constraint: Trends in buffer size allocation

The trends in buffer sizes allocated by the MOO CEM algorithm may be investigated to determine whether or not the results make sense. To do so, the known Pareto front sets were sorted in ascending order with the WIP as sort key. The trends observed in the case of the serial topologies are:

1. For *low* values of WIP (and necessarily low values of the throughput rate), the buffers downstream in the line receive *more* space.

2. For *high* values of throughput rate and consequently high values of WIP, the buffers upstream receive *more* allocations with progressively *less* space downstream.

Examples of these trends are shown in Figures 5.6 – 5.11. Note that the size of the approximate Pareto set shown in each figure differs (see Table 5.4 and Table 5.5), and each set member is numbered along the axis labelled "Minimum WIP – Maximum WIP." The set members representing the lowest WIP have the lowest numbers.

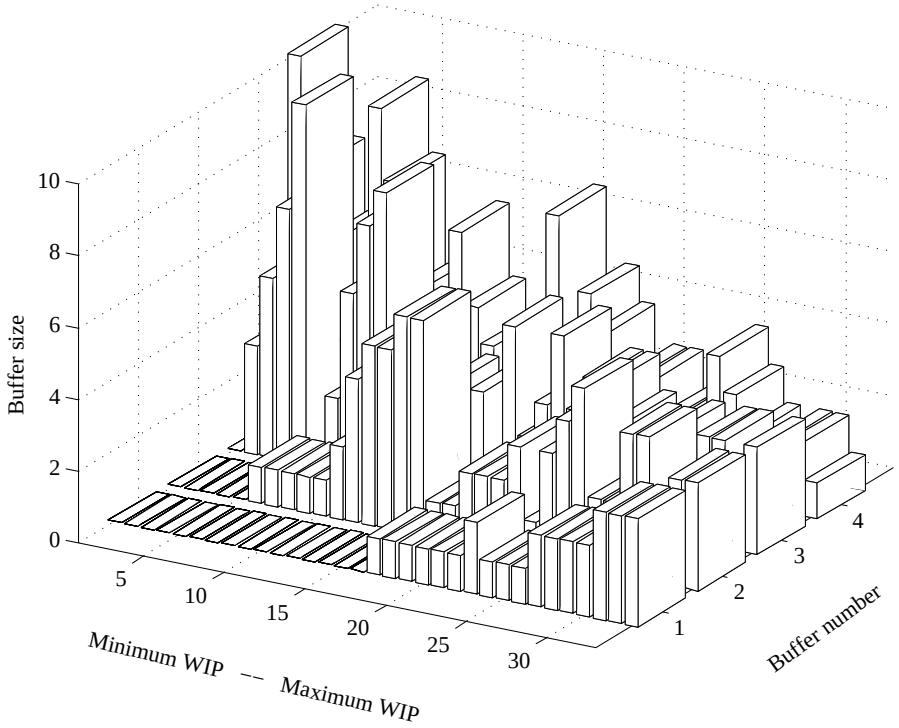

Figure 5.7: Buffer allocations for $m = 5$, $n = 10$, Erlang_2 processing times.

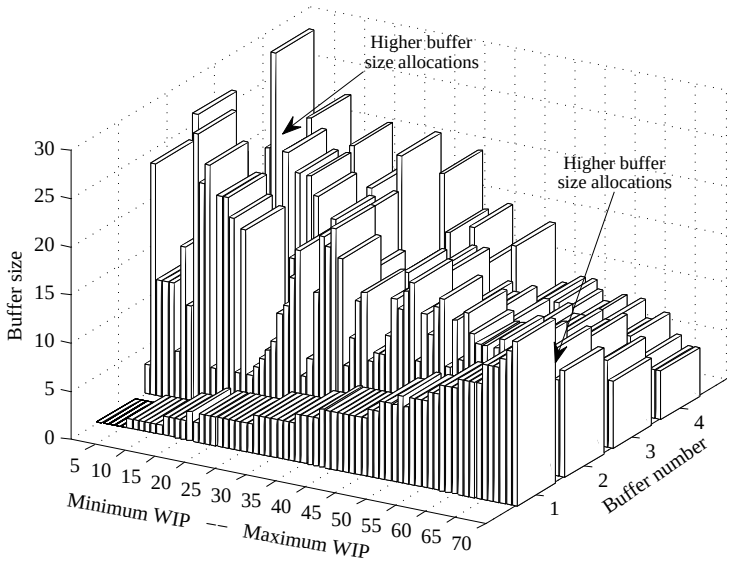

Figure 5.8: Buffer allocations for $m = 5$, $n = 40$, exponential processing times.

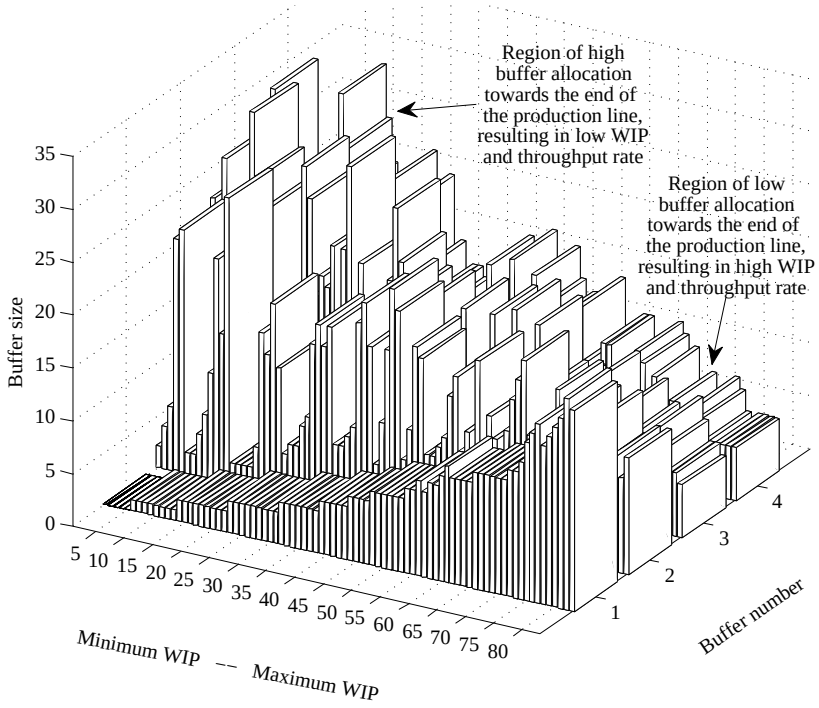

Figure 5.9: Buffer allocations for $m = 5$, $n = 40$, Erlang$_2$ processing times.

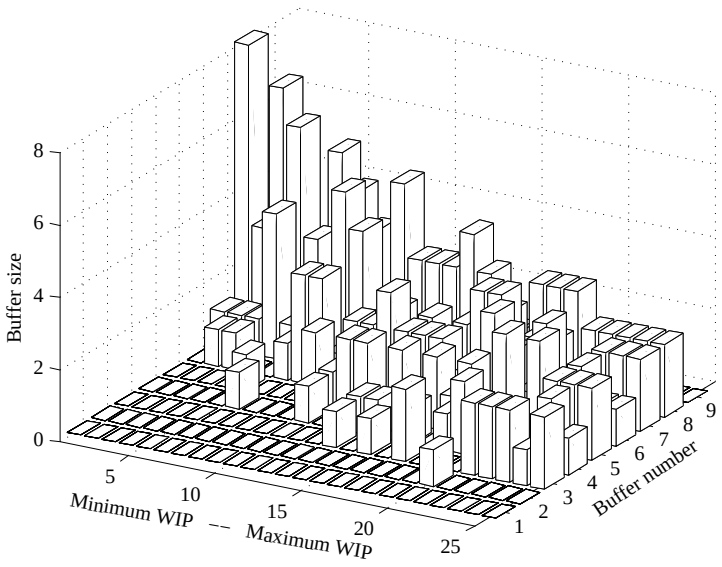

Figure 5.10: Buffer allocations for $m = 10$, $n = 10$, exponential processing times.

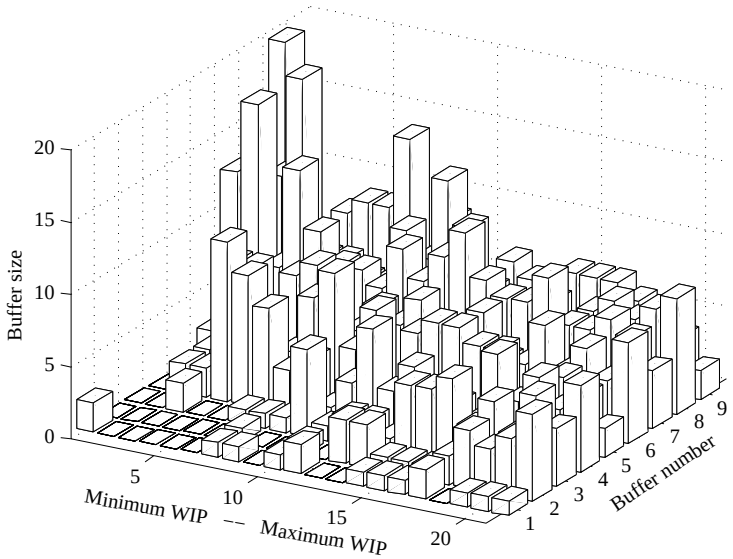

Figure 5.11: Buffer allocations for $m = 10$, $n = 40$, exponential processing times.

The trends are explained as follows (see also Figure 5.12): to minimise WIP, the available buffer space is allocated to the faster machines downstream where WIP is processed faster than upstream. In doing so, the algorithm avoids large WIP being stored in the system where slower machines operate. Of course, the throughput rate decreases. On the other hand, to ensure a high throughput rate, the available buffer spaces are allocated mostly to the slower machines upstream, which ensures that these machines do not starve the faster, downstream machines. The last machine in the sequence consistently received the least number of buffer spaces because it cannot affect the downstream process. In the context of MOO, these trends make sense and the observed MOO CEM results are encouraging.

In the case of the 16-node non-serial topology studied, buffer sizes associated with the known Pareto fronts of the six scenarios of that instance were used. The average buffer size allocation was calculated and the resulting values are shown in Table 5.8 (the buffer numbers refer to those in Figure 5.5). The MOO CEM algorithm assigned more or less the same buffer sizes to machines that have the same processing rate (with the exception of $n = 60$, at $B3$). It may therefore be concluded that the algorithm is consistent.

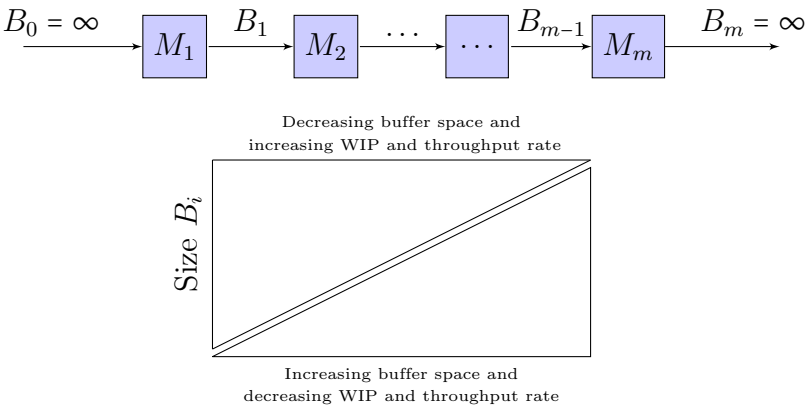

Figure 5.12: Throughput and WIP requirements in a serial processing line.

	Buffer number														
n	1	2	6	3	7	4	5	8	9	10	11	12	13	14	15
10	1.0	0.4	0.2	1.4	1.2	0.5	0.5	0.8	0.6	0.4	0.5	0.4	0.1	1.1	1.0
20	1.2	0.3	0.7	2.1	2.2	0.8	1.5	1.4	1.3	1.0	1.4	0.4	1.3	3.2	1.2
30	1.3	1.2	0.7	3.6	3.5	2.7	2.2	1.7	2.2	1.7	1.6	1.7	1.1	3.6	1.5
40	2.6	1.3	2.0	5.5	4.2	1.6	2.2	2.8	2.4	2.2	1.6	2.9	1.3	4.9	2.8
50	1.6	2.5	2.3	5.2	5.0	3.5	3.3	4.3	3.7	2.4	3.3	2.6	1.6	3.9	4.9
60	2.5	2.1	3.4	9.5	5.8	4.8	4.8	4.5	6.5	2.7	2.6	2.7	1.3	4.1	2.7

Table 5.8: Average buffer size allocation for the 16-node non-serial topology and various values of n.

5.2.4 Finding buffer allocations with an inequality constraint

In the introduction to this section (5.2, p. 87) the BAP was formulated with an equality constraint in (5.4), and it was subsequently shown that maximum throughput rates can be achieved if the exact number of n buffer spaces are acquired by the decision maker, but less spaces can be considered if the decision maker is willing to accept a lower throughput rate.

In this section, the problem formulation is changed, and the decision maker now states a willingness to pay for n_i or fewer niches per buffer. The n_i-values need not be equal. The purpose of this experiment was to show that the MOO CEM algorithm can be applied to a problem with inequality constraints. The problem

n_i	Mean WIP	Mean max. WIP	Mean T_R	Number of evaluations[1]	Size Elite
5	8.60	19.80	0.69	460	30
10	11.49	29.19	0.76	300	35
20	14.93	39.81	0.82	660	40
30	15.63	41.21	0.83	580	46
40	17.10	47.37	0.85	1 500	50

[1] To obtain Elite set.

Table 5.9: Buffer allocation estimations for an inequality constraint.

formulation now becomes

$$\text{Minimise } S_B(\mathbf{x}) := [-T_R(\mathbf{x}), W_P(\mathbf{x})] \tag{5.5}$$

$$\text{subject to } \mathbf{x} \in \mathcal{X}',$$

$$x_i \leq n_i,\ i = 1, \ldots, m-1. \tag{5.6}$$

The n_i-values may be zero, so the problem size is $\Pi_{i=1}^{m-1}(n_i+1)$, and the maximum total number of niches that can be allocated is $\sum_{i=1}^{m-1} n_i$.

The results of two experiments are reported in this subsection. In the first, the expected WIP and T_R were determined via the MOO CEM algorithm, and in the second, a different formulation of the WIP response was investigated.

5.2.4.1 Experimenting with the BAP WIP under the inequality constraint

The BAP with exponential processing times and $m = 5$ for various values of n_i was used as test instance for this experiment. Also, the *expected* maximum WIP was estimated along with the expected WIP. These WIP values are, as before, the estimation of the *physical* number of product units in the system, while n_i is the allowable space in buffer i. The results are shown in Table 5.9, where the mean values are those of the values in the elite sets.

The approximate Pareto front values for $m = 5$ and various values of n_i are shown in Figure 5.13, together with the *accompanying* estimated expected maximum values for the WIP. The values of n_i were the same for all buffers of a given instance. The maximum WIP values thus do not form fronts, but they show the maximum buffer space required to achieve a desired throughput rate. The label "A" points to the lowest average WIP, while "B" points to the maximum WIP observed when

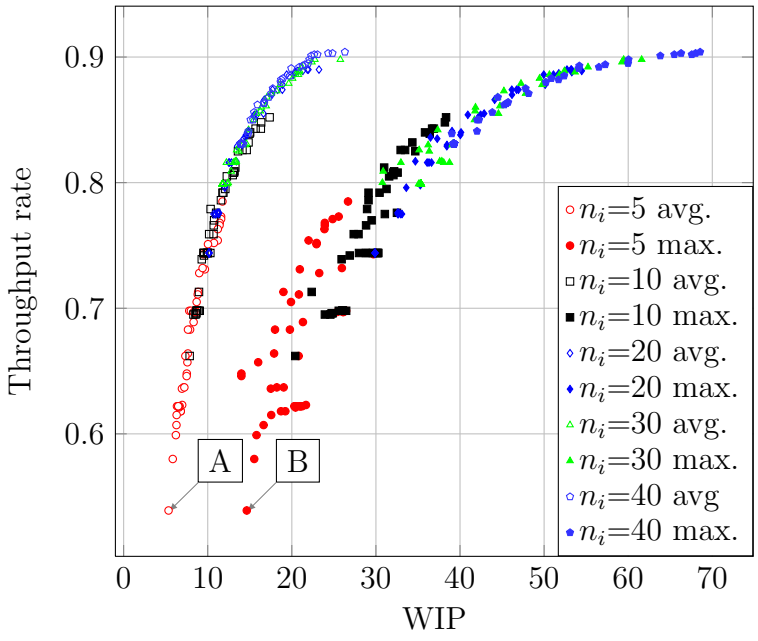

Figure 5.13: Approximate fronts for $m = 5$ and various values of n_i, with the maximum estimated WIP.

the point at "A" was generated during the optimisation. Also note that the group of symbols on the left (representing the approximation fronts combined) shows smaller variation than the maximum values, and that the throughput rate increases with increasing n_i. For $n_i = 5$, the throughput rate is between 0.539 and 0.785, for $n_i = 10$ it is higher (0.621 to 0.852), for $n_i = 20$ it ranges between 0.622 to 0.890, for $n_i = 30$ the range is 0.622 to 0.898, and for $n_i = 40$ it is 0.622 to 0.904.

The WIP values in Figure 5.13 indicate that the decision maker should consider values of $n_i \leq 20$ per buffer, since the maximum required buffer space is less than 70. The sum of the four buffers in this problem ($m = 5$) for $n_i = 20$ is 80. For $n_i = 30$, or 120 buffer spaces, the expected maximum buffer space required was found to be 61.6, and for $n_i = 40$ (160 buffer spaces) it was 68.55. The top ten estimated maximum values of throughput rate per n_i were taken from Figure 5.13 and are shown in Figure 5.14.

It can be seen that the required buffer space levels out towards the maximum throughput rate of 0.9, but at $n_i = 20$, a point of diminishing return is reached. Further analysis should thus be limited to $n_i \leq 20$, as mentioned before.

The findings above prompted a desire for further investigation, namely to minimise the expected *maximum* WIP, instead of the expected WIP. The resulting

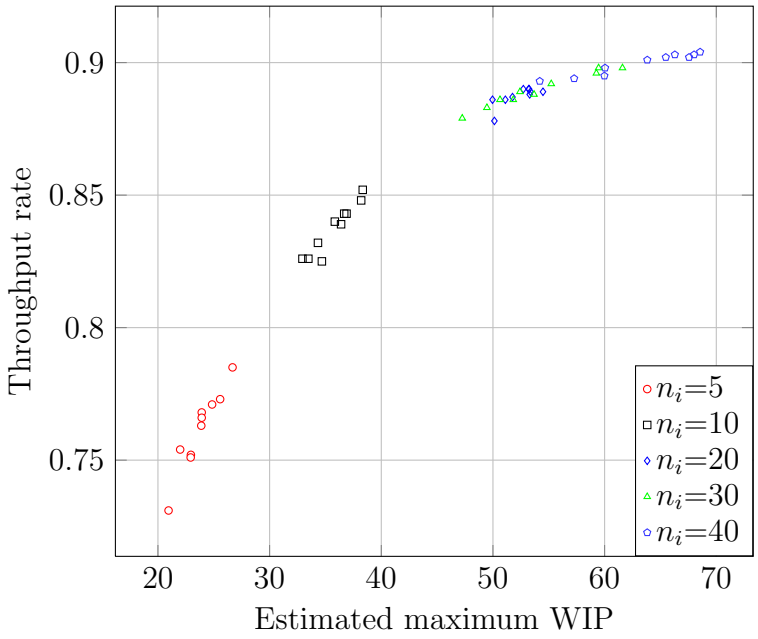

Figure 5.14: Estimated maximum physical buffer space required for $m = 5$ and various n_i (derived).

approximate fronts for $n_i = 5, 10, 20$ and 30 are shown in Figure 5.15.

The point of diminishing return is reached at approximately 60 units. The approximate Pareto sets for $n_i = 20$ and $n_i = 30$ were analysed and the solutions with throughput rate greater than 0.88 were grouped together and then sorted with the sum of the allocated buffer spaces as sort key. The values are shown in Table 5.10.

From these, the decision maker may select a possible implementation, or a group of possibilities for further analysis. It is also clear that investigation of values for n_i beyond 30 is unnecessary in this instance.

The results obtained so far inspired further experimentation: from Figure 5.13 and Figure 5.14 it follows that the observed maximum WIP tends to become significantly larger with larger buffer sizes than the expected WIP. Also, in Table 5.10, the sum of the buffer allocations shows a variation range of 52%, while the estimated maximum WIP shows a variation range of 24% and the T_R only 2%. This means that, if the decision maker wants to be more realistic, and chooses to have the expected maximum WIP as objective instead of the expected WIP, there is still the risk of acquiring too many buffers. The question arises: will it

n_i	B_1	B_2	B_3	B_4	$\sum B_i$	Estimated maximum	T_R
20	16	16	8	8	48	47.70	0.880
20	20	11	9	11	51	48.04	0.883
20	20	11	11	13	55	48.45	0.884
20	19	16	8	13	56	50.76	0.885
20	18	16	13	10	57	49.64	0.884
20	20	18	10	15	63	52.28	0.888
30	27	17	11	13	68	56.83	0.895
30	22	14	11	22	69	51.94	0.889
30	20	14	22	15	71	51.18	0.887
30	30	22	9	11	72	62.77	0.898
20	19	20	18	16	73	52.50	0.889
20	18	19	18	18	73	51.01	0.887
30	22	15	22	15	74	53.36	0.891
30	22	14	27	12	75	52.95	0.891
30	26	14	18	18	76	55.43	0.894
30	27	18	14	18	77	57.79	0.896
30	23	15	15	24	77	53.73	0.892
30	22	18	18	22	80	54.07	0.892
30	21	15	19	26	81	52.44	0.890
30	29	22	9	22	82	61.87	0.898
30	22	15	27	18	82	53.39	0.891
30	30	14	19	20	83	59.28	0.897
30	27	11	19	26	83	55.13	0.893
30	22	15	19	27	83	52.47	0.890
30	27	28	15	14	84	59.52	0.898
30	26	17	15	27	85	56.98	0.896
30	26	14	19	28	87	55.66	0.894
30	26	24	23	18	91	59.32	0.897
30	28	15	26	22	91	57.95	0.897
30	24	26	26	15	91	57.95	0.896
30	22	22	23	24	91	55.27	0.893
30	27	30	15	28	100	60.79	0.898

Table 5.10: Solutions for highest throughput rates and sums of buffer allocations.

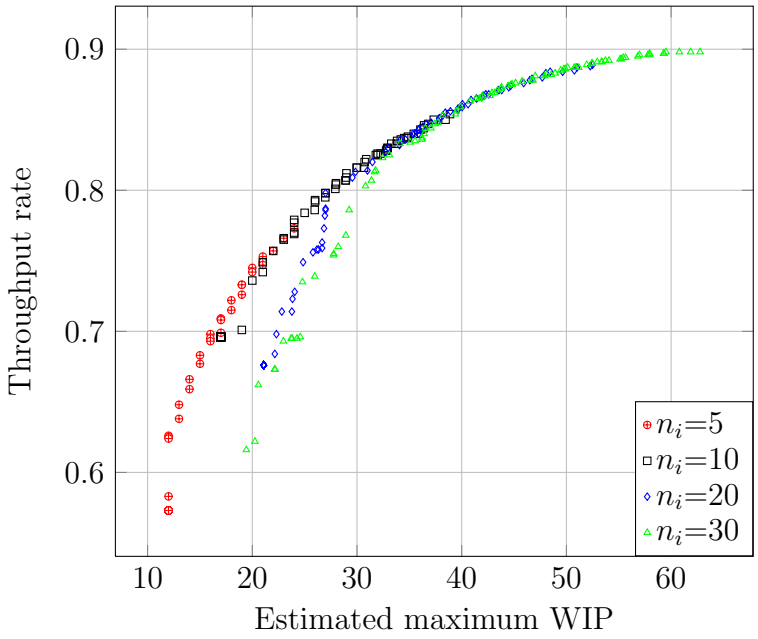

Figure 5.15: Estimated maximum physical buffer space required for $m = 5$ and various values of n_i (complete optimisation).

be worthwhile to use the expected maximum WIP as an objective, instead of the estimated mean WIP? This question is answered in the next section.

5.2.4.2 A new objective for the BAP

Researchers usually attempt to minimise the total number of buffer spaces while maximising the throughput rate $T_R(\mathbf{x})$ of a given queuing network. In the experiment described in this subsection, a modified, more practical second objective is proposed, based on the findings in Subsection 5.2.4.1. The principle is to measure the actual system occupation due to *observed* WIP as it persists over *time*, instead of just the maximum WIP level. Since the WIP varies over time, the time duration of the WIP at each level $(0, 1, 2, \ldots)$ is recorded, and after a finite time T, the p_q-th percentile of these time-based levels is observed. Let T_j denote the total time that the WIP was at level $W_j = j$, $(j = 0, 1, 2, \ldots, \sum_{i=1}^{m-1} n_j)$ during the simulation run, and let n_{p_q} be the WIP level of the p_q-th time percentile. W_M Then n_{p_q} is

the solution of the problem of

$$\begin{aligned} &\text{Minimising } n_{p_q} \\ \text{subject to } &\frac{\sum_{j=0}^{n_{p_q}} T_j \times W_j}{\sum_{j=0}^{W_M} T_j \times W_j} \geq p_q, \qquad (5.7) \\ &\sum_{j=0}^{W_M} (T_j \times W_j) > 0. \qquad (5.8) \end{aligned}$$

In (5.7), W_M is the maximum WIP level observed during T, $0 \leq W_M \leq \sum_{i=1}^{m-1} n_i$. If (5.8) is violated, i.e. all the W_j are zero, then $n_{p_q} = 0$. The reason for using a *time-based* percentile is to consider the *intensity* of each WIP level. If only the observed values W_i of the WIP level are used, a few high values may occur for a short time during the simulation run, resulting in more buffer spaces being allocated by the algorithm. These spaces will then hardly be used once the real system is implemented, resulting in wasted space. The time-based percentile rules out buffer size extremities which exist for short time periods, or will include them if they are significant. This objective is again denoted by $W_P(\mathbf{x})$ and must be minimised. Note that, since a percentile is used, the system will not be able to accommodate WIP for $1 - p_q$ percent of the time, and the throughput rate will be reduced.

To further explain the proposed objective, a simple but typical time-persistent graph of WIP is shown in Figure 5.16. The shaded area is the denominator in (5.7) and is equal to 38.75 in this case, with the time-average of 2.583 units. The plot shows a spike that reaches seven units over time period [4; 4.25], and so W_M=7 in (5.7). The WIP levels and their respective areas are shown in Table 5.11 with the cumulative proportional contribution of each WIP level. Suppose one is interested in the 95-th percentile of the WIP intensity ($p_q = 0.95$), then this value is exceeded for the first time at WIP level = 4 (see the column with heading "4" in Table 5.11), or $n_{p_q} = 4$ in (5.7). In this case, the decision maker will be advised to acquire four units of buffer space, although as many as eight units of WIP were observed. Note that this maximum cannot exceed the sum of the allowed maximum buffer sizes n_i.

The problem formulation is the same as in (5.5) and the values of the objectives are again estimated by means of simulation models of the BAP instances. Since the number of buffer spaces per buffer (niche) is unbounded, the search must be contained. Also, the probability matrix P of the CEM requires a finite number

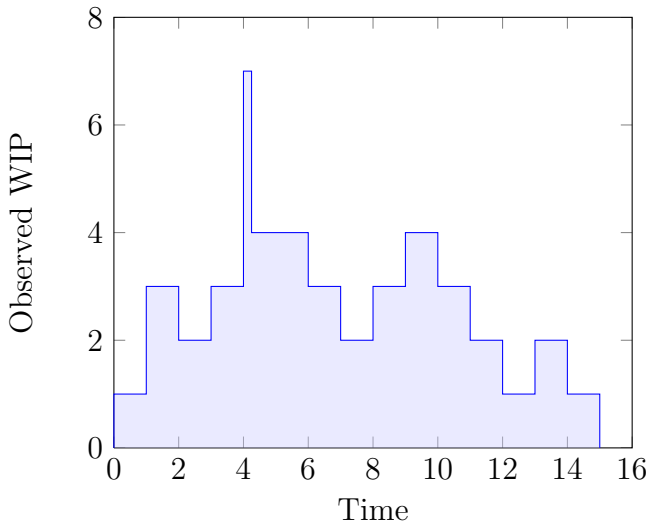

Figure 5.16: A simple graph illustrating WIP intensities over time.

of values. The author suggests that a preliminary run of the simulation be done using infinite buffers, and estimating the maximum individual buffer occupations, and then assigning these values to n_i (see Subsection 5.2.4.1). These values are not guaranteed to be the absolute maximum, but they provide a starting point.

Since the probability matrix P can become large if the number of buffers is large, and there is a large number allowed per buffer, the probability structure was modified to be represented by truncated Poisson distributions denoted by Pois_T. The Poisson distribution is defined on the integer range $0, 1, ..., \infty$, but in this experiment, the range of each buffer i was limited to $[0, n_i]$, thus implying *truncated Poisson distributions*. The literature does not prescribe a specific discrete distribution for the CEM, and usually Bernoulli distributions are used on algorithm initialisation. The advantage of the truncated Poisson approach is, however, that it requires only a vector of Poisson rates to be maintained, with the number of elements equal to $m - 1$. The cumulative distribution of the truncated Poisson

WIP level	0	1	2	3	4	5	6	7	8
WIP Area i	0	3	8	15	11	0	0	1.75	0
Proportion WIP i	0	0.077	0.284	0.671	0.955	0.955	0.955	1.000	1.000

Table 5.11: Example of WIP intensity proportions.

distribution with rate λ_i on the range $[0, n_i]$ is given by

$$F_i(x, \lambda_i, n_i) = \sum_{x=0}^{n_i} \frac{e^{-\lambda_i} \lambda_i^x}{x!}. \tag{5.9}$$

For the problems described here, the CEM parameter vector $\mathbf{v}$ is estimated using (3.14) and (5.9). The random vector $\mathbf{X} = (X_1, \ldots, X_n) \sim \text{Pois}_T(\lambda)$ and the joint probability mass function for n variables is

$$f(\mathbf{X}, \lambda) = \prod_{i=1}^{n} \frac{e^{-\lambda_i} \lambda_i^{X_i}}{F_i \times X_i!}. \tag{5.10}$$

Working towards (3.14), one takes the natural logarithms on both sides in (5.10) to obtain

$$\ln f(\mathbf{X}, \lambda) = -\sum_{i=1}^{n} [\lambda_i + X_i \ln \lambda_i - \ln F_i - \ln X_i!],$$

and the optimal parameter vector is found by solving for

$$\begin{aligned} \frac{\partial}{\partial \lambda_i} \sum_{j=1}^{N} I_{\{\rho_j = 0\}} \ln f(\mathbf{X}_j; \lambda_i) &= \frac{\partial}{\partial \lambda_i} \sum_{j=1}^{N} I_{\{\rho_j = 0\}} \sum_{i=1}^{n} [-\lambda_i + X_i \ln \lambda_i - \ln F_i - \ln X_i!] \\ &= \sum_{j=1}^{N} I_{\{\rho_j = 0\}} [-1 + \sum_{i=1}^{n} X_i / \lambda_i] \\ &= 0, \end{aligned}$$

which yields

$$\begin{aligned} \lambda_i &= \frac{\sum_{j=1}^{N} I_{\{\rho_j = 0\}} \sum_{i=1}^{n} X_i}{\sum_{j=1}^{N} I_{\{\rho_j = 0\}}} \\ &= \frac{\sum_{j=1}^{N} I_{\{\rho_j = 0\}} X_{ji}}{\sum_{j=1}^{N} I_{\{\rho_j = 0\}}}. \end{aligned} \tag{5.11}$$

Note that $I_{\{\rho_j = 0\}}$ in (5.11) is the indicator function that returns all non-dominated solutions after ranking (see Algorithm 3 and also Step 5 in Algorithm 7). The parameter vector is now taken as $\Lambda = \{\lambda_1, \ldots, \lambda_{m-1}\}$.

The truncated Poisson distribution on the given range $[0, n_i]$ is obtained by normalising the fundamental Poisson distribution, using

$$f_i(x, \lambda_i, n_i) = \frac{e^{-\lambda_i} \lambda_i^x}{F_i(x, \lambda_i, n_i) \times x!}. \tag{5.12}$$

Algorithm 8 Method to sample from a truncated Poisson distribution

State λ and n.
Set $C = \sum_{j=0}^{n} e^{-\lambda}\lambda^j/j!$.
Generate a random number U.
Set $i = 0$, $p_i = e^{-\lambda}$, $F = p_i/C$.
while $U < F$ **do**
    $p_{i+1} = \frac{\lambda}{1+i}p_i$.
    $F \leftarrow F + p_{i+1}/C$.
    $i \leftarrow i + 1$.
end while
$X = i$.

Sampling from this distribution is simple and a buffer value X on the defined range is easily obtained. The elements of Λ are arbitrarily initialised as $\lambda_i = n_i \cdot U(0,1)$. A simple method to sample from a truncated Poisson distribution with parameter λ over the range $[0, n]$ is shown in Algorithm 8.

The experimental setup is discussed next.

5.2.4.3 Experimental setup for the new BAP objective

For this experiment, the following methodology was followed, assuming a valid simulation model exists:

1. Determine the duration of the simulation transient period, as well as the required number of replications to obtain sufficient confidence intervals for the output parameters.

2. Using the simulation model, estimate the maximum buffer sizes for the unrestricted case of infinite buffers.

3. Decide on a value for p_q, say 0.95, $N = 20$ to $N = 25$, $\alpha = 0.6$ to $\alpha = 0.8$, and $\delta = 10^{-2}$.

4. Execute Algorithm 7, using (5.11).

Some of the BAP instances considered previously are again used in this experiment, but with input data modifications in some cases. The short names are continued from Table 5.7 and the instances considered are:

1. BAP17: The same as BAP1, i.e. no modifications, with $m = 5$, $n = 10$ and exponential processing and repair times. The values of B_i will be limited by a maximum value for each (see Table 5.13).

2. BAP18: A manufacturing line similar to BAP1, but with realistic real-world processing times: the times are offset by a minimum (a task cannot be executed in less time), and the distributions are truncated (a task has a finite duration). All the distributions are defined on the positive domain, since processing time cannot be negative. The lognormal distribution was found to be a satisfactory descriptor of the processing times.

3. BAP19: A manufacturing line similar to BAP18, but each workstation can rework a product just completed if it fails a quality inspection, or a product produced by the workstation can be completely rejected (Table 5.12). This type of system thus has feedback, and the number of completed products leaving the system is thus generally smaller than the number entered.

4. BAP20: The sixteen-node network of (Cruz *et al.*, 2008) studied previously (BAP16).

5. BAP21: The same as BAP20, but now all machines are allowed to fail. It is assumed that the failure rate is the same for all machines, and failure occurrences are count-based with distribution Poisson(19), while the downtimes are exponentially distributed with mean two hours.

6. BAP22: The same as BAP7, i.e. no modifications, with $m = 10$, $n = 10$ and exponential processing and repair times. The values of B_i will again be limited by a maximum value for each (see Table 5.13).

7. BAP23: The same as BAP4, i.e. no modifications, with $m = 5$, $n = 10$ and Erlang_2 processing and exponential repair times. The values of B_i will, as in the cases of BAP17 and BAP22, be limited by a maximum value for each (see Table 5.13).

5.2.4.4 Experimental results with the new BAP objective

The results for the seven BAP instances are shown in Table 5.13. The buffer sizes allowed are the maximum buffer occupation estimated via an independent experiment.

An example of the progress of the distribution parameters are shown in Figure 5.17 for the case of BAP17, in Figure 5.18, and in Figure 5.19 for BAP20, and finally in Figure 5.20 for BAP23. Three cases are presented here to cover the three

	Processing times		Rejection rate BAP19	
Machine	BAP17	BAP18, 19	Quality	Final
1	1.0	1.090+logn(0.080, 0.21)/0.41	3%	2%
2	1/1.1	1.080+logn(0.075, 0.17)/0.48	2%	2%
3	1/1.2	1.071+logn(0.068, 0.17)/0.58	2%	2%
4	1/1.3	1.069+logn(0.059, 0.17)/0.67	2%	2%
5	1/1.4	1.058+logn(0.058, 0.17)/0.74	2%	2%

Table 5.12: Model parameters for BAP17–19.

Instance	Buffer size B_i allowed	Problem size	Number of evaluations[1]	Size estimated PF	Max. T_R found
BAP17	35, 30, 25, 25	656 250	340	55	0.905
BAP18	30, 25, 20, 20	300 000	280	38	0.719
BAP19	25, 20, 15, 15	112 500	520	44	0.622
BAP20	15 for all i	$> 10^{17}$	1050	34	5.023
BAP21	15 for all i	$> 10^{17}$	400	59	4.043
BAP22	200 for all i	$> 10^{20}$	2 000	154	4.277
BAP23	35, 25, 20, 15	262 500	320	43	0.477

[1] To obtain Elite set.

Table 5.13: Simulated results for the seven BAP instances, for given maximum buffer sizes.

classes of problems described in Section 5.2.4.3. The σ_i-values of all the problems reduce quickly and then approach almost steady values. This behaviour is due to the nature of the CEM, since it is deduced from a variance reduction method. The σ_i-values approach zero in the case of single-objective optimisation by means of the CEM, but can only approach a finite value in the case of MOO, since there is always more than one solution in the final solution set.

Some solutions obtained for BAP17 are briefly discussed. In Figure 5.21, three labels ("A", "B" and "C") with solutions are shown. The values in the first pair of parenthesis show the buffer sizes allocated between each pair of machines. The second pair of parenthesis contains the estimated values for the WIP percentile and the throughput rate. The label marked with an "A" in Figure 5.21 shows the buffer allocations for the lowest throughput rate, and a WIP percentile of 0. This value is zero because all buffers sizes are zero, and the worst T_R is 0.484. The decision maker may require a higher value for T_R, in which case the solution shown

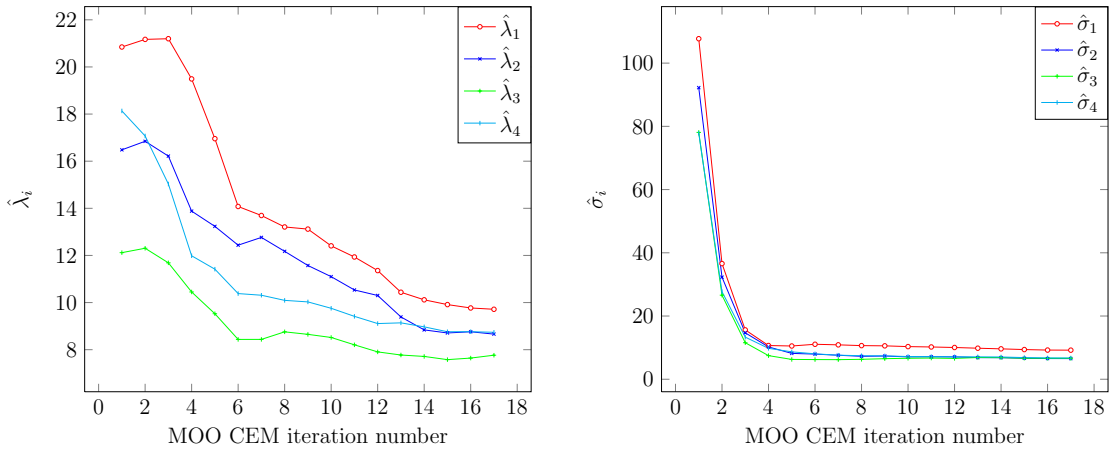

Figure 5.17: Progression of the values of $\hat{\lambda}_i$ and $\hat{\sigma}_i$ for the case of BAP17.

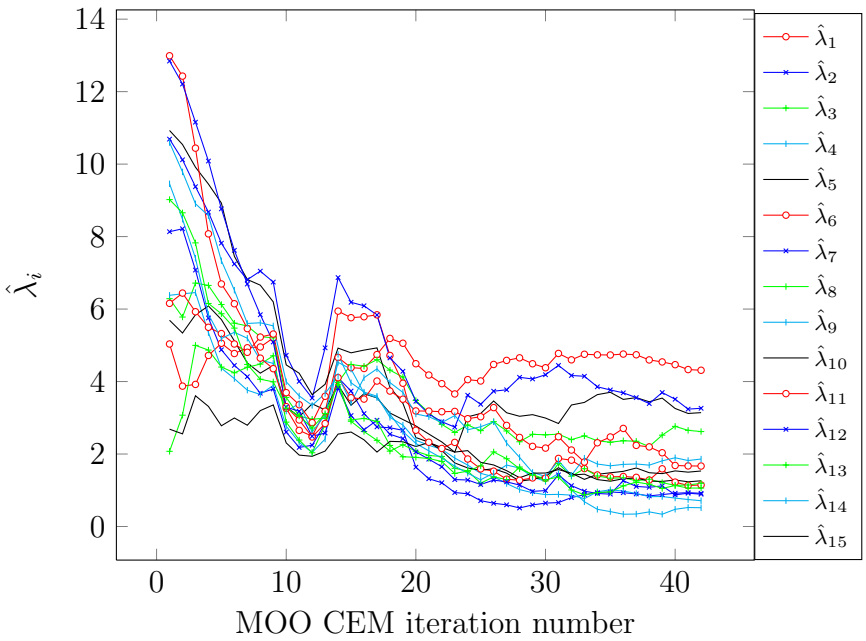

Figure 5.18: Progression of the values of $\hat{\lambda}_i$ for the case of BAP20.

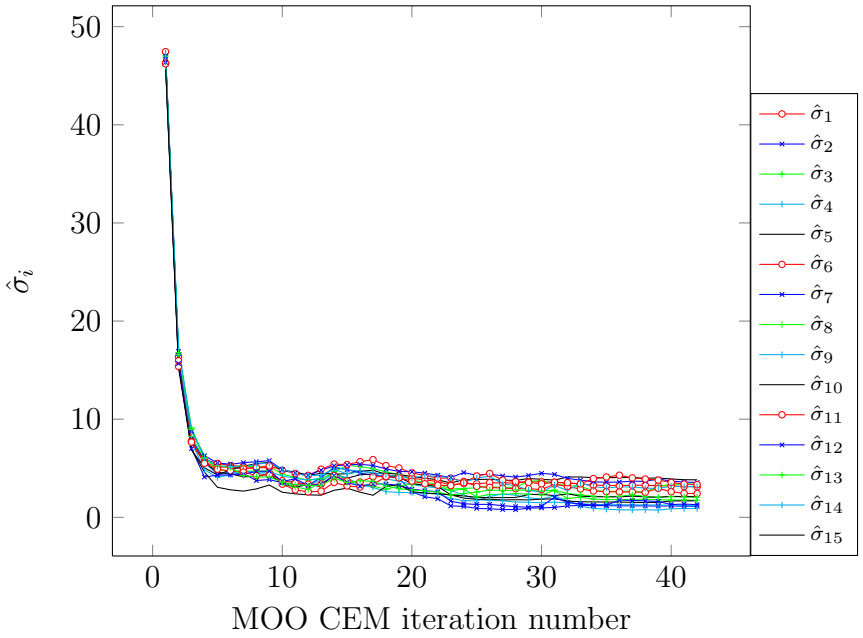

Figure 5.19: Progression of the values of $\hat{\sigma}_i$ for the case of BAP20.

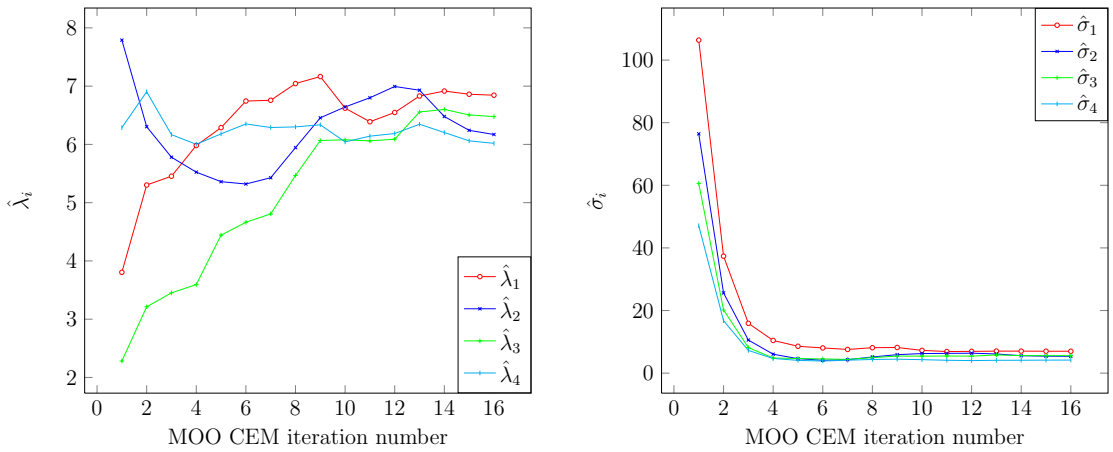

Figure 5.20: Progression of the values of $\hat{\lambda}_i$ and $\hat{\sigma}_i$ for the case of BAP23.

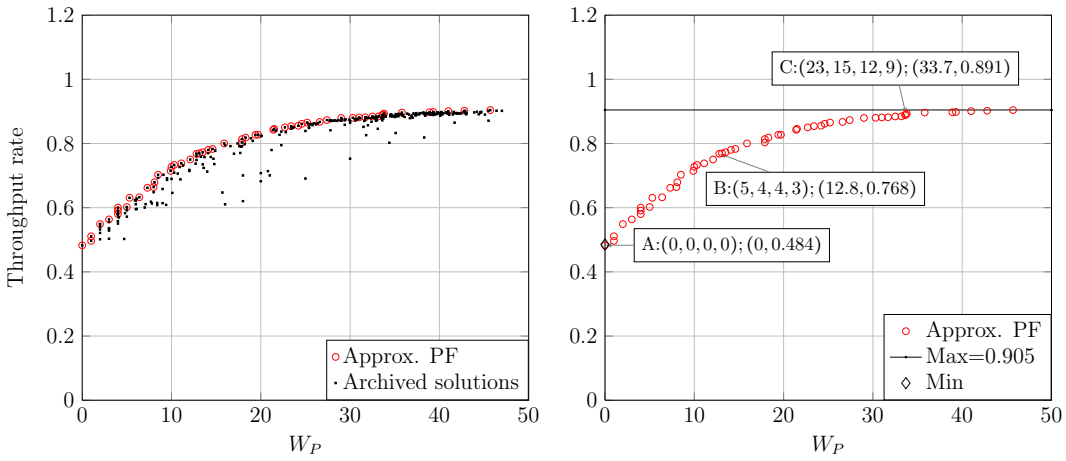

Figure 5.21: Approximate Pareto front and archive for BAP17.

by label "B" can be considered: the T_R is 0.768 while the 95-th WIP percentile is 12.8. An even higher T_R can be achieved, as shown by label "C", with a WIP percentile of 33.7 and a T_R of 0.891, which approaches the maximum T_R value of 0.904. If the decision maker requires that the WIP percentile be no more than 10, it follows from the detail solution set (not shown) that an allocation of (4, 3, 2, 5) must be implemented, with an expected T_R of 0.727.

Similar plots for BAP18 to BAP23 are shown in Figures 5.22 – 5.27, respectively. In each plot, three typical solutions are shown, similar to those in Figure 5.21 for BAP17. For each problem, the archived values are also included to show the search space covered, and to put the approximate Pareto fronts in perspective. The extreme minima and maxima of T_R were estimated for each problem, i.e. for the case of all buffers set to zero, and for all buffers infinitely large. The latter case is indicated by means of a solid line. It is encouraging to see that for each problem, both values of these extremes were found by the algorithm.

The case of BAP20 requires further discussion. The arrival rate is $\lambda = 5$ and the service rate at machine 1 is $\mu = 10$. Suppose buffer 1 is set to a small number, for example it is equal to 2, then, if buffer 1 is filled, machine 1 is blocked. However, arrivals still occur at a rate of 5, but Cruz *et al.* (2010) do not state what happens to these arrivals. If they are allowed to queue in front of machine 1, then the throughput rate remains 5, irrespective of the buffer allocations. Under this condition, it was found, for example, that the throughput rate is still 5 when all buffers are set to zero. The infinite queue in front of machine 1 ensures that there

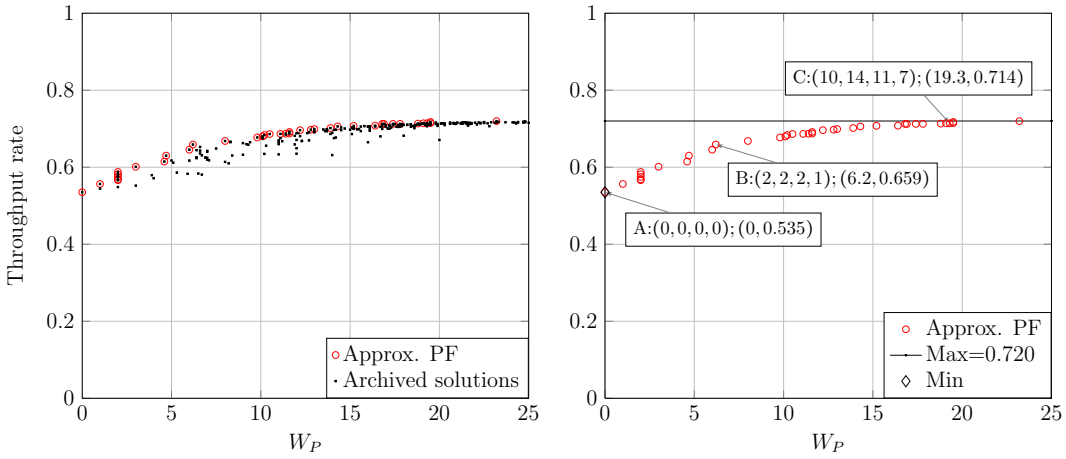

Figure 5.22: Approximate Pareto front and archive for BAP18.

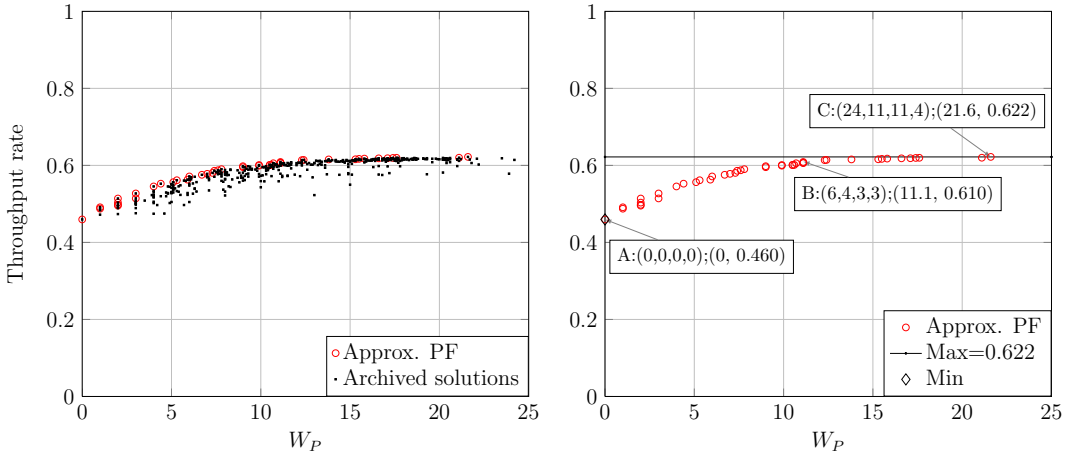

Figure 5.23: Approximate Pareto front and archive for BAP19.

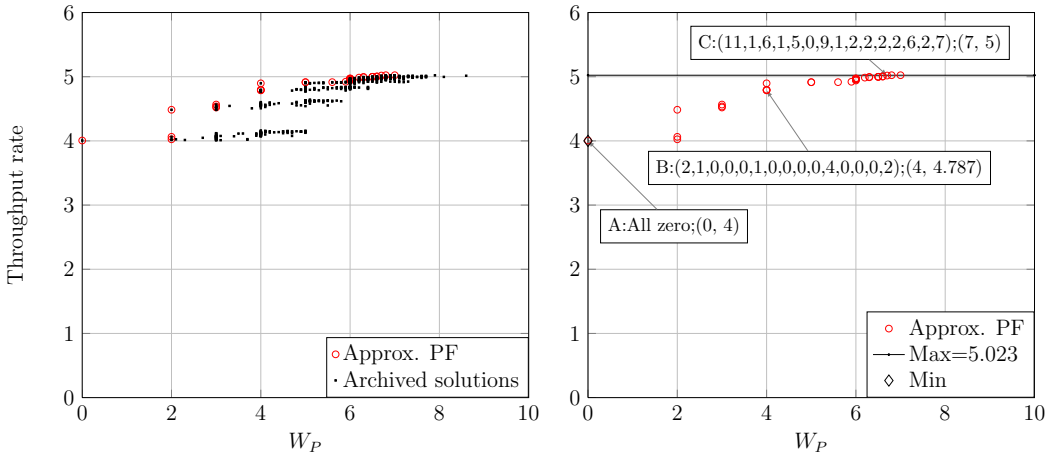

Figure 5.24: Approximate Pareto front and archive for BAP20.

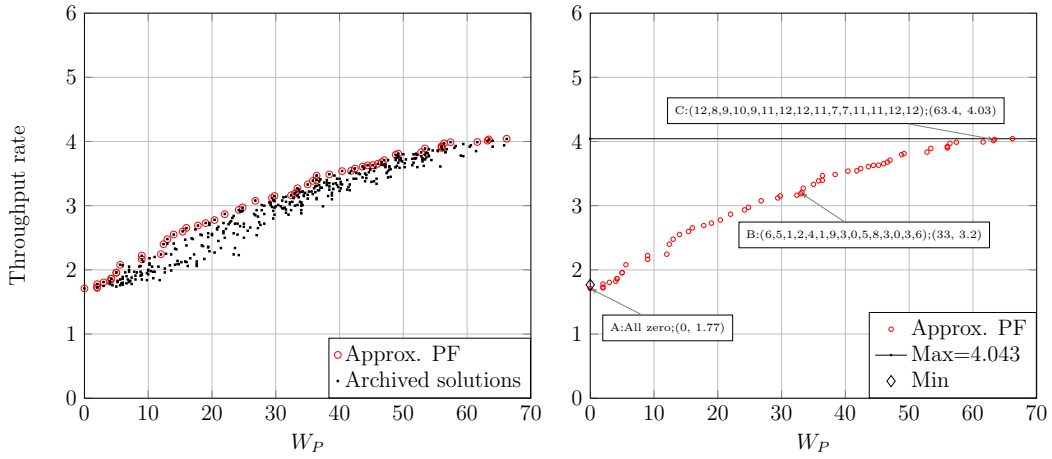

Figure 5.25: Approximate Pareto front and archive for BAP21.

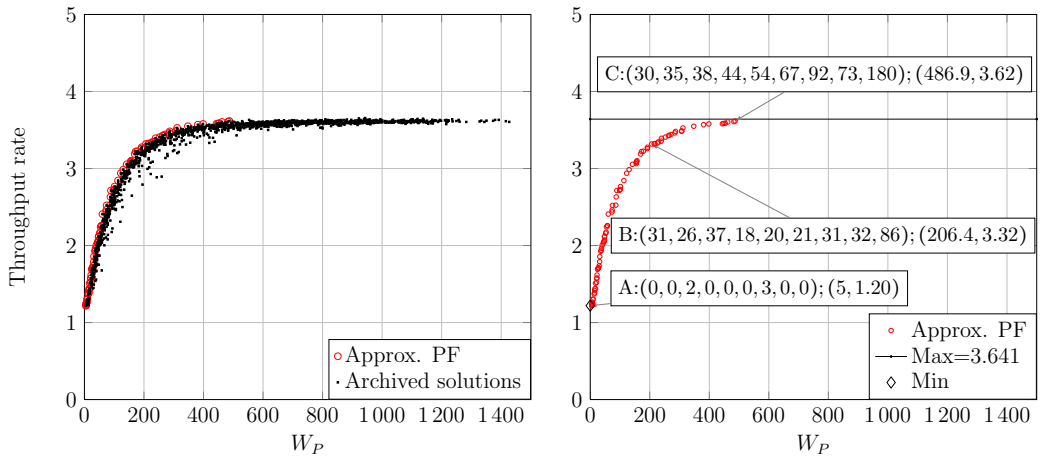

Figure 5.26: Approximate Pareto front and archive for BAP22.

is always work, and the slowest processors work at a rate of 5. If the arrivals are rejected when machine 1 is occupied, then a nett lower arrival rate is induced, i.e. the arrivals are effectively thinned. The author adopted the latter approach, which seems more realistic. Alternatively, the queue in front of machine 1 could also be considered a buffer to be minimised, but the original problem was not formulated in this manner.

This concludes the analysis of the BAP using the MOO CEM algorithm, and in this subsection it was shown that the MOO CEM algorithm can be used to find buffer sizes under an inequality constraint.

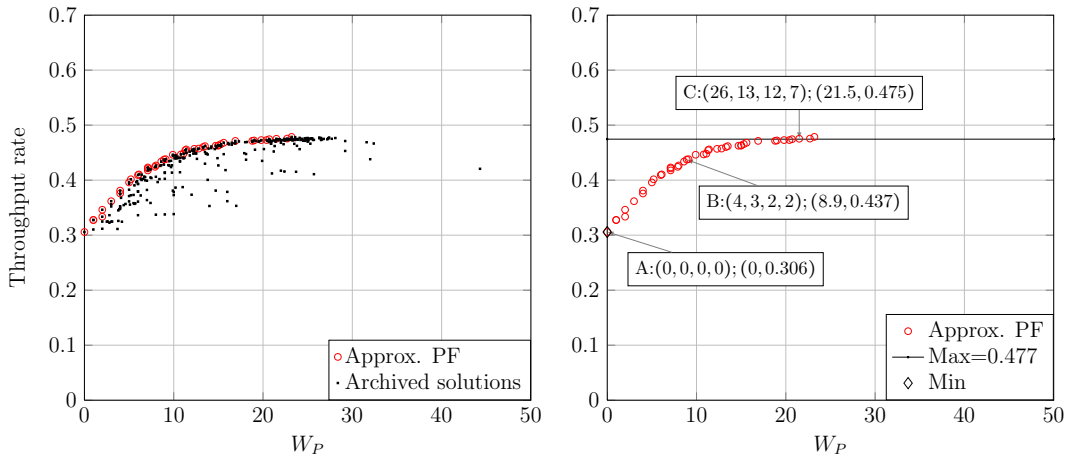

Figure 5.27: Approximate Pareto front and archive for BAP23.

5.2.5 The BAP: Summary and conclusions

In this section (Section 5.2), application of the MOO CEM algorithm for problems with discrete solution spaces was discussed. The well-known buffer allocation problem (BAP) was used as a test bench while varying the number of resources, the maximum allowable number of buffer spaces and the queuing network topology. It is often only possible to evaluate proposed solutions of discrete, dynamic stochastic processes by means of computer simulation, which can be time consuming, especially when the solution space is large, and it was shown that the CEM as a multi-objective optimiser can minimise the total buffer space by appropriately allocating space to each buffer while maximising throughput rate. The study of the BAP using the MOO CEM algorithm has been applied to estimating buffer allocations under the premise that the decision maker is willing to 1) pay for a fixed total number of buffers, or 2) to accept a stated maximum number of buffer spaces per buffer. It was shown that good solutions can be obtained and that a point of diminishing return in terms of total buffer allocation is reached.

The MOO CEM algorithm found good solutions (approximation Pareto sets) for the various BAP instances using fairly few computationally expensive simulations for solution evaluations. This supports the research hypothesis. Finding an approximate shape for the Pareto front enables the decision maker to better understand the relationship among objectives, which means that exact solutions are thus not always necessary.

The approach presented is pragmatic and aimed at practical solutions: the

Matlab® code for the MOO CEM is simple to use, and the decision maker has to provide a valid simulation model. No assumptions were made regarding the time and failure distributions of the simulated processes. A manuscript presenting the work in Subsection 5.2.4 has been conditionally accepted by the *International Journal of Simulation Modelling* (Bekker, 2012).

Further research can be done by applying the proposed MOO CEM algorithm to other types of queuing networks that have more than two objectives. Also, combining the MOO CEM algorithm, which is population-based, with a local search method might increase the convergence speed when analysing BAP instances with higher dimensions.

5.3 A reconfigurable manufacturing system

In this section, the application of the MOO CEM algorithm to a *reconfigurable manufacturing system* (RMS) is presented. Du Preez (2011) defines an RMS as *[a system having] an ability to reconfigure hardware and control resources at all of the functional and organisational levels, in order to quickly adjust production capacity and functionality in response to sudden changes in market or in regulatory requirements.* The system studied is still in an experimental phase. At the time of writing, a real-life prototype was under development by the Department of Mechanical and Mechatronic Engineering at Stellenbosch University.

This prototype emulates real industrial processes in which electrical switch gear is manufactured, including several variants of electrical circuit breakers. It consists of a closed conveyor that serves a number of workstations, and the system is intended to operate automatically. The workstations comprise a set of part feeders, welders, and quality inspection and part removal stations, while an inspection camera also serves the system. A number of expensive pallets capable of radio frequency communication are placed on the conveyor. Parts of a chosen product unit are loaded on a pallet by a part feeder and the pallet is conveyed to the welder where some welding steps are executed. The pallet with the finished product then proceeds to an inspection station, where it is removed from the pallet or recycled for rework. The empty pallet returns to the part feeder for its next cycle. This study was executed under the supervision of the author and the detail is described in Du Preez (2011). A schematic of the RMS layout is shown in Figure 5.28.

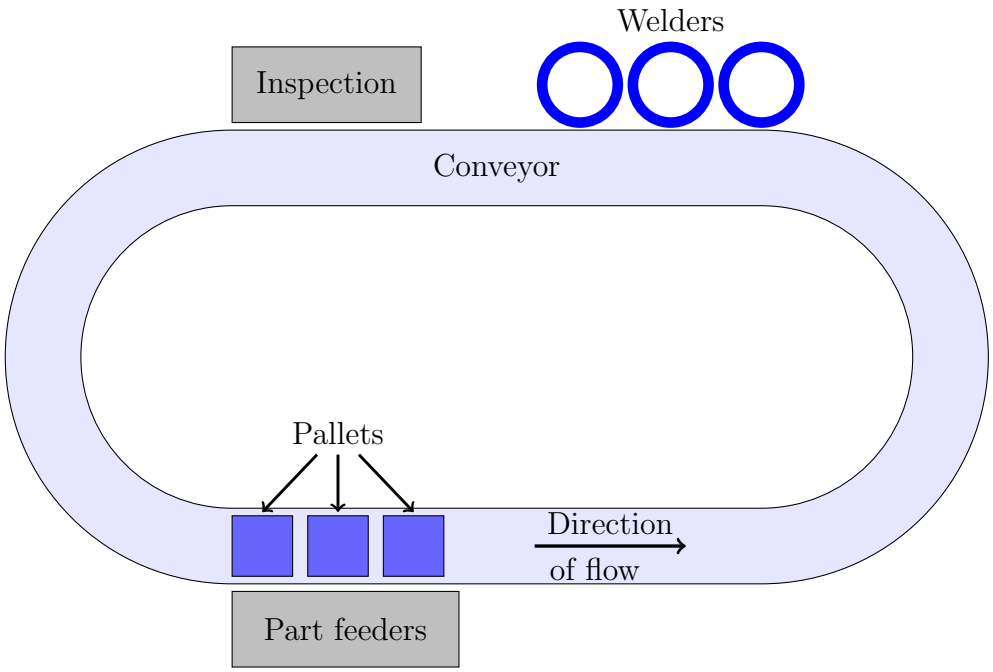

Figure 5.28: Schematic of a reconfigurable manufacturing system.

A manufacturing configuration of this system is defined by the physical layout and resource arrangement. Several manufacturing configurations can be designed, given the required operations and available resources. The best configurations can be determined via computer simulation in conjunction with the MOO CEM algorithm. The decision variables are the candidate configurations, the number of each type of resource applied (capacity variation), the number of pallets, the conveyor speed and pre-defined operational rules. An operational rule may be to allow pallets to cycle through the system if they cannot access a welder, or to allow the pallets to queue at the welders while remaining on the conveyor.

The optimisation objectives of this system are to maximise the throughput rate while minimising the work in progress, which are similar to the BAP described in Section 5.2. The total number of solution pairs for the current laboratory setup is 5 928. The objective values of each of these solution pairs were determined via exhaustive enumeration and are shown in Figure 5.29.

The true Pareto front for this problem is thus available, and after applying the MOO CEM algorithm to this problem, the estimated Pareto front can be compared to the true front. The true Pareto front for the solution set of the reconfigurable manufacturing system is shown in Figure 5.30 (31 solutions), together with an

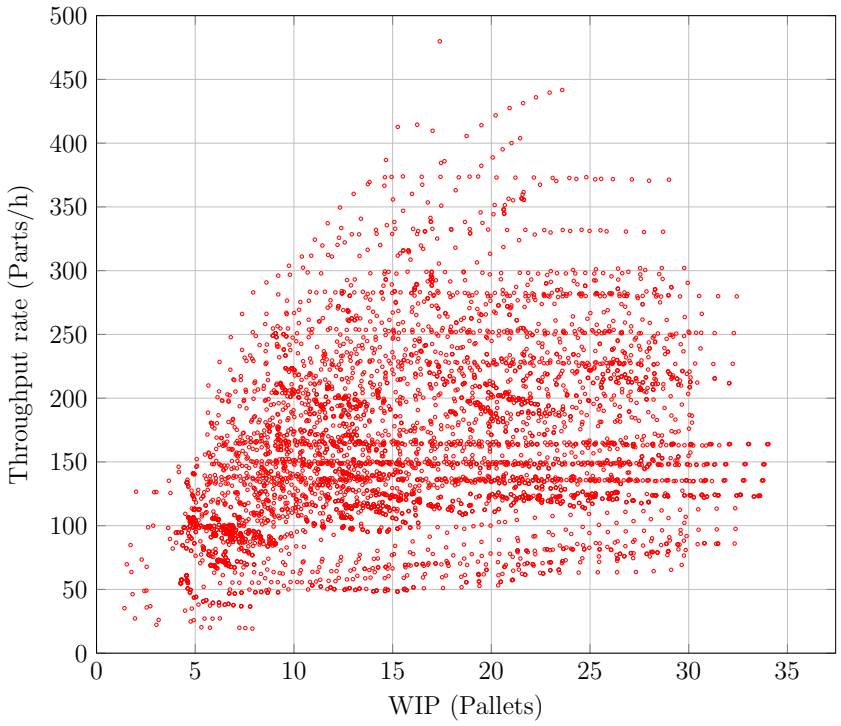

Figure 5.29: Results of an exhaustive enumeration of solutions for the reconfigurable manufacturing system illustrated in Figure 5.28.

approximate front consisting of 24 solutions found by the MOO CEM algorithm. The maximum number of evaluations allowed for the MOO CEM algorithm was 2 900, with a population size of 40.

This problem will form part of the experimentation in Chapter 6. Next, a test problem from the process engineering domain is analysed.

5.4 An extrusion equipment design problem

A deterministic, continuous MOO test problem dealing with the design of a polymer extrusion unit has been developed by Gaspar-Cunha & Covas (2003) and was made available to the research domain. This problem is presented here and analysed using the MOO CEM algorithm.

Extrusion is widely used in the process engineering domain to mass-produce products such as pipes and tubing for various industries. The process is typically achieved using a single Archimedes-type screw which rotates at a certain rate inside a heated barrel. A schematic from Gaspar-Cunha & Covas (2003) is shown

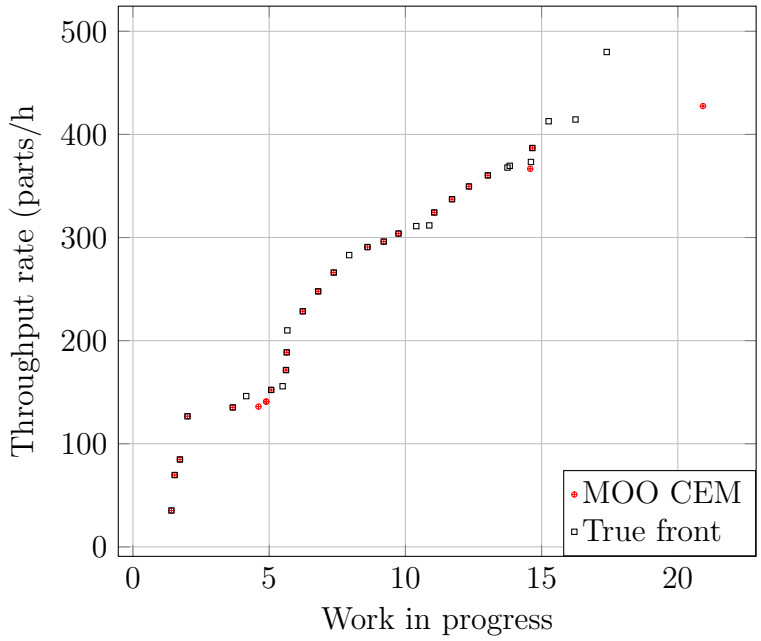

Figure 5.30: True and approximate Pareto front for the reconfigurable manufacturing system illustrated in Figure 5.28.

in Figure 5.31.

Polymer pellets are gravity-fed from a hopper into the heated barrel while the screw advances the mixture of solids (initial state) and molten material towards the moulding die (final viscoelastic state). This equipment has material-, geometry- and operating properties that define its performance for a given polymer. The performance depends on mass output (kg/h), mixing quality (to be maximised), screw length required (mm), mechanical power consumption (W), average residence time inside the equipment (t_r) and level of viscous dissipation. From these, the viscous dissipation and the resulting exit temperature must be minimised. For given material properties, the designer may vary the following parameters: the screw rotation rate (N_s), barrel temperatures in three zones (T_{b1}, T_{b2}, T_{b3} [°C]), degree of mixing (W_{ts}), internal diameters (D_i' [mm]), section lengths (ls_i [mm]), screw pitch (s_r), and the flight thickness (e_t [mm]).

Gaspar-Cunha & Covas (2003) provided a "black box" executable program which models the behaviour of the extrusion equipment, given the inputs stated above. The detail of their model is available in their publication and will not be discussed here. The model takes values from a text file and provides the resulting

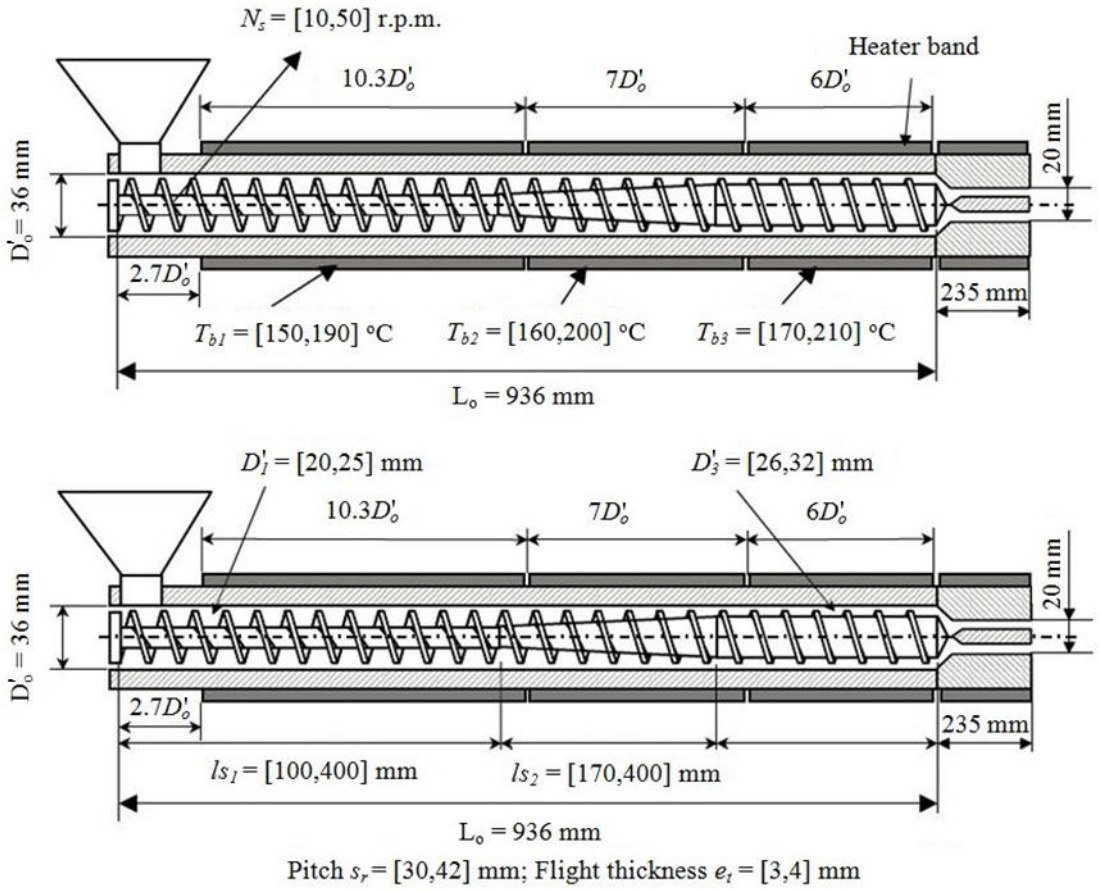

Figure 5.31: Schematic of a polymer extrusion unit (Gaspar-Cunha & Covas, 2003).

output in a different text file. For the purpose of this study, the model is assumed to be valid, and the MOO CEM algorithm code will manipulate the input text file while reading the output text file for the optimisation.

Two MOO experiments were conducted, and the first, which is referred to as "Design 1", is as follows: minimise the power consumption and maximise the output by changing the section lengths ls_1 and ls_2, the diameters D'_1 and D'_3, the screw pitch s_r and the flight thickness e_t. These are geometric design parameters, and their valid ranges are shown in Table 5.14. The operating parameters kept constant are as follows: N_s = 50 revolutions per minute and T_{b1} = 170 °C, T_{b2} = 190 °C and T_{b3} = 200 °C. The power may not exceed 9 200 W and the screw length is limited to 900 mm.

The result of the optimisation of Design 1 is shown in Figure 5.32. A subset of the solution population with 1 115 values was obtained, as opposed to the 4 000

ls_1 [mm]	ls_2 [mm]	D_1' [mm]	D_3' [mm]	s_r [mm]	e_t [mm]
100–400	170–400	20–26	26–32	30–42	3–4

Table 5.14: Ranges for design parameters of extrusion equipment.

evaluations allowed by Gaspar-Cunha & Covas (2003) for their experiments. The approximation set has 57 solutions.

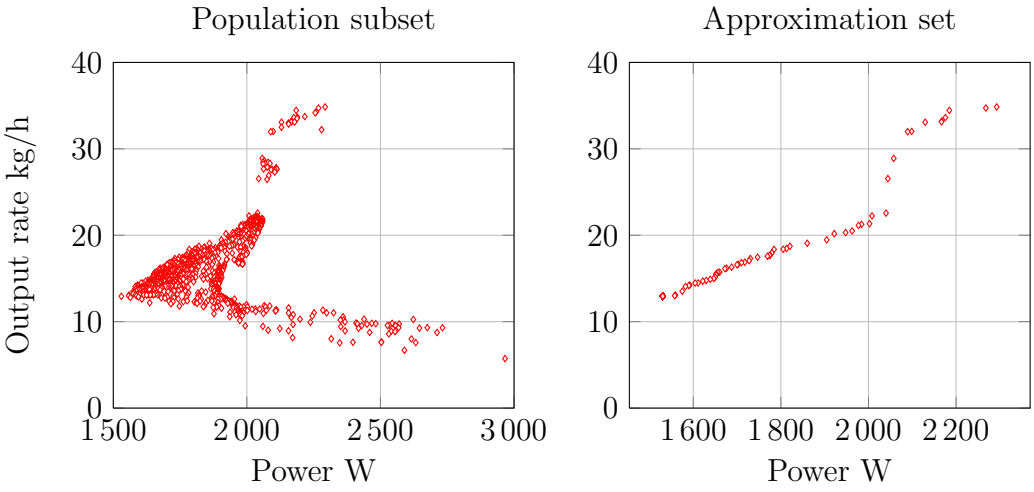

Figure 5.32: Population subset and optimality approximation set for Design 1 of an extrusion process.

The design requirements are shown in Table 5.15 for the solution vector with lowest values, as well as the solution vector with highest values. The designer can see from these values what are typical design ranges per geometric decision variable, and what their effects are on the output of the equipment.

The second experiment is called "Design 2" and it has 10 decision variables: the six decision variables of Design 1 were again considered, but operating decision variables were introduced, namely the screw rotation speed N_s and the three temperatures T_{bi}, while the same optimisation objectives were adopted. The valid

ls_1 mm	ls_2 mm	D_1' mm	D_3' mm	s_r mm	e_t mm	Power W	Output kg/h
400.00	397.67	26.00	26.45	41.22	3.93	2 292.89	34.85
312.66	311.00	20.00	26.00	31.90	3.07	1 530.33	12.94

Table 5.15: Extreme solution values of the extrusion design (Design 1).

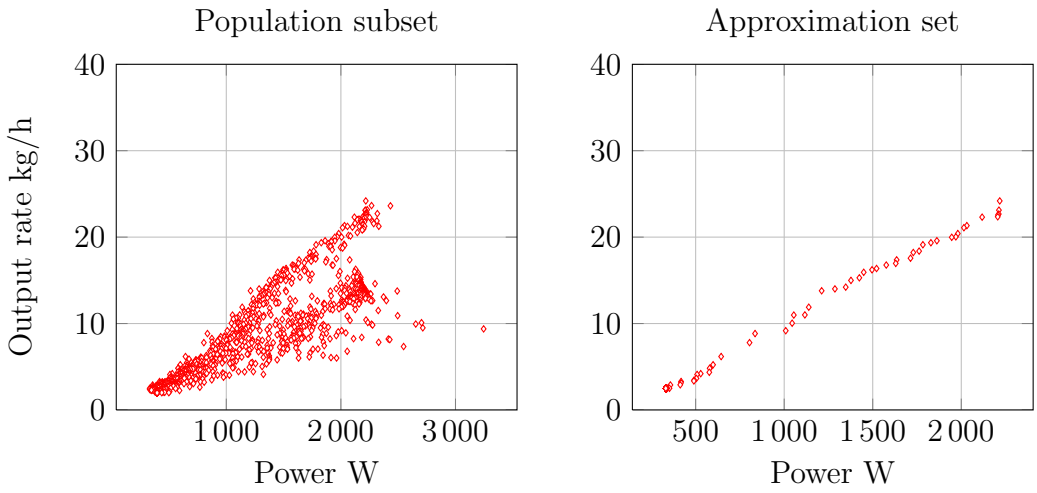

Figure 5.33: Population subset and optimality approximation set for Design 2 of an extrusion process.

ranges for the operating decision variables are 150 °C $\leq T_{bi} \leq$ 210 °C and 10 $\leq N_s \leq$ 60 revolutions per minute.

The results for this optimisation are shown in Figure 5.33, with 1 095 solutions in the population subset and 53 solutions in the approximation set. The shape of the approximate front agrees with that of Gaspar-Cunha & Covas (2003), while the front shown covers a wider range: 1 kg/h to 9 kg/h compared to 2.5 kg/h to 24 kg/h (MOO CEM), and 400 W to 2 000 W compared to 330 W to 2 217 W (MOO CEM).

Engineering design decisions can be made from the approximation front shown. Suppose the designer decides to set the power allowed at 2 000 W, then it is known that the extrusion unit will deliver product at a rate of approximately 20 kg/h, and to achieve this, the design values shown in Table 5.16 must be implemented, and the unit must be operated at the stated rotation speed and temperatures. The decimal parts of the decision variable values may be ignored if they are impractical to implement, or insignificant.

N_s r.p.m.	T_{b1} °C	T_{b2} °C	T_{b3} °C	ls_1 mm	ls_2 mm	D_1' mm	D_3' mm	s_r mm	e_t mm
55.60	203.41	208.37	205.44	400.00	389.16	25.80	28.28	41.21	4.00

Table 5.16: Proposed design and operating values for extrusion unit (Design 2).

The MOO CEM algorithm has been applied to a practical problem in process engineering design of a polymer extrusion unit. Two designs were presented: Design 1 defines six decision variables pertaining to geometry, while in Design 2 four operating decision variables were added to the problem, resulting in an optimisation problem with 10 decision variables. In both designs, the power requirement of the extrusion unit was minimised while it was attempted to maximise the mass output rate. The MOO CEM algorithm produced acceptable results while requiring fewer evaluations than the two algorithms (NSGA-II, RPSGAe) used by Gaspar-Cunha & Covas (2003).

5.5 CO gas management at an ilmenite smelter

An application of the MOO CEM algorithm was performed by Stadler (2012) under supervision of the author. This problem has stochastic, dynamic elements and provided a real-world opportunity for application of the MOO CEM algorithm, namely at the smelters of a heavy minerals mine. The problem and the related industry are briefly described, followed by a presentation of the results of the MOO CEM algorithm application. The aim of the study was to determine whether or not *carbon monoxide* (CO) gas can be re-used in smelting operations, since it would reduce the generation of *carbon dioxide* (CO_2) gas, and also reduce the acquisition cost of methane gas from an external supplier.

5.5.1 Background on the CO gas problem domain

The problem leading to the study originated at the smelter complex of Tronox KZN Sands, a South African mining company. The site is located near Richards Bay on the KwaZulu-Natal coast. Heavy minerals are mined from mineral sand deposits, and the minerals zircon, ilmenite and rutile are of importance. From ilmenite and rutile, titanium dioxide (TiO_2) is obtained. TiO_2 is widely used today for pigment production which is an intermediate product in paint manufacturing. A small percentage of TiO_2 is processed for titanium production. Titanium is used in the aircraft industry, as it has a high strength-to-mass ratio.

At Tronox KZN Sands, ilmenite is smelted in two direct current electric arc furnaces, and CO is a by-product of this process. CO is a hazardous gas, since it is highly inflammable and deadly when inhaled. The gas and other waste particles are captured in the off-gas systems of the furnaces, and the CO is flared into

the atmosphere. This means that the gas produced is allowed to burn, which is effectively a method of disposing of it.

The plant consists of sub-plants requiring various levels of energy for drying and preheating of materials. For these, methane gas is acquired at increasing cost from a supplier (the 2012 cost of the gas is triple that of the 2006 price). The question arose: can the CO gas not be re-used, instead of the expensive methane gas? Some engineering considerations must be mentioned when addressing this question. On the negative side, the hazards associated with CO gas induce risks and require special operational procedures and precautions. Also, CO gas has lower energy content (the calorific value) than methane gas, so more CO gas per volume will be required to yield a certain amount of energy. (The gas energy yield at this plant is measured in GJ.) Switching between gas types induces production delays, because the flow lines must be purged with nitrogen when CO was used. This is done to prevent flashbacks in the system, and the switching process takes time, as well as manpower. On the positive side, most burners on the plant can handle both gas types, and the piping, valves, instrumentation and control systems currently in use can accommodate either gas type. Re-using the CO gas will decrease the carbon footprint of the mine since less CO_2 will be produced.

There are seven sub-plants requiring gas in quantities varying over time. These requirements are unpredictable, and the supply volume of CO gas also varies. Ideally, a buffer pressure vessel should be installed to absorb fluctuations of supply and demand, but the cost is forbidding. The proposed alternative was to use CO gas when available, and to buy methane from the current supplier in order to complement unfulfilled requirements. The CO gas problem formulation is presented next.

5.5.2 Formulation of the CO gas problem

Since there are seven sub-plants with different demands, it was natural to ask "what-if" questions in terms of CO gas assignments. For example, is it better to distribute gas according to a predetermined priority policy, or should gas always be provided to the larger consumers only? The equipment at the sub-plants experience downtimes due to maintenance and failures, which add to the complexity of the decisions. To address these, a computer simulation model of the processes was developed by Stadler (2012), and various reliability data and other drivers were obtained from the information system of Tronox KZN Sands.

Variable	Pertaining to	Range
DV_1	Sub-plant 1	0 or 1
DV_2	Sub-plant 2	0 or 1
DV_3	Sub-plant 3	0 or 1
DV_4	Availability 1	80% – 100%, step 0.5%
DV_5	Availability 2	86%, 90%, 94%
DV_6	Availability 3	86%, 90%, 94%
DV_7	Availability 4	86%, 90%, 94%

Table 5.17: Decision variables used in the CO gas problem.

After conducting preliminary experiments, it was concluded that the problem exhibits a multi-objective nature, and seven decision variables and two objectives were identified. Three of the seven sub-plants were eventually considered in the study due to a safety-driven decision by the management of the plant. The decision variables associated with these sub-plants are boolean, i.e. a specific sub-plant either receives CO gas or not. The availability of other equipment led to the identification of four more decision variables related to availability, and are expressed as percentages. Subject-matter experts working at the plant provided ranges and resolution requirements. The decision variables are summarised in Table 5.17, and the size of the decision space is 7 560 possible scenarios $(2^3-1)\times(3^3)\times 40$, the -1 is due to the fact that at least one of DV_1, DV_2 or DV_3 must be "1").

The objectives of the problem are f_1, the average number of hours lost per day due to switching of gas type, which must be minimised, and f_2, the average overall saving per month on buying methane gas, which must be maximised.

5.5.3 Results of the CO gas problem

The simulation model was integrated with the MOO CEM algorithm in Matlab®, as was done before for other simulation models. The settings for the algorithm were as follows: $N = 20$, $\delta = 10^{-4}$ and $\alpha = 0.6$. Stadler (2012) performed an exhaustive enumeration of the problem for research purposes, and the result, together with the true Pareto front, is shown in Figure 5.34.

An approximate Pareto front was found after 300 evaluations, and is shown in Figure 5.35, together with the true Pareto front.

Management decisions can be made from the approximate solution set and the associated decision variable values. Suppose management aims to maximise saving on methane gas expense, then it must be accepted that five hours per day will be

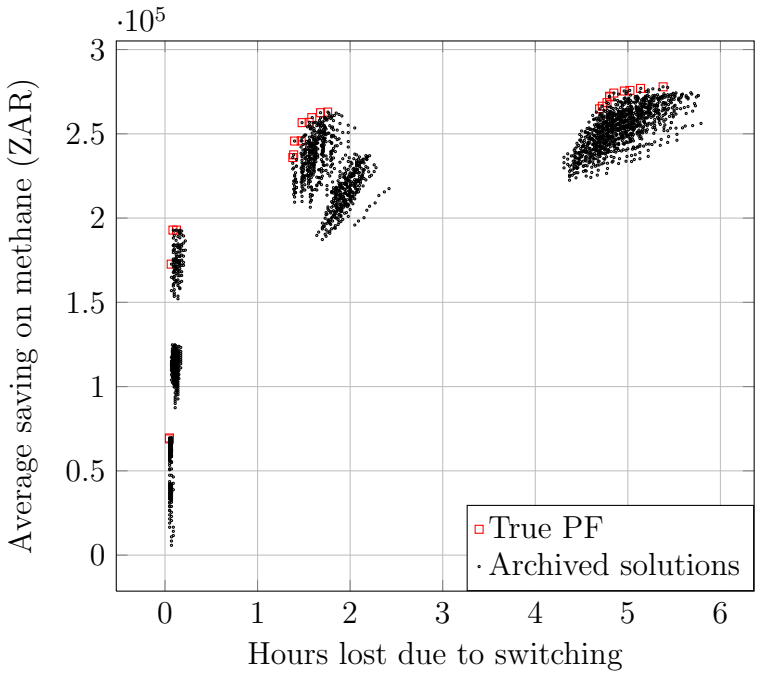

Figure 5.34: The complete solution set for the CO gas problem with the true Pareto front.

lost due to switching, but approximately ZAR275 000 will be saved on methane cost, per month. The operational requirements associated with this decision are presented in Table 5.18 as Scenario 1. In the same table, Scenario 2 is based on the requirement that no more than two hours be lost, and the saving on methane for the stated operational conditions is approximately ZAR262 000 per month.

Variable	Pertaining to	Scenario 1	Scenario 2
DV_1	Sub-plant 1	1, CO gas supplied.	1, CO gas supplied.
DV_2	Sub-plant 2	1, CO gas supplied.	1, CO gas supplied.
DV_3	Sub-plant 3	1, CO gas supplied.	0, no CO gas supplied.
DV_4	Availability 1	Ensure 94.5% availability.	Ensure 94.5% availability.
DV_5	Availability 2	Ensure 94.0% availability.	Ensure 94.0% availability.
DV_6	Availability 3	Ensure 86.0% availability.	Ensure 94.0% availability.
DV_7	Availability 4	Ensure 94.0% availability.	Ensure 94.0% availability.

Table 5.18: Decision variables used in two scenarios pertaining to the CO gas problem.

From a practical point of view, it should be noted that the feasibility of these proposals are subject to other engineering considerations. The maintenance

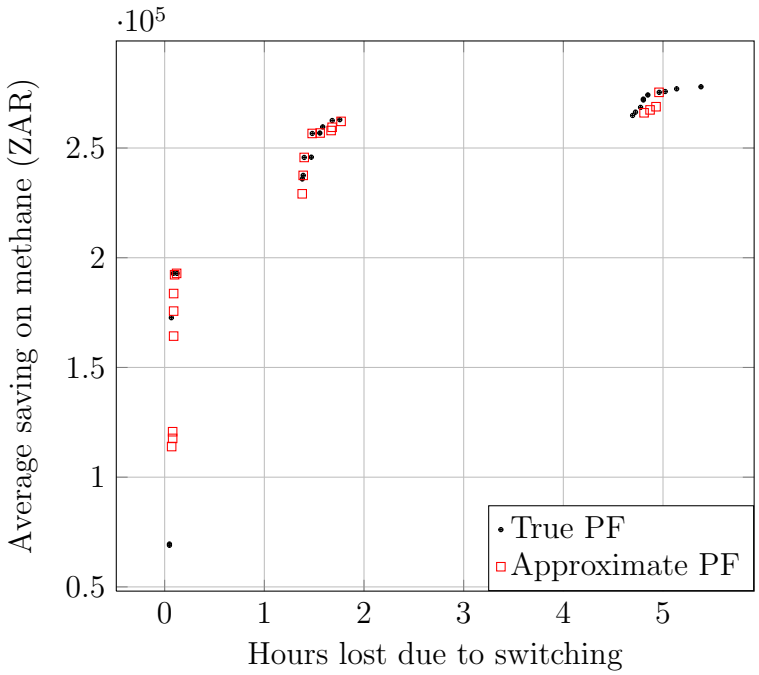

Figure 5.35: The true and approximate Pareto fronts for the CO gas problem.

engineers and teams need to demonstrate that the required equipment availability can be achieved, and that the associated cost does not mitigate the saving on methane gas. This study is still valuable, as it provided the management of Tronox KZN Sands with new information and system insight. The mining industry is under pressure to reduce its environmental impact, and the results of this study can lead to a smaller carbon footprint.

This concludes the discussion of the CO gas problem at Tronox KZN Sands.

5.6 Summary: Chapter 5

The MOO CEM algorithm has been applied to a number of dynamic, stochastic problems in order to provide supporting evidence of the research hypothesis. These problems were a generalisation of the (s, S) inventory problem, the buffer allocation problem in a number of variants, and a reconfigurable manufacturing system. In addition, polymer extrusion equipment design has been studied as an engineering design case. This design problem is continuous and deterministic. Finally, a real-world, practical engineering problem from the heavy minerals mining industry was presented. This problem is dynamic with stochastic elements, and the results provide useful guidance for management decisions. All the problems studied have

a varying number of decision variables and two objectives. In all cases there was evidence of relatively low numbers of objective function evaluations required for MOO.

For some problems, quality indicators were determined, and for others, graphical assessment was performed. A final assessment experiment which consolidates performance quality of the MOO CEM algorithm on the same basis with respect to some of these problems will be presented in the next chapter. In the experiment, the performance quality of the MOO CEM algorithm is measured against that of a commercial MOO algorithm.

CHAPTER 6

COMPARATIVE ASSESSMENT OF THE PROPOSED ALGORITHM

The proposed algorithm has been tested on several diverse problems and it was found that for standard benchmark problems, it requires fewer evaluations than other MOO algorithms (see Chapter 4 and Chapter 5). A final assessment is to determine whether or not the MOO CEM algorithm can compete with other MOO algorithms in terms of the qualities of solutions that it produces. A natural choice for comparison is the MOO genetic algorithm (GA) of Matlab®, since the latter is a commercial product based on the NSGA-II algorithm of Deb *et al.* (2002), and the NSGA-II is used as a benchmark in many algorithm-comparative studies. All the problems previously studied are again considered for assessment. These include the continuous benchmark problems (Chapter 4), the discrete stochastic inventory problem (Section 5.1), the variants of the discrete BAP (Section 5.2), the discrete reconfigurable manufacturing problem (Section 5.3), and the CO gas problem (Section 5.5). The assessment experiments and results are presented and discussed in this chapter.

6.1 Introduction to algorithm assessment

Before the assessment information is presented, a few points need discussion. Firstly, the assessment is based on a tutorial by Knowles *et al.* (2006), in which methods and assessment quality indicators are suggested. Only essential aspects from the tutorial are repeated here. Secondly, the term "quality performance" is used as explained before (see Subsection 2.3.4) instead of the general term "performance," because the latter pertains to *time* and *quality*. The term "quality indicator" refers

to quantification of the quality performance. In the experiments in this chapter only quality is assessed, as will be shown later. The time quality indicator is assessed by implication, as the number of objective function evaluations is restricted for each problem. Thirdly, the quality assessment is performed in the objective (or solution) space, while the solutions produced by the two MOO algorithms are approximations to the true Pareto front of each optimisation problem. These are referred to as "approximation sets." For assessment, these approximation sets are compared, while considering the true Pareto fronts of the benchmark problems. Since the two algorithms contain stochastic elements and also operate on stochastic problems in some cases, the resulting Pareto approximation solution sets vary from experimental run to experimental run. Any quality statements resulting from experimental assessments are thus probabilistic in nature.

Knowles *et al.* (2006) state that there is no "best" performance assessment technique. They discuss in detail three approaches to algorithm performance assessment, which are specifically applicable when algorithms are compared. These approaches are:

1. The *attainment function* approach, which represents the result of an optimisation algorithm as a probability density function,

2. The *indicator* approach, which summarises the result of an algorithm in one or more quantitative indicators (discussed above), and

3. The *dominance-based ranking* approach, in which many approximation sets are created by the algorithms, which are pooled and ranked. The set of ranks is then used to determine whether or not there is a significant statistical difference between the rank distributions of the output of the algorithms.

The second approach is used in this experiment.

6.2 Quality indicators

In optimisation one often uses a preference structure to judge the quality of solutions. For example, $A < B$ means A "is better than" B when minimising. This naturally leads to the need for quantitative quality indicators, which quantify quality performance. A unary quality indicator

$$I_q : \Omega_q \mapsto \mathbb{R}, \tag{6.1}$$

maps the set of all approximation sets Ω_q to the set of real numbers (Knowles *et al.*, 2006), which means that "A is preferable to B" if and only if $I_q(A) < I_q(B)$ when lower indicator values indicate preferable solution sets.

Many quality indicators have been developed to assess the performance of MOO algorithms. Some of these were discussed in Subsection 2.3.4, while performance quality values for the MOO CEM algorithm with respect to test problems were presented in Table 4.2. Although these quality indicators give good indications of the quality performance of an algorithm, Knowles *et al.* (2006) showed that they should not be used when comparing algorithms. A "good" quality indicator is *Pareto compliant*, which means that if approximation set A is preferable to B, the quality indicator value for A should be at least as good as the indicator value for B, with respect to weak Pareto dominance (Knowles *et al.*, 2006). Formally, it is stated as: The indicator in (6.1) is *Pareto compliant* if and only if for all $A, B \in \Omega_q$ such that $A \preceq B$ it holds that $I_q(A) \leq I_q(B)$, assuming that lower indicator values represent higher quality, as before. Having values for the indicators means one can compare the results produced by multi-objective algorithms by comparing the corresponding indicator values.

In the experiments presented here, the *hypervolume* and the *epsilon* indicators are used. These are both recommended by Knowles *et al.* (2006), are Pareto compliant and fairly simple to estimate. It should be noted that the quality indicators in Table 4.2 are all Pareto non-compliant.

The hypervolume indicator I_H is to be maximised and measures the portion of objective space that is weakly dominated by an approximation set A. In this experiment, I_H is referred to as a "hyperarea" since all problems presented here have two objectives. The objective space must be bounded, or a *strictly dominated* reference point must be provided. A simple example is shown in Figure 6.1, with minimisation Pareto set $\{(1.5, 2), (2, 1), (3, 0.5)\}$, $I_H = 14.5$ and the reference point at $(5, 5)$. It is easy to see that the I_H indicator measures spread and proximity ("closeness") of the approximation set to the true Pareto front.

The second indicator is the unary epsilon indicator, which has a multiplicative and additive version. The latter, $I_{\epsilon+}$, is used in this study. With respect to solution set A and $\mathcal{P}_R$ as reference set, it is defined as

$$I_{\epsilon+}(A) = I_{\epsilon+}(A, \mathcal{P}_R). \tag{6.2}$$

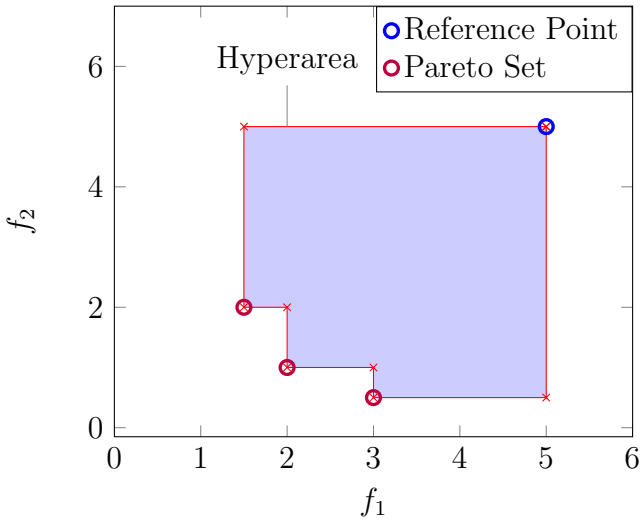

Figure 6.1: Example of a hyperarea and reference point.

It is interpreted as the minimum value by which each coordinate in the approximation set A must be adjusted to ensure that this set A dominates the reference set $\mathcal{P}_R$. Both epsilon indicators must be minimised.

6.3 Assessment experiment

In this study, the indicator approach was followed for algorithm quality performance assessment. Approximation sets were generated and from these, quality indicator values were calculated. The indicator values were then compared using standard statistical procedures because both the algorithms and some assessment problems have stochastic elements, as mentioned before. Researchers often use non-parametric statistical tests to compare the output of algorithms, such as the Mann-Whitney-Wilcoxon rank-sum test (Igel *et al.*, 2007), since no assumption of the underlying output distributions is made. In this experiment, 1 000 observations were made per algorithm per test case. The standard two-sample t-test (not the paired t-test) was used to test the hypotheses described below, because it allows for a statement regarding the direction of difference, i.e. "the mean of sample A is equal/less/greater than the mean of sample B". The basic null hypothesis of this test states that data in the two test sets are independent random samples from normal distributions with equal means and unequal, unknown variances, against the alternative that the means are not equal. In this assessment, a right-tailed test was performed on the hyperarea output, and a left-tailed test on the epsilon

output. The test statement is thus "the data in the two test sets are independent random samples from normal distributions with equal means and unequal, unknown variances, against the alternative that the mean due to the MOO CEM algorithm is greater than the mean of the Matlab® MOO GA" (hyperarea case).

The hypothesis tests were conducted at a significance level of 5% and are formulated as follows:

1. Hyperarea: Let $m^{(i)}_{CI_H}$ be the mean of the approximation sets produced by the MOO CEM algorithm of the i-th test problem, and that of the Matlab® set $m^{(i)}_{MI_H}$. Then the one-sided right-tail hypothesis for assessment can be stated as

$$\begin{aligned} H_0 &: m^{(i)}_{CI_H} \leq m^{(i)}_{MI_H} \\ H_1 &: m^{(i)}_{CI_H} > m^{(i)}_{MI_H}. \end{aligned}$$

2. Epsilon indicator: Let $m^{(i)}_{CI_{\epsilon+}}$ be the mean of the approximation sets produced by the MOO CEM algorithm of the i-th benchmark problem, and that of the Matlab® set $m^{(i)}_{MI_{\epsilon+}}$. Then the one-sided left-tail hypothesis for assessment can be stated as

$$\begin{aligned} H_0 &: m^{(i)}_{CI_{\epsilon+}} \geq m^{(i)}_{MI_{\epsilon+}} \\ H_1 &: m^{(i)}_{CI_{\epsilon+}} < m^{(i)}_{MI_{\epsilon+}}. \end{aligned}$$

Rejection of both null hypotheses in favour of the alternative hypotheses is desirable.

Also, to find possible evidence in support of the research hypothesis, the number of test problem evaluations was limited for both algorithms. Since the MOO CEM produced good results while limited to 10 000 evaluations in the continuous case, this limit was also set in this experiment with a fixed population size of $N = 100$. For the discrete cases, the archives developed earlier (see Section 5.2) were considered as the populations of solutions, and the maximum number of evaluations was set to $N\lfloor N_a/(2N)\rfloor$ per problem, with N_a the archive size as before, and $N = 25$. This is necessary because it does not make sense to search a solution space of, for example, 286 solutions (BAP1) by performing 10 000 evaluations.

The Matlab® MOO GA has various options. The Pareto fraction was set to one, while the default settings were accepted otherwise. To allow for 10 000

evaluations with a population size of 100, the maximum number of generations was set to 100. For the discrete problems, the maximum generation number was set to $\lfloor N_a/(2N) \rfloor$, with $N = 25$.

The test procedure below was followed for each MOO test problem instance:

1. Run the MOO CEM algorithm for 1 000 pseudo-independent replications, allowing the maximum number of objective function evaluations per replication.

2. For each replication, observe the values of the unary Pareto non-compliant quality indicators as well as the hyperarea, and the epsilon indicator.

3. Repeat the two previous steps using the MOO genetic algorithm in the Matlab® Optimisation Toolbox, allowing the same maximum number of function evaluations.

4. Determine the means and confidence interval half-width values for the four unary quality indicators SP, GD, ME and CV.

5. Perform the two-sample t-test (`ttest2` in Matlab®) on both the hyperarea (right-tail) and epsilon indicator (left-tail) sets, and record the outcomes of the respective hypothesis tests. The algorithm producing the largest significant hyperarea can be considered superior, while it is required that the epsilon indicator be as small as possible.

6. Produce a box-whisker plot with notches for each indicator.

The results emanating as a result of executing these steps are presented next.

6.4 Algorithm assessment results

The mean values of the Pareto non-compliant unary quality indicators are listed in Table 6.1.

The means and 95% confidence interval half-widths h_w of the hyperarea (HA) and epsilon quality indicators for both algorithms are shown in Table 6.2. Also shown are the best known hyperareas, which were calculated using the known Pareto fronts (continuous problems) and the Pareto fronts estimated from the archives (discrete cases, see Section 5.2). The maximum number of objective

	MOO CEM				Matlab® MOO GA			
Test problem	*SP*	*ME*	*GD*	*CV*	*SP*	*ME*	*GD*	*CV*
MOP1	0.0351	0.1004	0.0027	0.0116	0.0291	0.0605	0.0031	0.0225
MOP2	0.001	0.0065	0.0004	0.0009	0.0059	0.0253	0.0012	0.0104
MOP3	0.0296	0.0554	0.0005	0.0036	0.0864	0.0985	0.0103	0.034
MOP4	0.0502	0.3254	0.002	0.0061	0.0794	0.2583	0.0117	0.0294
MOP6	0.0028	0.0243	0	0.0011	0.0145	0.0534	0.0112	0.0108
ZDT1	0.0024	0.0175	0.0003	0.002	0.0174	0.0776	0.1447	0.0139
ZDT2	0.0067	0.0318	0.0023	0.0058	0	1000.0965	0.326	1000.013
ZDT3	0.0095	0.0514	0.0007	0.0049	0.0182	0.0674	0.1484	0.013
BAP1	0.1402	0.2511	0.0121	0.072	0.1064	0.1858	0.0123	0.0372
BAP2	0.1487	0.2407	0.0136	0.0564	0.1552	0.1856	0.0155	0.0621
BAP3	0.195	0.2961	0.0141	0.0428	0.2397	0.2161	0.0231	0.0787
BAP4	0.1747	0.326	0.0187	0.0946	0.0347	0.0768	0.0158	0.0092
BAP5	0.2625	0.3609	0.0349	0.1321	0.0889	0.1536	0.0313	0.0143
BAP6	0.4739	0.5072	0.043	0.1488	0.1548	0.1785	0.0386	0.0265
BAP7	0.5898	0.6264	0.0947	0.1658	0.1737	0.1758	0.107	0.0253
BAP8	0.3312	0.4707	0.0611	0.1128	0.1398	0.204	0.0697	0.038
BAP9	0.5075	0.5946	0.0512	0.1439	0.2147	0.2244	0.0132	0.0638
BAP10	0.5926	0.6957	0.0636	0.1136	0.3226	0.275	0.0983	0.0792
BAP11	0.6284	0.6155	0.0397	0.1162	0.4362	0.3154	0.0528	0.0937
BAP12	0.6717	0.6399	0.1063	0.1157	0.4435	0.3105	0.1617	0.0975
BAP13	1.7412	1.4215	0.0621	0.2038	0.7388	0.5772	0.0811	0.0884
BAP14	0.1294	0.2482	0.018	0.0576	0.1036	0.1936	0.0182	0.0421
BAP15	0.1641	0.2612	0.0128	0.0584	0.171	0.1934	0.0194	0.0612
BAP16	0.1598	0.2723	0.0125	0.0373	0.2503	0.2218	0.0362	0.0802
BAP17	0.6877	0.468	0.0509	0.1161	0.8405	0.462	0.0645	0.1744
BAP18	0.6025	0.4794	0.1349	0.125	0.5478	0.3963	0.1763	0.1023
BAP19	0.5174	0.5181	0.3313	0.1022	0.4201	0.3614	0.404	0.0845
BAP20	0.3007	0.5455	0.0276	0.0841	0.1397	0.2757	0.0206	0.02
BAP21	0.8976	0.5049	0.0586	0.131	1.2664	0.5229	0.0682	0.2113
BAP22	21.4659	4.2398	2.3194	0.3063	20.9593	2.4789	5.4758	0.6099
BAP23	0.4109	0.3302	0.0438	0.0898	0.4839	0.3247	0.0722	0.1167
INVN	52.0778	5.2197	34.829	1.3040	35.7459	2.8955	36.4917	0.7238
RMS	12.443	2.751	0.958	0.6230	7.260	1.165	1.051	0.3150
COGas	5 730.3708	52.4308	2 526.9208	7.678	1 753.1257	17.1469	1 049.4605	1.4018

Table 6.1: Mean indicator values found during comparative testing.

function evaluations and number of generations of the Matlab® MOO GA are also included.

The results of the hypothesis tests for the continuous problems using the hyperarea indicator are shown in Table 6.3.

The results for the epsilon indicator and the continuous problems (MOP1–4, MOP6, ZDT1–3) are shown in Table 6.4.

When the hypotheses are rejected, it counts in favour of the MOO CEM algorithm, and since two indicators are used, the MOO CEM is only considered superior to the Matlab® algorithm when both hypotheses for a given problem instance are rejected. When one hypothesis is rejected, the outcome is considered inconclusive for that problem instance, and when both hypotheses are not rejected, the MOO CEM algorithm is considered inferior for that problem instance.

The epsilon indicator was found to be zero for all the discrete problems, since both algorithms found at least one coordinate in the objective space that coincided with the exact coordinate on the Pareto front. This indicator is therefore not considered for the discrete problems. The results of the hypothesis tests for the hyperareas of the discrete problems are shown in Table 6.5.

The box-whisker plots for all tests are shown in Appendix C. They are included in the appendix to further support the tabled results. This concludes the presentation of the MOO CEM and Matlab® MOO GA comparison. Next, the MOO CEM algorithm is compared to another commercial optimiser using two test problems, followed by test conclusions.

6.5 Comparison between the MOO CEM algorithm and OptQuest®

OptQuest® by Optek Systems Inc. (www.opttek.com, cited on 10 August 2012) is a modern, powerful commercially available optimisation suite. It is included as third-party software by simulation packages like ProModel, Arena and the latest simulation software, Simio of Simio LLC (www.simio.com, cited on 10 August 2012). The version of OptQuest® included with Simio can perform MOO. OptQuest® uses a number of metaheuristics for optimisation, including scatter search, tabu search, and neural networks. The latter is used to screen out trial solutions that are predicted to have inferior solution values, thus reducing the number of objective function evaluations. OptQuest® uses two other prediction technologies,

		MOO CEM				Matlab® MOO GA				Search	Number of
Test problem	Reference HA	Mean HA	h_w	Mean epsilon	h_w	Mean HA	h_w	Mean epsilon	h_w	space size limit	generations allowed
MOP1	13.421	12.832	0.087	$< 10^{-3}$	$< 10^{-3}$	13.4	0.001	$< 10^{-3}$	$< 10^{-3}$	10 000	100
MOP2	0.332	0.322	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$	0.279	0.003	$< 10^{-3}$	$< 10^{-3}$	10 000	100
MOP3	360.919	357.9	0.038	$< 10^{-3}$	$< 10^{-3}$	337.231	3.484	$< 10^{-3}$	$< 10^{-3}$	10 000	100
MOP4	26.199	25.567	0.014	$< 10^{-3}$	$< 10^{-3}$	22.835	0.237	$< 10^{-3}$	$< 10^{-3}$	10 000	100
MOP6	0.515	0.514	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$	0.507	0.002	$< 10^{-3}$	$< 10^{-3}$	10 000	100
ZDT1	0.668	0.587	0.004	$< 10^{-3}$	$< 10^{-3}$	0.002	0.001	$< 10^{-3}$	$< 10^{-3}$	10 000	100
ZDT2	0.334	0.254	0.006	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$	$< 10^{-3}$	10 000	100
ZDT3	0.79	0.756	0.002	$< 10^{-3}$	$< 10^{-3}$	0.032	0.003	$< 10^{-3}$	$< 10^{-3}$	10 000	100
BAP1	3.282	3.141	0.008	$< 10^{-3}$	$< 10^{-3}$	3.065	0.011	$< 10^{-3}$	$< 10^{-3}$	150	6
BAP2	6.327	6.03	0.017	$< 10^{-3}$	$< 10^{-3}$	5.749	0.028	$< 10^{-3}$	$< 10^{-3}$	300	12
BAP3	12.038	11.383	0.029	$< 10^{-3}$	$< 10^{-3}$	9.73	0.074	$< 10^{-3}$	$< 10^{-3}$	700	28
BAP4	5.642	5.032	0.028	$< 10^{-3}$	$< 10^{-3}$	4.487	0.032	$< 10^{-3}$	$< 10^{-3}$	850	34
BAP5	10.431	10.007	0.034	$< 10^{-3}$	$< 10^{-3}$	9.819	0.04	$< 10^{-3}$	$< 10^{-3}$	425	17
BAP6	22.64	20.504	0.106	$< 10^{-3}$	$< 10^{-3}$	19.613	0.105	$< 10^{-3}$	$< 10^{-3}$	600	24
BAP7	29.373	25.019	0.226	$< 10^{-3}$	$< 10^{-3}$	24.194	0.247	$< 10^{-3}$	$< 10^{-3}$	900	36
BAP8	50.168	49.198	0.092	$< 10^{-3}$	$< 10^{-3}$	48.954	0.106	$< 10^{-3}$	$< 10^{-3}$	475	19
BAP9	82.747	78.51	0.323	$< 10^{-3}$	$< 10^{-3}$	74.294	0.264	$< 10^{-3}$	$< 10^{-3}$	225	9
BAP10	133.061	129.101	0.5	$< 10^{-3}$	$< 10^{-3}$	129.033	0.447	$< 10^{-3}$	$< 10^{-3}$	625	25
BAP11	211.166	188.745	1.212	$< 10^{-3}$	$< 10^{-3}$	182.723	1.226	$< 10^{-3}$	$< 10^{-3}$	575	23
BAP12	180.773	178.979	0.108	$< 10^{-3}$	$< 10^{-3}$	178.203	0.127	$< 10^{-3}$	$< 10^{-3}$	625	25
BAP13	291.135	269.653	2.027	$< 10^{-3}$	$< 10^{-3}$	271.48	2.03	$< 10^{-3}$	$< 10^{-3}$	250	10
BAP14	3.214	3.162	0.005	$< 10^{-3}$	$< 10^{-3}$	3.093	0.011	$< 10^{-3}$	$< 10^{-3}$	150	6
BAP15	7.442	6.705	0.031	$< 10^{-3}$	$< 10^{-3}$	6.566	0.026	$< 10^{-3}$	$< 10^{-3}$	250	10
BAP16	11.298	10.426	0.04	$< 10^{-3}$	$< 10^{-3}$	10.405	0.029	$< 10^{-3}$	$< 10^{-3}$	1 150	46
BAP17	36.795	35.495	0.085	$< 10^{-3}$	$< 10^{-3}$	35.231	0.095	$< 10^{-3}$	$< 10^{-3}$	325	13
BAP18	15.67	15.107	0.031	$< 10^{-3}$	$< 10^{-3}$	15.29	0.034	$< 10^{-3}$	$< 10^{-3}$	100	4
BAP19	12.71	11.181	0.071	$< 10^{-3}$	$< 10^{-3}$	11.081	0.069	$< 10^{-3}$	$< 10^{-3}$	150	6
BAP20	33.276	25.086	0.267	$< 10^{-3}$	$< 10^{-3}$	23.936	0.264	$< 10^{-3}$	$< 10^{-3}$	1 050	42
BAP21	210.274	199.892	0.623	$< 10^{-3}$	$< 10^{-3}$	205.774	0.585	$< 10^{-3}$	$< 10^{-3}$	400	16
BAP22	3179.534	3134.987	3.832	$< 10^{-3}$	$< 10^{-3}$	3113.062	8.355	$< 10^{-3}$	$< 10^{-3}$	1 500	60
BAP23	10.109	9.906	0.012	$< 10^{-3}$	$< 10^{-3}$	9.875	0.013	$< 10^{-3}$	$< 10^{-3}$	325	13
INVN	131 066.389	124 366.789	221.143	$< 10^{-3}$	$< 10^{-3}$	114 391.546	889.957	$< 10^{-3}$	$< 10^{-3}$	300	12
RMS	4 619.760	4 532.293	5.786	$< 10^{-3}$	$< 10^{-3}$	3 841.628	33.156	0.001	$< 10^{-3}$	2 900	116
COGas	9 621 731.086	1 332 233.849	3 496.192	$< 10^{-3}$	$< 10^{-3}$	1 274 349.295	4 349.576	$< 10^{-3}$	$< 10^{-3}$	850	34

Table 6.2: Mean values and 95% confidence interval half-widths for the hyperarea and epsilon indicator.

	Hyperarea				
Test problem	p-value	CI low	CI upper	t-stat	Outcome
MOP1	1	-0.640	∞	-12.841	No reject
MOP2	0	0.041	∞	28.395	Reject
MOP3	0	17.680	∞	12.025	Reject
MOP4	0	2.533	∞	22.615	Reject
MOP6	0	0.005	∞	7.957	Reject
ZDT1	0	0.581	∞	258.253	Reject
ZDT2	0	0.249	∞	84.818	Reject
ZDT3	0	0.721	∞	389.436	Reject

Table 6.3: Outcomes of the hypothesis tests for the hyperarea indicator: continuous problems.

	Epsilon indicator				
Test problem	p-value	CI low	CI upper	t-stat	Outcome
MOP1	0	0	∞	4.865	No reject
MOP2	1	0	∞	-30.201	Reject
MOP3	1	0	∞	-26.991	Reject
MOP4	1	0	∞	-25.790	Reject
MOP6	1	0	∞	-17.238	Reject
ZDT1	1	0	∞	-18.360	Reject
ZDT2	1	0	∞	-12.849	Reject
ZDT3	1	0	∞	-17.856	Reject

Table 6.4: Outcomes of the hypothesis tests for the epsilon indicator: continuous problems.

namely satisfiability data mining (Sat-DM) and Markov Blankets (MB). These also contribute to reducing the number of objective function evaluations. OptQuest® speeds up optimisation by utilising parallel processing, for example four possible solutions are developed simultaneously on a current laptop with four cores. For detail see Laguna (2011).

In this section, the MOO CEM algorithm is compared to the MOO capability of OptQuest® using two problems studied previously. These are the (s, S) inventory problem and BAP17. Although OptQuest® already incorporates the CEM, it is not clear whether it is in the MOO context or not. Two data sets were created for each test problem. The one data set was created using Arena and the MOO CEM

	Hyperarea				
Test problem	p-value	CI low	CI upper	t-stat	Outcome
BAP1	0	0.065	∞	11.015	Reject
BAP2	0	0.254	∞	16.770	Reject
BAP3	0	1.586	∞	40.536	Reject
BAP4	0	0.508	∞	24.798	Reject
BAP5	0	0.145	∞	7.095	Reject
BAP6	0	0.766	∞	11.716	Reject
BAP7	0	0.545	∞	4.848	Reject
BAP8	0	0.126	∞	3.399	Reject
BAP9	0	3.867	∞	19.847	Reject
BAP10	0.421	-0.494	∞	0.200	No reject
BAP11	0	4.576	∞	6.855	Reject
BAP12	0	0.636	∞	9.102	Reject
BAP13	0.894	-4.233	∞	-1.250	No reject
BAP14	0	0.059	∞	11.020	Reject
BAP15	0	0.105	∞	6.757	Reject
BAP16	0.199	-0.020	∞	0.846	No reject
BAP17	0	0.157	∞	4.075	Reject
BAP18	1	-0.222	∞	-7.839	No reject
BAP19	0.024	0.017	∞	1.986	Reject
BAP20	0	0.835	∞	6.002	Reject
BAP21	1	-6.598	∞	-13.508	No reject
BAP22	0	14.215	∞	4.681	Reject
BAP23	0	0.016	∞	3.404	Reject
INVN	0	9 205.952	∞	21.346	Reject
RMS	0	662.429	∞	40.269	Reject
COGas	0.024	0.017	∞	1.986	Reject

Table 6.5: Outcomes of the hypothesis tests for the hyperarea indicator: discrete problems.

in Matlab® as before, while the other was created using Simio and OptQuest® because the MOO version of OptQuest® was not available in the author's version of Arena. It was attempted to ensure a fair experiment with no setting giving one optimiser an advantage. To compare the two optimisers, the hyperarea was used as before, and 50 pseudo-independent hyperarea values were created via an optimisation trial by each optimiser. It was necessary to change the random streams in Simio at each stochastic point for each of the 50 trials to ensure that Simio creates a different solution per trial. The experiments are described next.

6.5.1 Experimental setup for the MOO CEM and OptQuest® comparison with the inventory problem

The reorder point s and reorder quantity S will again be determined to minimise inventory cost and maximise the service level, as described in Subsection 5.1. The maximum number of evaluations is 1 140, while five replications are allowed per evaluation. There are three stochastic points in this model: the arrival rate, the demand and the lead time, and these all require different random number streams. The random number stream indices for Simio were randomly selected, and are shown in Table 6.6. This was not necessary in Arena, as the optimisation trials were executed using a programmed loop.

6.5.2 Experimental results for the MOO CEM and OptQuest® comparison with the inventory problem

The hyperareas of the inventory problem comparison between the MOO CEM algorithm and OptQuest® are shown in Table 6.6.

A hypothesis test for the difference of the hyperareas was performed as before, and the formulation of the hypothesis test is (as before)

$$\begin{aligned} H_0 &: m_{CI_H} \leq m_{OI_H} \\ H_1 &: m_{CI_H} > m_{OI_H} \end{aligned}$$

where m_{CI_H} is the mean of the hyperareas produced by the MOO CEM algorithm, and m_{OI_H} is the mean of the hyperareas produced by OptQuest®.

The outcome is shown in Table 6.7.

It follows from the values in the table that the MOO CEM algorithm in general has created larger hyperareas, and is thus superior to OptQuest®, *for the inventory problem*. A supporting box plot is presented in Figure 6.2.

For perspective, two extreme approximate fronts, one representing the smallest and the other the largest hyperarea, are shown in Figure 6.3 for both optimisers. The Pareto front for this problem is also shown (see Subsection 5.1).

6.5.3 Experimental setup for the MOO CEM and OptQuest® comparison with BAP17

In this experiment, the number of buffers must be optimised as in BAP17, and the objectives are chosen as minimising the average WIP and maximising the throughput rate. The problem limits of (0,0,0,0) to (35, 30, 25, 25) were set for

Trial	Hyperarea MOO CEM	Hyperarea OptQuest®	Random stream index			Trial	Hyperarea MOO CEM	Hyperarea OptQuest®	Random stream index		
			Arrivals	Demand	Lead time				Arrivals	Demand	Lead time
1	66 320.2	72 021.9	4	10	2	26	73 748.8	45 552.9	6	7	5
2	72 018.0	69 141.2	4	2	8	27	59 734.2	75 716.6	1	9	6
3	66 720.8	36 123.2	8	10	6	28	65 664.6	51 070.7	6	6	5
4	69 629.5	44 708.0	8	7	5	29	74 284.0	63 815.9	1	9	7
5	72 015.0	80 289.5	4	9	7	30	71 134.4	36 863.2	4	1	4
6	66 744.9	38 680.0	2	7	10	31	70 972.0	52 135.2	6	1	7
7	71 307.9	63 051.5	10	4	8	32	61 738.7	59 705.8	2	4	9
8	81 194.6	76 551.4	6	6	7	33	68 137.2	64 538.0	2	2	9
9	66 157.8	82 884.9	5	3	1	34	59 825.5	47 447.4	3	1	6
10	66 086.9	50 676.5	1	1	3	35	66 584.2	74 370.4	6	9	1
11	70 249.4	83 892.8	8	5	3	36	70 986.7	105 930.4	6	2	1
12	72 154.9	43 262.9	4	4	9	37	69 371.1	83 499.7	3	5	10
13	64 273.7	52 160.7	2	7	9	38	65 656.7	63 751.6	2	8	3
14	69 222.4	72 044.0	4	6	2	39	75 492.1	67 082.5	1	8	5
15	72 989.9	89 946.9	9	8	5	40	61 738.3	59 962.3	4	9	9
16	64 258.6	34 982.8	8	9	2	41	67 845.6	78 194.8	4	3	4
17	70 538.8	79 586.7	1	9	1	42	74 447.3	72 175.4	5	5	10
18	65 652.5	16 569.3	8	10	5	43	75 186.4	35 552.7	4	2	6
19	62 355.0	74 763.4	9	8	4	44	68 902.5	80 522.1	9	3	5
20	72 297.5	81 531.7	1	5	2	45	64 595.7	28 514.7	10	3	5
21	76 011.5	67 640.0	7	7	3	46	71 832.6	68 598.1	1	1	10
22	76 390.5	79 703.2	2	8	1	47	78 234.4	97 181.7	5	9	1
23	67 804.0	26 269.6	4	10	9	48	71 891.4	52 447.4	8	6	7
24	67 695.5	61 493.0	8	1	8	49	67 956.9	64 524.0	5	3	6
25	76 388.2	66 859.7	1	8	8	50	72 843.1	85 174.3	8	7	4

Table 6.6: Hyperareas for the MOO CEM and OptQuest® comparison using the inventory problem. The Simio random number stream indices per trial are included.

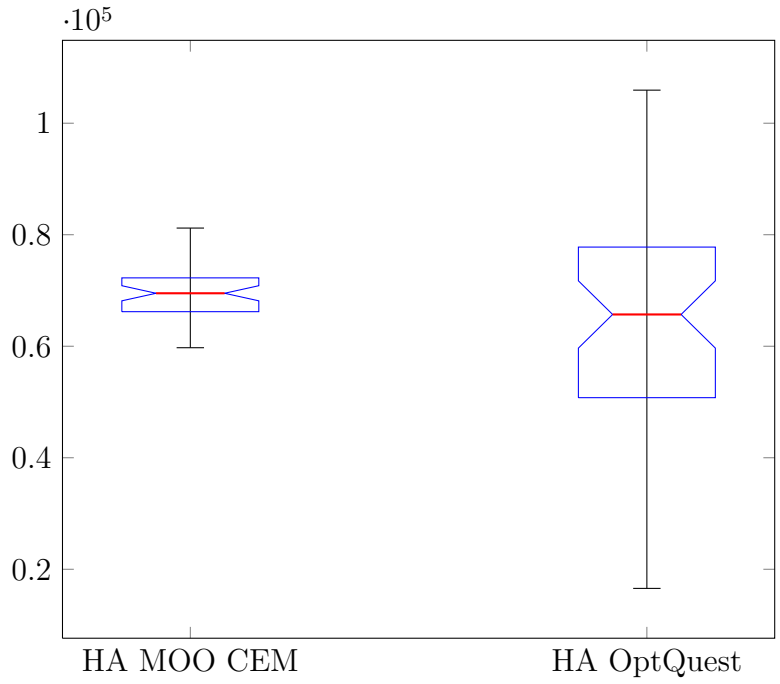

Figure 6.2: Box plot for the hyperarea comparison of the MOO CEM algorithm and OptQuest® using the inventory problem.

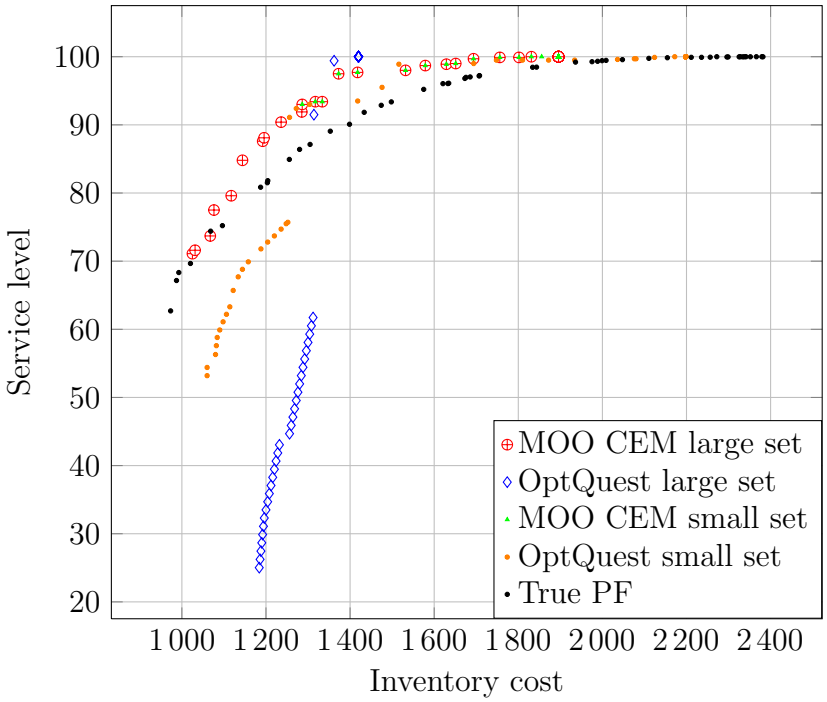

Figure 6.3: Best and worst approximation fronts found by the MOO CEM algorithm and OptQuest, for the inventory problem.

	Hyperarea				
Test problem	p-value	CI lower	CI upper	t-stat	Outcome
INVN: MOO CEM and OptQuest®	0.0145	1.605×10^3	∞	2.2423	Reject H_0

Table 6.7: Outcome of the hypothesis test for the hyperarea indicator of the inventory problem: MOO CEM and OptQuest®.

both optimisers as before. Because the aim is to obtain good solutions with few evaluations, the maximum number of evaluations was set to 100. For this, the MOO CEM algorithm was assigned a population size of 20, allowing five iterations. The "Max Scenarios" property of OptQuest® was set to 100, and the minimum and maximum number of simulation replications were set to 10.

The random number stream indices were randomly selected for the processing time distributions of each of the five machines, and the random number stream indices for these five stochastic points in BAP17 are shown in Table 6.8.

6.5.4 Experimental results for the MOO CEM and OptQuest® comparison with BAP17

The hyperareas of the comparison between the MOO CEM algorithm and OptQuest® for BAP17 are shown in Table 6.8 and the outcome of the hypothesis test for the difference of the hyperareas is shown in Table 6.9.

It follows from the values in Table 6.9 that OptQuest® in general has created larger hyperareas, and is thus superior to the MOO CEM algorithm, *for BAP17*. A supporting box plot is presented in Figure 6.4.

It should be noted that the search strategy of OptQuest® always includes the user-specified extreme search limits, while the MOO CEM algorithm has to find these. It can thus be expected that the hyperarea of OptQuest® will be larger due to the inclusion of the extremes. In the BAP17, the lowest WIP of 3.2 was achieved when all buffer sizes were set equal to zero ($n_i = 0$), and the highest throughput rate for the maximum buffer sizes is 0.904. Two extreme approximate fronts, one representing the smallest and the other the largest hyperarea, are shown in Figure 6.5 for both optimisers.

In the next section, the results of the assessment experiment are discussed and conclusions are drawn.

Trial	Hyperarea MOO CEM	Hyperarea OptQuest	Random stream index					Trial	Hyperarea MOO CEM	Hyperarea OptQuest	Random stream index				
			M1	M2	M3	M4	M5				M1	M2	M3	M4	M5
1	1 129.5	2 156.7	2	7	4	1	3	26	1 579.0	1 808.6	6	2	9	9	7
2	1 143.3	1 295.0	1	1	1	1	1	27	1 988.8	1 505.8	9	2	6	3	7
3	1 566.8	1 161.5	4	3	5	1	2	28	1 456.9	2 042.7	2	2	2	3	7
4	1 577.9	2 145.7	10	3	4	2	5	29	1 332.0	1 681.4	2	10	7	7	3
5	1 660.6	1 993.9	8	9	3	4	2	30	1 239.6	1 782.1	6	1	8	1	9
6	1 756.0	1 956.5	1	10	6	5	2	31	1 416.7	1 796.4	1	7	9	3	6
7	1 572.0	2 118.9	6	2	4	2	10	32	1 222.5	1 491.0	6	10	9	1	3
8	1 397.4	1 993.8	1	5	2	3	9	33	1 830.2	1 267.7	7	10	3	6	4
9	1 429.0	1 499.9	4	1	10	8	7	34	1 621.3	1 132.0	3	5	3	8	5
10	1 199.5	1 456.0	7	7	3	10	7	35	1 737.6	1 055.2	4	2	6	8	10
11	1 925.5	1 369.2	7	1	3	9	6	36	1 432.7	1 489.6	1	8	5	5	1
12	1 702.1	2 100.5	1	9	1	9	4	37	915.6	2 350.3	1	1	5	6	10
13	1 547.5	1 579.8	3	9	9	3	8	38	1 520.0	1 594.2	1	7	2	9	5
14	925.8	1 952.0	7	9	7	5	4	39	1 564.3	1 530.0	1	8	8	10	10
15	1 595.2	1 561.4	6	5	5	2	1	40	1 341.3	1 517.2	8	2	6	10	4
16	1 632.1	1 837.9	3	10	6	6	1	41	1 357.0	2 192.6	5	10	4	8	5
17	1 472.0	1 337.2	3	8	8	9	7	42	1 607.3	1 646.9	2	6	4	4	5
18	1 585.2	1 760.3	6	8	4	9	9	43	1 742.5	2 060.8	8	5	1	1	4
19	1 626.5	1 610.1	8	5	6	10	1	44	1 375.1	1 216.7	3	7	7	9	4
20	1 539.2	2 057.8	5	5	3	4	8	45	966.1	1 467.8	6	6	2	9	5
21	1 587.1	1 838.6	7	7	7	5	3	46	1 247.3	1 717.0	9	2	9	10	9
22	1 581.1	1 829.2	4	3	4	4	7	47	1 593.1	1 366.0	8	7	2	6	1
23	1 436.4	1 003.1	5	7	9	9	9	48	1 646.5	836.8	10	10	2	10	10
24	1 617.9	1 054.1	10	1	8	8	7	49	1 519.8	1 760.2	1	2	4	7	9
25	1 608.1	1 437.2	8	5	7	4	5	50	1 229.6	1 222.2	4	5	8	4	7

Table 6.8: Hyperareas for the MOO CEM and OptQuest® comparison using BAP17. The Simio random number stream indices per trial are included.

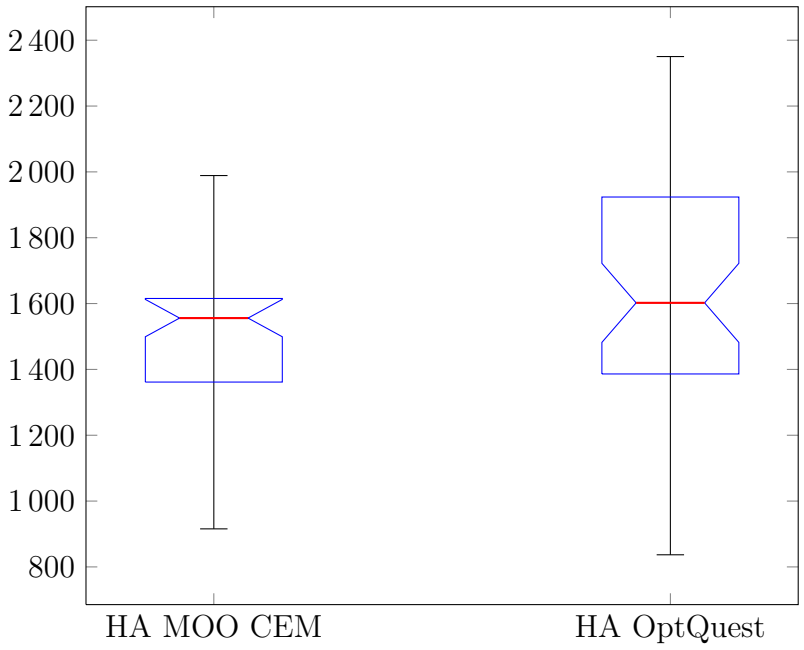

Figure 6.4: Box plot for the hyperarea comparison of the MOO CEM algorithm and OptQuest® using BAP17.

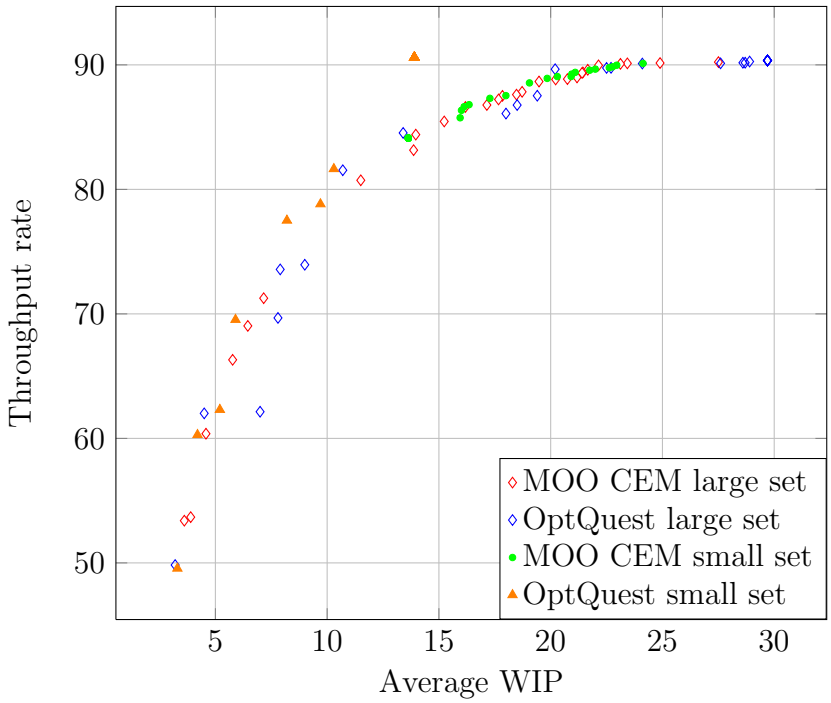

Figure 6.5: Best and worst approximation fronts found by the MOO CEM algorithm and OptQuest, for BAP17.

	Hyperarea				
Test problem	p-value	CI lower	CI upper	t-stat	Outcome
BAP17: MOO CEM and OptQuest	0.9915	-247.17	∞	-2.434	Do not reject H_0

Table 6.9: Outcome of the hypothesis test for the hyperarea indicator of BAP17: MOO CEM and OptQuest®.

6.6 Conclusions: Algorithm performance quality assessment

An experiment was described in this chapter for assessing the performance solution quality of the MOO CEM algorithm. This assessment was performed using 34 MOO problems while creating values for two recognised qualitative quality indicators (hyperarea and additive epsilon indicator) by the MOO CEM algorithm and the commercial MOO GA of Matlab®. The presumption was that the performance quality of the MOO CEM should be comparable to that of at least one other similar algorithm, and a commercial algorithm was selected for this comparison. Also, it was attempted to follow the best-known and accepted experimental procedure to ensure that the assessment and its outcomes are sound; hence the application of the work by Knowles *et al.* (2006).

The MOO CEM algorithm found better quality indicator values than the Matlab® MOGA for both the hyperarea and epsilon indicators in all the continuous cases, except for MOP1. The fact that the outcomes are consistent for both indicators, i.e. the hypotheses are rejected for both or not rejected for any one is encouraging, as different outcome pairs (for example Reject/No rejection) will mean that the approximation sets are incomparable.

In the discrete problem set, the MOO CEM algorithm found better hyperarea indicator values for 21 out of 26 problems. For BAP10, BAP13, BAP16, BAP18 and BAP21 the two algorithms performed similarly or the Matlab® algorithm was superior. It is concluded that the MOO CEM algorithm has merit in MOO applications of both deterministic, continuous problems and discrete, stochastic problems. One cannot conclude that the Matlab® MOO GA will always be outperformed – the conclusion made in this study pertains only to the problems studied and in the context of the quality indicators used. It is possible that

different option settings for the Matlab® algorithm may lead to superiority over the MOO CEM algorithm. However, changes of option settings were investigated during informal, unreported experiments, but that did not lead to better results in favour of the Matlab® MOO GA. It became clear that the Matlab® MOO GA requires more evaluations (in the order of 20 000 or more), and when it was allowed more freedom, it found Pareto sets of high quality. It is also necessary to mention the "Free leftovers" theorem of Corne & Knowles (2003): *over the space of permutation problems, every algorithm has some companion algorithm(s) which it outperforms, according to a certain well-behaved metric, when comparative performance is summed over all problems in the space.* However, the MOO CEM algorithm is proposed in this research as a viable option, since it was not the *companion algorithm* that was being outperformed in the MOO CEM – Matlab® MOO GA relationship.

In a final experiment, the MOO CEM algorithm was compared to the commercial optimisation package OptQuest® using two problems. OptQuest® outperformed the MOO CEM algorithm with the problem instance BAP17, although the MOO CEM algorithm produced denser approximate Pareto fronts. However, the MOO CEM algorithm created better hyperareas than OptQuest® when optimising the (s, S) inventory problem. The MOO CEM algorithm performed well against the Matlab® MOO GA, but more problems must be studied to compare it against OptQuest® for a conclusive comparison.

This concludes the description of the experimental work and the results. Next, final conclusions and recommendations of the overall research project are presented.

CHAPTER 7

RESEARCH SUMMARY AND CONCLUSIONS

The research project is summarised in this chapter, and research conclusions are presented. Suggestions for further research are listed, and a brief philosophical view is given to conclude the project.

7.1 Project summary and conclusions

The research hypothesis was stated in Section 1.2. This research aimed to find an algorithm which could perform multi-objective optimisation of dynamic, stochastic problems while utilising an economic computational burden. Problems of this nature often require time-consuming simulations, since objective function values must be estimated via statistical procedures that require many simulation replications.

The CEM for optimisation (Rubinstein & Kroese, 2004) has been extended to MOO in this research, which is considered the main contribution to the body of knowledge. The author came upon this method during a survey of scholarly works on MOO, and there was convincing evidence that it converges fast when applied to single-objective optimisation problems. Since it is also a relatively simple method, both in principle and to program, the research question arose: "Will the CEM also allow MOO problems to converge fast towards quality approximate Pareto fronts?"

Before investigating this question, the author provided a scholarly overview in **Chapter 2** of the recent work in the field of MOO, which was deliberately brief. It was attempted to provide pointers to MOO applications of the many available MOO metaheuristics and MOO algorithms, and to provide a reference to a recent survey of these, where possible. The theory behind the CEM for optimisation was presented as a stand-alone part of the dissertation in **Chapter 3**.

The research on the MOO CEM algorithm followed a deliberate path: after development, it was assessed using standard benchmark problems, which were all deterministic and continuous (Section 4.2, **Chapter 4**). These problems pose challenges, by design, to an optimisation algorithm, as the number of decision variables vary, the decision space and the solution space are both often discontinuous, and the problems are scalable and rotatable. The MOO CEM algorithm performed satisfactorily with respect to these problems. Since the ultimate aim was to apply it to dynamic, stochastic problems, the (s, S) inventory problem was studied in conjunction with computer simulation (Section 5.1, **Chapter 5**). The latter is an integral part of the optimisation problem, as it is used to estimate objective function values of non-closed, dynamic, stochastic problems. The results of these assessments were published in Bekker & Aldrich (2010) to establish a research base, and to inform the international research community of the research intent.

The focus then moved towards computationally more demanding dynamic, stochastic problems, and the classical buffer allocation problem (BAP) was selected for further application of the MOO CEM algorithm (Section 5.2, **Chapter 5**). This is a combinatorial problem and hence the solution space quickly grows in size as the problem dimensions increase. Simulation of a typical scenario took a few minutes on current state-of-the art laptop computers, so that seeking best solutions to problems with many possible solutions became computationally prohibitive. The MOO CEM algorithm was applied to many variants of this problem, i.e. various numbers of machines, various numbers of niches, linear topologies and a network topology, exponential processing times and also Erlang processing times. A manuscript reporting on this work was accepted for publication by the *International Journal of Simulation Modelling* (Bekker, 2012).

The development of an autonomous, reconfigurable manufacturing system is a practical problem under study by a national research group driven by the Department of Mechanical and Mechatronic Engineering at Stellenbosch University. In support to this study, Du Preez (2011) provided a population of solutions for MOO, and the MOO CEM algorithm could also be tested with respect to that study. An engineering design problem with deterministic, continuous decision variables was analysed as a first practical problem. In this problem, a polymer extruder design was optimised. A final problem analysis focused on decision support in a case study at a heavy minerals mining plant. The aim of this study was to determine whether or not it is feasible to recycle CO gas, which is a by-product

of the mineral smelting operation. The study showed that a useful approximate Pareto set of solutions can be found by means of the MOO CEM algorithm, and that acquisition cost of methane gas can potentially be reduced, while less CO_2 is produced.

During the assessments of the MOO CEM algorithm in the contexts of the problems mentioned above, values for four simple Pareto non-compliant quality indicators were calculated for all Pareto approximation solutions. Three of these indicators assume that the true Pareto fronts are known. Their estimated values depend on the proximity of the generated solution set as well as its spread. These indicators were used for assessment of the MOO CEM algorithm during development.

A second phase of assessment was conducted by comparing the quality performance of the algorithm against that of a commercial package. For this purpose, the multi-objective genetic algorithm of Matlab® was selected, and all previously studied problems were again assessed while using the Pareto compliant hyperarea and epsilon quality indicators (**Chapter 6**). The assessment experiment was conducted according to guidelines by Knowles *et al.* (2006), and in 28 out of 34 problems, the MOO CEM algorithm performed better than the Matlab® MOO GA. The author respects the work and product developments of others, so this claim is made with circumspection, as was discussed in Section 6.6, **Chapter 6**. Detailed results of the experiments are presented in **Appendix C**.

In a final experiment, the performance of the MOO CEM algorithm was compared to that of OptQuest®, a powerful, commercial optimisation package. The (s, S) inventory problem and BAP17 were used in two experiments. The MOO CEM algorithm created significantly larger hyperareas in the case of the inventory problem. In the case of BAP17 the MOO CEM algorithm generated denser approximation sets, but the OptQuest® hyperarea was superior. The MOO CEM algorithm thus performed better than the Matlab® MOO GA, but the outcome of its performance against OptQuest® is inconclusive.

An implementation guideline of the proposed MOO CEM algorithm in conjunction with a computer simulation product is given in **Appendix D**.

To summarise, the research aim and objectives set out in **Chapter 1** were achieved because:

1. The computational burden when developing an approximate Pareto front using the MOO CEM algorithm is generally lower compared to the algorithm of an established commercial package, also in the case of dynamic, stochastic problems.

2. The approximate Pareto fronts obtained for the test problems are effective and efficient, considering the appropriate quality indicators.

3. The MOO CEM algorithm is applicable to both discrete and continuous multi-objective optimisation problems.

The summary and conclusions lead to the following suggestions for further research.

7.2 Further research

The research presented in this dissertation is not complete and a few suggestions for further research are as follows:

1. Apply the MOO CEM algorithm to other engineering problems, typically in continuous manufacturing and control of processes. In process control, adjustment of control parameters in real time is important, and it is possible to investigate whether or not the MOO CEM algorithm can adjust such parameters quickly, compared to neural networks, for example.

2. All the objectives in this research were limited to two objectives, but the MOO CEM algorithm should be assessed with respect to problems of higher objective dimensions.

3. Continuous, constrained multi-objective problems should be optimised using the MOO CEM algorithm and the results should be assessed.

4. The correlation of solution sets should be investigated to improve search efficiency of the MOO CEM algorithm. A covariance structure similar to that of the MO-CMA-ES reported by Igel *et al.* (2007) may be considered.

5. The MOO CEM algorithm may be compared to the many other MOO algorithms available, for example SPEA and PAES, as well as metaheuristics like simulated annealing and ant-colony optimisation.

In the next section, a brief philosophical view is presented to conclude the dissertation.

7.3 Philosophy

In this final section of the dissertation, the author wishes to share some reflections on optimisation of engineered (man-made) systems. A system consists of interdependent objects that cooperate to achieve and sustain a goal that is defined before conception (man-made systems) or inferred after-the-fact (natural systems). A man-made system is usually complex and often imperfect on delivery, but operates satisfactorily. The system environment evolves and often drives the system towards obsolescence, but engineers either replace the existing system with a new one or adapt and improve the existing system so that its reason for existence is justified. Optimisation is such an effort, namely to design a system as well as possible or to improve it.

The Gaia Theory (Lovelock, 2000), in simplistic terms states that the Earth is an organism in equilibrium. Every time the balance is disturbed, the organism adjusts itself to reach equilibrium again. It is argued that a system is also such an organism, and each time we change the system via what we perceive as improvements, a new state of apparent satisfaction is achieved at a cost. New imperfections manifest themselves, and new optimisation challenges arise. Optimisation thus leads to new states of sub-optimality, and the optimisation effort is therefore never complete.

In life, humans also strive for the better, which, in a highly capitalistic and consumer-driven society, means to have more, but this state will never be achieved by most. The "optimal better" is thus not to have more, but to appreciate what one has.

BIBLIOGRAPHY

Ali, H., Shahzad, W. and Khan, F.A. (2011). Energy-efficient clustering in mobile ad-hoc networks using multi-objective particle swarm optimization. *Applied Soft Computing*, **In Press, Accepted Manuscript**. 26

Almeida, L.M. and Ludermir, T.B. (2010). A multi-objective memetic and hybrid methodology for optimizing the parameters and performance of artificial neural networks. *Neurocomputing*, **73**, 1438–1450. 30

Alon, G., Kroese, D., Raviv, T. and Rubinstein, R. (2005). Application of the cross-entropy method to the buffer allocation problem in a simulation-based environment. *Annals of Operations Research*, **134**, 137–151. 44, 45

Anderson, S., Kadirkamanathan, V., Chipperfield, A., Sharifi, V. and Swithenbank, J. (2005). Multi-objective optimization of operational variables in a waste incineration plant. *Computers and Chemical Engineering*, **29**, 1121–1130. 35

Apostolopoulos, T. and Vlachos, A. (2011). Application of the firefly algorithm for solving the economic emissions load dispatch problem. *International Journal of Combinatorics*, **2011**, 23, doi:10.1155/2011/523806. 31

Baesler, F. and Sepúlveda, J.A. (2001). Multi-objective simulation optimization for a cancer treatment center. *Proceedings of the 2001 Winter Simulation Conference*, 1405–1411. 32

Banks, J., ed. (1998). *Handbook of Simulation*. Wiley-Interscience. 2

Bashkar, V., Gupta, S.K. and Ray, A.K. (2001). Multiobjective optimization of an industrial wiped film poly(ethylene terephthalate) reactor: some further insights. *Computers and Chemical Engineering*, **25**, 391–407. 35

Bashyam, S. and Fu, M. (1998). Optimization of (s, S) inventory systems with random lead times and a service level constraint. *Management Science*, **44**, 243–256. 80

Beausoleil, R.P. (2006). "MOSS" multiobjective scatter search applied to non-linear multiple criteria optimization. *European Journal of Operational Research*, **169**, 426–449, Feature Cluster on Scatter Search Methods for Optimization. 14

Bekker, J. (2012). Multi-objective buffer space allocation with the cross-entropy method. *International Journal of Simulation Modelling*, accepted. 117, 150

Bekker, J. and Aldrich, C. (2010). The cross-entropy method in multi-objective optimisation: An assessment. *European Journal of Operational Research*, **211**, 112–121. 54, 150

Bernardino, H. and Barbosa, H. (2009). Artificial immune systems for optimization. In R. Chiong, ed., *Nature-Inspired Algorithms for Optimisation*, vol. 193 of *Studies in Computational Intelligence*, 389–411, Springer, Berlin/Heidelberg, 10.1007/978-3-642-00267-0-14. 28

Bersini, H. and Varela, F.J. (1991). The immune recruitment mechanism: A selective evolutionary strategy. In R.K. Belew and L.B. Booker, eds., *Proceedings of the Fourth International Conference on Genetic Algorithms*. 28

Bettonvil, B., Del Castillo, E. and Kleijnen, J.P. (2009). Statistical testing of optimality conditions in multiresponse simulation-based optimization. *European Journal of Operational Research*, **199**, 448–458. 3, 80

Beyer, H.G. and Schwefel, H.P. (2002). Evolution strategies – a comprehensive introduction. *Natural Computing*, **1**, 3–52, 10.1023/A:1015059928466. 28

Blum, C. (2005). Ant colony optimization: Introduction and recent trends. *Physics of Life Reviews*, **2**, 353–373. 25

Bosman, P. and Thierens, D. (2003). The balance between proximity and diversity in multi-objective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, **7**, 174–188. 17

Burke, E.K., Causmaecker, P.D., Maere, G.D., Mulder, J., Paelinck, M. and Berghe, G.V. (2010). A multi-objective approach for robust airline scheduling. *Computers & Operations Research*, **37**, 822–832. 31

Campelo, F., Guimares, F. and Igarashi, H. (2007). Overview of artificial immune systems for multi-objective optimization. In S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu and T. Murata, eds., *Evolutionary Multi-Criterion Optimization*, vol. 4403 of *Lecture Notes in Computer Science*, 937–951, Springer, Berlin/Heidelberg, 10.1007/978-3-540-70928-2-69. 28

Castro-Gutierrez, J., Landa-Silva, D. and Moreno Perez, J. (2011). Nature of real-world multi-objective vehicle routing with evolutionary algorithms. In *2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 257–264, IEEE. 69, 71, 72, 74, 75

Chen, A. and Chyu, C. (2010). Applying memetic algorithm in multi-objective resource allocation among competing projects. *Journal of Software*, **5**, 802–9. 30

Chiam, S.C., Tan, K.C. and Mamun, A.M. (2009). A memetic model of evolutionary PSO for computational finance applications. *Expert Systems with Applications*, **36**, 3695–3711. 31

Chica, M., Cordón, O. and Damas, S. (2011). An advanced multiobjective genetic algorithm design for the time and space assembly line balancing problem. *Computers & Industrial Engineering*, **61**, 103–117. 34

Cinnella, P. and Hercus, S. (2010). Robust optimization of dense gas flows under uncertain operating conditions. *Computers & Fluids*, **39**, 1893–1908. 38

Coello Coello, C.A. (2009). Evolutionary multi-objective optimization: some current research trends and topics that remain to be explored. *Frontiers of Computer Science in China*, **3**, 18–30, 10.1007/s11704-009-0005-7. 13, 14, 15, 16, 39, 61

Coello Coello, C.A. and Corts, N.C. (2005). Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines*, **6**, 163–190, 10.1007/s10710-005-6164-x. 28

Coello Coello, C.A., Pulido, G. and Lechuga, M. (2004). Handling multiple objectives with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, **8**, 256–279. 15

Coello Coello, C.A., Lamont, G.B. and Van Veldhuizen, D.A. (2007). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2nd edn. 4, 11, 12, 14, 15, 16, 18, 20, 22, 23, 24, 25, 26, 27, 28, 32, 78

Cohen, I., Golany, B. and Shtub, A. (2005). Managing stochastic, finite capacity, multi-project systems through the cross-entropy methodology. *Annals of Operations Research*, **134**, 183–199, 10.1007/s10479-005-5730-1. 51

Corne, D.W. and Knowles, J.D. (2003). No Free Lunch and Free Leftovers Theorems for multiobjective optimisation problems. In C.M. Fonseca, P.J. Fleming, E. Zitzler, K. Deb and L. Thiele, eds., *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, 327–341, Springer. Lecture Notes in Computer Science. Volume 2632, Faro, Portugal. 59, 148

Corne, D.W., Knowles, J.D. and Oates, M.J. (2000). The Pareto Envelope-based Selection Algorithm for multiobjective optimization. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo and H.P. Schwefel, eds., *Proceedings of the Parallel Problem Solving from Nature VI Conference*, 839–848, Springer. Lecture Notes in Computer Science No. 1917, Paris, France. 14

Cromvik, C., Lindroth, P., Patriksson, M. and Strömberg, A.B. (2011). A new robustness index for multi-objective optimization based on a user perspective. Tech. Rep., Chalmers University of Technology, University of Gothenburg, Department of Mathematical Sciences, Division of Mathematics, Chalmers University of Technology, University of Gothenburg. 38

Cruz, F., Duarte, A. and Van Woensel, T. (2008). Buffer allocation in general single-server queuing networks. *Computers & Operations Research*, **35**, 3581–3598, part Special Issue: Topics in Real-time Supply Chain Management. 84, 86, 109

Cruz, F., Van Woensel, T. and Smith, J.M. (2010). Buffer and throughput trade-offs in M/G/1/K queuing networks: A bi-criteria approach. *International Journal of Production Economics*, **125**, 224–234. 84, 87, 93, 113

Dantzig, G. and Ramser, J. (1959). The truck dispatching problem. *Management Science*, 80–91. 68

Das, I. (1999). A preference ordering among various Pareto optimal alternatives. *Structural and Multidisciplinary Optimization*, **18**. 17

De Boer, P.T., Kroese, D.P., Mannor, S. and Rubinstein, R.Y. (2005). A tutorial on the cross-entropy method. *Annals of Operations Research*, **134(1)**, 19–67. 72

Deb, K. (2001). *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley. 4, 14

Deb, K. and Jain, S. (2002). Running performance metrics for evolutionary multi-objective optimization. Tech. Rep. 2002004, Indian Institute of Technology. 21, 22

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6**, 182–197. 14, 36, 39, 69, 87, 93, 130

Doerner, K., Merkle, D. and Stützle, T. (2009). Special issue on ant colony optimization. *Swarm Intelligence*, **3**, 1–2, 10.1007/s11721-008-0025-1. 25

Dolgui, A., Eremeev, A., Kolokolov, A. and Sigaev, V. (2002). A genetic algorithm for the allocation of buffer storage capacities in a production line with unreliable machines. *Journal of Mathematical Modelling and Algorithms*, **1**, 89–104, 10.1023/A:1016560109076. 86, 89

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*. Ph.D. thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, (in Italian). 24

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. The MIT Press. 24

D'Souza, R.G.L., Sekaran, K.C. and Kandasamy, A. (2010). Improved NSGA-II based on a novel ranking scheme. *Journal of Computing*, **2**, 91–95. 17

Du Preez, J. (2011). *A Study of Reconfigurable Manufacturing Systems with Computer Simulation*. Master's thesis, Stellenbosch University. 117, 150

Erickson, M., Mayer, A.S. and Horn, J. (1999). Development of a Multi-Objective Optimization Framework for groundwater remediation design using the Niched-Pareto Genetic Algorithm. In *EOS: Transactions of American Geophysical Union*, F840–F843, Am. Geophys. Union. 14

Evans, G.E., Keith, J.M. and Kroese, D.P. (2007). Parallel cross-entropy optimisation. In S. Henderson, B. Biller, M.H. Hsieh, J. Shortle, J. Tew and R. Barton, eds., *Proceedings of the 2007 Winter Simulation Conference*, 2196–2202. 51

Fang, H., Qian, W., Yi-Cheng, T. and Horstemeyer, M.F. (2008). An efficient non-dominated sorting method for evolutionary algorithms. *Evolutionary Computation*, **16**, 355–384. 16, 67, 80

Farina, M., Deb, K. and Amato, P. (2004). Dynamic multiobjective optimization problems: test cases, approximations, and applications. *IEEE Transactions on Evolutionary Computation*, **8**, 425–442. 31

Fonseca, C.M. and Fleming, P.J. (1993). Genetic Algorithms for multiobjective optimization: Formulation, discussion and generalization. In S. Forrest, ed., *Proceedings of the Fifth International Conference on Genetic Algorithms*, 416–423, University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers, San Mateo, California. 14

Fonseca, C.M. and Fleming, P.J. (1998). Multiobjective optimization and multiple constraint handling with Evolutionary Algorithms – Part I: A Unified Formulation. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, **28**, 26–37. 35

Frutos, M., Olivera, A.C. and Tohme, F. (2010). A memetic algorithm based on a NSGA-II scheme for the flexible job-shop scheduling problem. *Annals of Operations Research*, **181**, 745–765. 30

Gao, W. and Engell, S. (2005). Iterative set-point optimization of batch chromatography. *Computers and Chemical Engineering*, **29**, 1401–1409, eSCAPE-14, The 14th European Symposium on Computer Aided Process Engineering. 37

Garcia-Najera, A. and Bullinaria, J. (2011). An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows. *Computers & Operations Research*, **38**, 287–300. 69

Gaspar-Cunha, A. and Covas, J. (2003). A real-world test problem for EMO algorithms. In C.M. Fonseca, P.J. Fleming, E. Zitzler, L. Thiele and K. Deb, eds., *Evolutionary Multi-Criterion Optimization*, vol. 2632 of *Lecture Notes in Computer Science*, 752–766, Springer Berlin Heidelberg. 119, 120, 121, 122, 123, 124

Geiger, M.J. (2008). A computational study of genetic crossover operators for multi-objective vehicle routing problem with soft time windows. *CoRR*, **abs/0809.0410**. 69

Gendreau, M. and Potvin, J.Y. (2010). Tabu search. In M. Gendreau and J.Y. Potvin, eds., *Handbook of Metaheuristics*, vol. 146 of *International Series in Operations Research & Management Science*, 41–59, Springer US, 2nd edn., 10.1007/978-1-4419-1665-5-2. 23, 24, 25, 28, 32

Gershwin, S. and Schor, J. (2000). Efficient algorithms for buffer space allocation. *Annals of Operations Research*, **93**, 117–144. 86

Ghisu, T., Parks, G., Jarrett, J.P. and Clarkson, P. (2011). Robust design optimization of gas turbine compression systems. *Journal of Propulsion and Power*, **27**, 282–295. 24

Gil, C., Márquez, A., Baños, R., Montoya, M.G. and Gómez, J. (2007). A hybrid method for solving multi-objective global optimization problems. *Journal of Global Optimization*, **38**, 265–281. 3, 12, 15

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, **13**, 533 – 549. 23

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic. 23

Goh, C. and Tan, K. (2007). An investigation on noisy environments in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, **11**, 354 –381. 5, 37

Goh, C., Tan, K., Cheong, C. and Ong, Y. (2010a). An investigation on noise-induced features in robust evolutionary multi-objective optimization. *Expert Systems with Applications*, **37**, 5960–5980. 37, 39

Goh, C., Tan, K., Liu, D. and Chiam, S. (2010b). A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design. *European Journal of Operational Research*, **202**, 42–54. 25

Goldberg, D. (1989). *Genetic Algorithms in search, optimization and machine learning*. Addison-Wesley Publishing Company. 16, 56, 88

Gupta, H. and Deb, K. (2005). Handling constraints in robust multi-objective optimization. In *2005 IEEE Congress on Evolutionary Computation*, vol. 1, 25–32. 37

Hájek, J., Szöllös, A. and Sístek, J. (2010). A new mechanism for maintaining diversity of Pareto archive in multi-objective optimization. *Advances in Engineering Software*, **41**, 1031–1057. 18

Hauman, C. (2012). *Multi-objective Optimisation Case Studies Using the Cross-entropy Method*. Master's thesis, Stellenbosch University. B-2

Hauman, C. and Bekker, J. (2012). Application of the multi-objective cross-entropy method to the vehicle routing problem with soft time windows. *ORiON*, submitted. 68, 73

Heavey, C., Papadopoulos, H.T. and Browne, J. (1993). The throughput rate of multistation unreliable production lines. *European Journal of Operational Research*, **68**, 69–89. 86

Hillier, F.S. and So, K.C. (1991). The effect of machine breakdowns and interstage storage on the performance of production line systems. *International Journal of Production Research*, **29**, 2043–2055. 86

Huband, S., Hingston, P., Barone, L. and While, L. (2006). A review of multi-objective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, **10**, 477–506. 18, 38, 39

Hudson, C., Carruthers, J. and Robinson, A. (2011). A comparison of three population-based optimization techniques for the design of composite sandwich materials. *Journal of Sandwich Structures and Materials*, **13**, 213–235. 25

Igel, C., Hansen, N. and Roth, S. (2007). Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, **15**, 1–28. 18, 29, 39, 133, 152

Iorio, A.W. and Li, X. (2004). Solving rotated multi-objective optimization problems using differential evolution. In *AI 2004: Advances in Artificial Intelligence: 17th Australian Joint Conference on Artificial Intelligence*, 861–872. 27

Isaacs, A., Puttige, V., Ray, T., Smith, W. and Anavatti, S. (2008). Development of a memetic algorithm for dynamic multi-objective optimization and its applications for online neural network modeling of UAVs. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 548–54, IEEE, Piscataway, NJ, USA. 31

Jaeggi, D., Parks, G., Kipouros, T. and Clarkson, P. (2008). The development of a multi-objective tabu search algorithm for continuous optimisation problems. *European Journal of Operational Research*, **185**, 1192–1212. 24

Jahanshahloo, G., Lotfi, F.H. and Izadikhah, M. (2006). Extension of the TOPSIS method for decision-making problems with fuzzy data. *Applied Mathematics and Computation*, **181**, 1544–1551. 3

Jaimes, A., Quintero, L. and Coello Coello, C.A. (2009). Ranking methods in many-objective evolutionary algorithms. In R. Chiong, ed., *Nature-Inspired Algorithms for Optimisation*, vol. 193 of *Studies in Computational Intelligence*, 413–434, Springer, Berlin/Heidelberg. 17

Jozefowiez, N., Semet, F. and Talbi, E. (2008). Multi-objective vehicle routing problems. *European Journal of Operational Research*, **189**, 293–309. 69

Kallehauge, B., Larsen, J., Madsen, O. and Solomon, M. (2005). Vehicle routing problem with time windows. *Column generation*, 67–98. 70, 71

Kelton, D., Sadowski, R. and Sturrock, D. (2007). *Simulation with Arena*. McGraw-Hill, 3rd edn. 82, 89

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimisation. In *IEEE International Conference on Neural Networks*, 1942–1948, IEEE Service Center, Piscataway, New Jersey. 25

Kersting, P. and Zabel, A. (2009). Optimizing NC-tool paths for simultaneous five-axis milling based on multi-population multi-objective evolutionary algorithms. *Advances in Engineering Software*, **40**, 452–463. 32

Kim, S.H. and Nelson, B.L. (2001). A fully sequential procedure for indifference-zone selection in simulation. *ACM Trans. Model. Comput. Simul.*, **11**, 251–273. 3

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, **220**, 671–680. 22

Kleijnen, J.P. and Wan, J. (2007). Optimization of simulated systems: OptQuest and alternatives. *Simulation Modelling Practice and Theory*, **15**, 354–362. 11, 32

Knowles, J. (2006). ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, **10**, 50–66. 6, 15

Knowles, J. and Corne, D. (2000). Approximating the Nondominated Front using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, **8**, 149–172. 14

Knowles, J., Thiele, L. and Zitzler, E. (2006). A tutorial on the performance assessment of stochastic multiobjective optimizers. 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, revised version. 20, 22, 130, 131, 132, 147, 151

Koopmans, T.C. (1951). Analysis of production as an efficient combination of activities. In *Activity Analysis of Production and Allocation, Cowles Commission Monograph*, 13, 33–97, John Wiley & Sons, Inc. 11

Kroese, D.P. (2010). Cross-entropy method. In J.J. Cochran, L.A. Cox, P. Keskinocak, J.P. Kharoufeh and J.C. Smith, eds., *Wiley Encyclopedia of Operations Research and Management Science*, John Wiley & Sons, Inc. 44

Kroese, D.P. and Rubinstein, R.Y. (2005). The cross-entropy method for combinatorial optimization, rare event simulation and neural computation. *Annals of Operations Research*, **134(1)**. 40

Kuhn, H.W. and Tucker, A. (1951). Nonlinear programming. In J. Neyman, ed., *Proceedings of the Second Berkeley Simposium on Mathematical Statistics and Probability*, 481–492, University of California Press. 11

Laguna, M. (2011). OptQuest – Optimization of complex systems. Online, http://www.opttek.com, accessed on 7 November 2012. 139

Lakshmi, K. and Rao, A. (2010). A memetic algorithm for combinatorial problems with multiple objectives. In *2010 2nd International Conference on Advanced Computing (ICoAC)*, 33–41, IEEE, Piscataway, NJ, USA, 2010 2nd International Conference on Advanced Computing (ICoAC), 14–16 December 2010, Chennai, India. 30

Langouët, H., Métivier, L., Sinoquet, D. and Tran, Q.H. (2011). Engine calibration: multi-objective constrained optimization of engine maps. *Optimization and Engineering*, 1–18, 10.1007/s11081-011-9140-8. 29

Laporte, G. (2007). What you should know about the vehicle routing problem. *Naval Research Logistics*, **54**, 811–819. 68

Lau, H., Chan, T., Tsui, W., Chan, F., Ho, G. and Choy, K. (2009). A fuzzy guided multi-objective evolutionary algorithm model for solving transportation problem. *Expert Systems with Applications*, **36**, 8255–8268. 69

Laumanns, M., Thiele, L., Deb, K. and Zitzler, E. (2002). Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary Computation*, **10**, 263–282. 17

Law, A.M. and Kelton, W.D. (2000). *Simulation modeling and analysis*. McGraw-Hill, 3rd edn. 2, 80, 89

Lee, J. (2007). A novel three-phase trajectory informed search methodology for global optimization. *IEEE Transactions on Evolutionary Computation*, **38**, 61–77. 15

Lee, K.M., Hsu, M.R., Chou, J.H. and Guo, C.Y. (2011). Improved differential evolution approach for optimization of surface grinding process. *Expert Systems with Applications*, **38**, 5680–5686. 27

Lee, L.H., Chew, E.P., Teng, S. and Goldsman, D. (2010). Finding the non-dominated Pareto set for multi-objective simulation models. *IIE Transactions*, **42**, 656–674. 6

Leonard, T. (2011). *Comparing airport apron layout designs using computer simulation and the cross-entropy method*. Master's thesis, Stellenbosch University. 51, 52

Leonard, T. and Bekker, J. (2012). The cross-entropy method applied to apron layout design and flight-to-gate assignment at Lanseria International Airport. *The South African Journal of Industrial Engineering*, **Submitted**. 52

Li, C., Zhang, X., Zhang, S. and Suzuki, K. (2009). Environmentally conscious design of chemical processes and products: Multi-optimization method. *Chemical Engineering Research and Design*, **87**, 233–243. 32

Li, H. and Zhang, Q. (2009). Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on evolutionary computation*, **13**, 284–302. 39

Liu, M., Sun, Z., Yan, J. and Kang, J. (2011). An adaptive annealing genetic algorithm for the job-shop planning and scheduling problem. *Expert Systems with Applications*, **38**, 9248–9255. 23

Lovelock, J. (2000). *Gaia: A New Look at Life on Earth*. Oxford University Press, USA. 153

Lü, Q., Bai, Z.H. and Xia, X.Y. (2008). Leader-based parallel cross-entropy algorithm for maximum clique problem. *Journal of Software*, **19**, 2899–2907. 50

Lu, Z., Sun, J. and Butts, K.R. (2009). Linear programming support vector regression with wavelet kernel: A new approach to nonlinear dynamical systems identification. *Mathematics and Computers in Simulation*, **79**, 2051–2063. 11, 87

Luh, G.C. and Chueh, C.H. (2004). Multi-objective optimal design of truss structure with immune algorithm. *Computers & Structures*, **82**, 829–844. 28

Lutz, C.M., Davis, K.R. and Sun, M. (1998). Determining buffer location and size in production lines using tabu search. *European Journal of Operational Research*, **106**, 301–316. 86

Ma, L., Zhang, W., Chablat, D., Bennis, F. and Guillaume, F. (2009). Multi-objective optimisation method for posture prediction and analysis with consideration of fatigue effect and its application case. *Computers & Industrial Engineering*, **57**, 1235–1246. 34

Mahapatra, N. and Maiti, M. (2005). Multi-objective inventory models of multi-items with quality and stock-dependent demand and stochastic deterioration. *AMO – Advanced modeling and optimization*, **7**, 69–84. 35

Malakooti, B. (1991). A multiple criteria decision making approach for the assembly line balancing problem. *International Journal of Production Research*, **29**, 1979–2001. 87

Mariano, C. and Morales, E. (1999). MOAQ an Ant-Q algorithm for multiple objective optimization problems. In W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar and R. Smith, eds., *Genetic and Evolutionary Computing Conference (GECCO 99)*, vol. 1, 894–901, Morgan Kaufmann, San Francisco, California. 24

Mariano, C. and Morales, E. (2000). A new approach for the solution of multiple objective optimization problems based on reinforcement learning. In O. Cairo, L. Sucar and F. Cantu, eds., *MICAI 2000: Advances in Artificial Intelligence*, 212–223, Springer-Verlag, Acapulco, Mexico. 26

Massim, Y., Yalaoui, F., Amodeo, L., Chatelet, E. and Zeblah, A. (2010). Efficient combined immune-decomposition algorithm for optimal buffer allocation in production lines for throughput and profit maximization. *Computers & Operations Research*, **37**, 611–620. 86

Mezura-Montes, E., Reyes-Sierra, M. and Coello Coello, C.A. (2008). Multi-objective optimization using differential evolution: A survey of the state-of-the-art. In U. Chakraborty, ed., *Advances in Differential Evolution*, vol. 143 of *Studies in Computational Intelligence*, 173–196, Springer, Berlin/Heidelberg, 10.1007/978-3-540-68830-3-7. 27

Miranda, G., de Armas, J., Segura, C. and Leon, C. (2010). Hyperheuristic codification for the multi-objective 2D Guillotine Strip Packing Problem. In *2010 IEEE Congress on Evolutionary Computation*, IEEE, Piscataway, NJ, USA, 2010 IEEE Congress on Evolutionary Computation, 18–23 July 2010, Barcelona, Spain. 32

Mishra, K. and Harit, S. (2010). A fast algorithm for finding the non dominated set in multi-objective optimization. *International Journal of Computer Applications*, **1**, 35–39. 16

Mishra, S., Panda, G., Meher, S. and Majhi, R. (2011). A study on multi-objective evolutionary algorithms and its applications to economics and finance. In *International Conference on Electronics Systems (ICES-2011)*, National Institute of Technology, Rourkela, India. 35

Moncayo-Martinez, L.A. and Zhang, D.Z. (2011). Multi-objective ant colony optimisation: A meta-heuristic approach to supply chain design. *International Journal of Production Economics*, **131**, 407–420. 25

Montazer-Rahmati, M.M. and Binaee, R. (2010). Multi-objective optimization of an industrial hydrogen plant consisting of a CO_2 absorber using DGA and a methanator. *Computers and Chemical Engineering*, **34**, 1813–1821. 37

Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Tech. Rep., Caltech, Caltech Concurrent Computation Program, report 826. 30

Muller, A. (2008). Developing the idea of the thesis and the protocol. In L.O. Lategan, ed., *An Introduction to postgraduate supervision*, 41–66, AFRICAN SUN MeDIA. 7

Neri, F. and Tirronen, V. (2010). Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review*, **33**, 61–106, 10.1007/s10462-009-9137-2. 27

Olsson, A.E., ed. (2011). *Particle Swarm Optimization: Theory, Techniques and Applications (Engineering Tools, Techniques and Tables)*. Nova Science Pub Inc. 26

Ombuki, B., Ross, B. and Hanshar, F. (2006). Multi-objective genetic algorithms for vehicle routing problem with time windows. *Applied Intelligence*, **24**, 17–30. 69

Paly, M.D., Schültze, N. and Zell, A. (2010). Determining crop-production functions using multi-objective evolutionary algorithms. In *2010 IEEE Congress on Evolutionary Computation*, 8 pp., IEEE, Piscataway, NJ, USA, 2010 IEEE Congress on Evolutionary Computation, 18–23 July 2010, Barcelona, Spain. 29

Panigrahi, B.K., Shi, Y. and Lim, M.H. (2011). *Handbook of Swarm Intelligence: Concepts, Principles and Applications*. Springer-Verlag, Berlin, 10.1007/978-3-642-17390-5. 26

Papadopoulos, H. and Vidalis, M. (2001). A heuristic algorithm for the buffer allocation in unreliable unbalanced production lines. *Computers & Industrial Engineering*, **41**, 261–277. 86

Papadopoulos, H.T. and Heavey, C. (1996). Queuing theory in manufacturing systems analysis and design: A classification of models for production and transfer lines. *European Journal of Operational Research*, **92**, 1–27. 84

Pedersen, M.E.H. (2010). *Tuning & Simplifying Heuristical Optimization*. Ph.D. thesis, School of Engineering Sciences, Computational Engineering and Design Group, University of Southampton. 27

Pierro, F.D., Khu, S. and Savi, D. (2007). An investigation on preference order ranking scheme for multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, **11**, 17–45. 17

Poles, S. and Lovison, A. (2009). A polynomial chaos approach to robust multiobjective optimization. In K. Deb, S. Greco, K. Miettinen and E. Zitzler, eds., *Hybrid and Robust Approaches to Multiobjective Optimization*, No. 09041 in Dagstuhl Seminar Proceedings, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Germany, Dagstuhl, Germany. 38

Purshouse, R. and Fleming, P. (2007). On the evolutionary optimization of many conflicting objectives. *IEEE Transactions on Evolutionary Computation*, **11**, 770–784. 16, 17, 20

Qin, H., Zhou, J., Lu, Y., Li, Y. and Zhang, Y. (2010). Multi-objective cultured differential evolution for generating optimal trade-offs in reservoir flood control operation. *Water Resources Management*, **24**, 2611–2632, 10.1007/s11269-009-9570-7. 27

Qu, B.Y. and Suganthan, P.N. (2009). Multi-objective evolutionary programming without non-domination sorting is up to twenty times faster. In *CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation*, 2934–2939, IEEE Press, Piscataway, NJ, USA. 16, 67

Raad, D., Sinske, A. and van Vuuren, J. (2011). Water distribution systems design optimisation using metaheuristics and hyperheuristics. *ORiON*, **27**, 17–44. 32

Rabbani, M., Bajestani, M.A. and Khoshkhou, G.B. (2010). A multi-objective particle swarm optimization for project selection problem. *Expert Systems with Applications*, **37**, 315–321. 25

Reddy, M.J. and Kumar, D.N. (2007). Multiobjective differential evolution with application to reservoir system optimization. *Journal of Computing in Civil Engineering*, **21**, 136–146. 27

Robič, T. and Filipič, B. (2005). Demo: Differential evolution for multiobjective optimization. In C.A. Coello Coello, A.H. Aguirre and E. Zitzler, eds., *Evolutionary Multi-Criterion Optimization*, vol. 3410 of *Lecture Notes in Computer Science*, 520–533, Springer, Berlin/Heidelberg, 10.1007/978-3-540-31880-4-36. 27

Rosen, S.L., Harmonosky, C.M. and Traband, M.T. (2007). A simulation optimization method that considers uncertainty and multiple performance measures. *European Journal of Operational Research*, **181**, 315–330. 3, 12

Rosen, S.L., Harmonosky, C.M. and Traband, M.T. (2008). Optimization of systems with multiple performance measures via simulation: Survey and recommendations. *Computers & Industrial Engineering*, **54**, 327–339. 3

Roy, T. and Maiti, M. (1998). Multi-objective inventory models of deteriorating items with some constraints in a fuzzy environment. *Computers & Operations Research*, **25**, 1085–1095. 35

Rubinstein, R.Y. and Kroese, D.P. (2004). *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Springer. 6, 40, 41, 42, 43, 47, 51, 67, 85, 89, 91, 149

Ruiz-Torres, A.J., Ho, J.C. and Ablanedo-Rosas, J.H. (2011). Makespan and workstation utilization minimization in a flowshop with operations flexibility. *Omega*, **39**, 273–282. 23

Sankararao, B. and Kyoo Yoo, C. (2011). Development of a robust multiobjective simulated annealing algorithm for solving multiobjective optimization problems. *Industrial & Engineering Chemistry Research*, **50**, 6728–6742. 23

Sasaki, D. and Obayashi, S. (2005). Efficient search for trade-offs by adaptive range multi-objective genetic algorithms. *Journal of Aerospace Computing, Information and Communication*, **2**, 44–64. 15

Shi, C. and Gershwin, S.B. (2009). An efficient buffer design algorithm for production line profit maximization. *International Journal of Production Economics*, **122**, 725–740. 86

Shixin Liu (2010). Model and algorithm for hot rolling batch planning in steel plants. *International Journal on Information and Management Sciences*, **21**, 247–63. 25

Shukla, P.K. and Deb, K. (2007). On finding multiple Pareto-optimal solutions using classical and evolutionary generating methods. *European Journal of Operational Research*, **181**, 1630–1652. 61

Singh, H.K., Ray, T. and Smith, W. (2010). C-PSA: Constrained Pareto simulated annealing for constrained multi-objective optimization. *Information Sciences*, **180**, 2499–2513. 23

Sivakumar, K., Balamurugan, C. and Ramabalan, S. (2011). Simultaneous optimal selection of design and manufacturing tolerances with alternative manufacturing process selection. *Computer-aided Design*, **43**, 207–218. 34

Smith, J.M. and Cruz, F.R.B. (2005). The buffer allocation problem for general finite buffer queuing networks. *IIE Transactions*, **37**, 343–365. 91, 92

Solomon, M.M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, **35**, 254–265. 69

Srinivas, N. and Deb, K. (1995). Comparative study of vector evaluated GA and NSGA applied to multiobjective optimization. In P.K. Roy and S.D. Mehta, eds., *Proceedings of the Symposium on Genetic Algorithms*, 83–90. 35

Stadler, J. (2012). *Multi-objective optimisation using the cross-entropy method in CO gas management at a South African ilmenite smelter*. Master's thesis, Stellenbosch University. 124, 125, 126

Storn, R. and Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, **11**, 341–359, 10.1023/A:1008202821328. 27

Suman, B. and Kumar, P. (2006). A survey of simulated annealing as a tool for single and multiobjective optimization. *The Journal of the Operational Research Society*, **57**, 1143–1160. 23

Summanwar, V.S., Jayaraman, V.K., Kulkarni, B.D., Kusumakar, H.S., Gupta, K. and Rajesh, J. (2002). Solution of constrained optimization problems by multi-objective genetic algorithm. *Computers & Chemical Engineering*, **26**, 1481–1492. 15

Surekha, B., Kaushik, L., Panduy, A., Vundavilli, P. and Parappagoudar, M. (2011). Multi-objective optimization of green sand mould system using evolutionary algorithms. *The International Journal of Advanced Manufacturing Technology*, 1–9, 10.1007/s00170-011-3365-8. 26

Szőllős, A., Šmíd, M. and Hájek, J. (2009). Aerodynamic optimization via multi-objective micro-genetic algorithm with range adaptation, knowledge-based reinitialization, crowding and [epsilon]-dominance. *Advances in Engineering Software*, **40**, 419–430. 18, 32

Tan, K., Chew, Y. and Lee, L. (2006). A hybrid multiobjective evolutionary algorithm for solving the vehicle routing problem with time windows. *Computational Optimization and Applications*, **34**, 115–151. 69

Tarafder, A., Rangaiah, G. and Ray, A.K. (2005). Multiobjective optimization of an industrial styrene monomer manufacturing process. *Chemical Engineering Science*, **60**, 347–363. 36

Tarafder, A., Rangaiah, G. and Ray, A.K. (2007). A study of finding many desirable solutions in multiobjective optimization of chemical processes. *Computers and Chemical Engineering*, **31**, 1257–1271. 37

Tavakkoli-Moghaddam, R., Rahimi-Vahed, A. and Mirzaei, A.H. (2007). A hybrid multi-objective immune algorithm for a flow shop scheduling problem with bi-objectives: Weighted mean completion time and weighted mean tardiness. *Information Sciences*, **177**, 5072–5090. 28

Toth, P. and Vigo, D. (2002). *The vehicle routing problem*, vol. 9. Society for Industrial Mathematics. 68

Tsou, C.S. (2008). Multi-objective inventory planning using MOPSO and TOPSIS. *Expert Systems with Applications*, **35**, 136–142. 4, 13, 35

Tsou, C.S. (2009). Evolutionary Pareto optimizers for continuous review stochastic inventory systems. *European Journal of Operational Research*, **195**, 364–371. 35

Van Veldhuizen, D. and Lamont, G. (2000). Multiobjective Optimization with Messy Genetic Algorithms. In *Proceedings of the 2000 ACM Symposium on Applied Computing*, 470–476, ACM, Villa Olmo, Como, Italy. 14

Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, **45**, 41–51, 10.1007/BF00940812. 22

Veldhuizen, A.V. (1999). *Multiobjective evolutionary algorithms: Classifications, Analyzes, and New Innovations*. Ph.D. thesis, Air Force Institute of Technology, Wright-Patterson AFB. 18

Voss, T., Trautmann, H. and Igel, C. (2011). New uncertainty handling strategies in multi-objective evolutionary optimization. In R. Schaefer, C. Cotta, J. Kolodziej and G. Rudolph, eds., *Parallel Problem Solving from Nature – PPSN XI*, vol. 6239 of *Lecture Notes in Computer Science*, 260–269, Springer, Berlin/Heidelberg, 10.1007/978-3-642-15871-1_27. 38

Vouros, G.A. and Papadopoulos, H.T. (1998). Buffer allocation in unreliable production lines using a knowledge based system. *Computers & Operations Research*, **25**, 1055–1067. 85, 86, 89, 90

Vrugt, J. and Robinson, B. (2007). Improved evolutionary optimisation from genetically adaptive multimethod search. In *Proceedings of the National Academy of Sciences*, vol. 104, 708–711. 32

Wang, Y. and Yang, Y. (2009). Particle swarm optimization with preference order ranking for multi-objective optimization. *Information Sciences*, **179**, 1944–1959. 16

Wang, Y.N., Wu, L.H. and Yuan, X.F. (2010). Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure. *Soft Computing – A Fusion of Foundations, Methodologies and Applications*, **14**, 193–209, 10.1007/s00500-008-0394-9. 17

Watkins, C. and Dayan, P. (1992). Q-learning. *Machine Learning*, **8**, 279–292. 26

Weise, T. (2009). *Global Optimization Algorithms – Theory and Application*. Self-published, 2nd edn., online available at http://www.it-weise.de/, accessed on 20 November 2011. 23, 24, 25, 26

Wolpert, D.H. and Macready, W.G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, **1**, 67–82. 59

Xiang, Y., Arora, J., Rahmatalla, S., Marler, T., Bhatt, R. and Abdel-Malek, K. (2010). Human lifting simulation using a multi-objective optimization approach. *Multibody System Dynamics*, **23**, 431–451, 10.1007/s11044-009-9186-y. 34

Yagmahan, B. and Yenisey, M.M. (2010). A multi-objective ant colony system algorithm for flow shop scheduling problem. *Expert Systems with Applications*, **37**, 1361–1368. 25

Yang, S., Wu, C. and Hu, S.J. (2000). Modeling and analysis of multi-stage transfer lines with unreliable machines and finite buffers. *Annals of Operations Research*, **93**, 405–421. 85

Yang, X.S. (2010). Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation*, **2**, 78–84. 31

Yu, V.F., Lin, S.W., Lee, W. and Ting, C.J. (2010). A simulated annealing heuristic for the capacitated location routing problem. *Computers & Industrial Engineering*, **58**, 288–299, scheduling in Healthcare and Industrial Systems. 23

Zadeh, L. (1963). Optimality and nonscalar-valued performance criteria. *IEEE Transactions on Automatic Control*, **AC-8**, 59–60. 11

Zhihuan, L., Yinhong, L. and Xianzhong, D. (2010). Non-dominated sorting genetic algorithm-II for robust multi-objective optimal reactive power dispatch. *IET Generation Transmission & Distribution*, **4**, 1000–1008. 38

Zhou, A., Qu, B.Y., Li, H., Zhao, S.Z., Suganthan, P.N. and Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, **1**, 32–49. 14, 15, 33, 61

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, **3**, 257–271. 14, 15

Zitzler, E., Deb, K. and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, **8**, 173–195. 18, 61

Zitzler, E., Laumanns, M., Thiele, L., Fonseca, C.M. and Fonseca, V.G.D. (2002). Genetic Algorithms: Why quality assessment of multiobjective optimizers is difficult. Genetic and Evolutionary Computing Conference (GECCO 2002). 20

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. and da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, **7**, 117–132. 20

APPENDIX *A*

PLOTS FOR THE APPROXIMATE PARETO FRONTS OF THE BAP

The approximate Pareto fronts for the BAP instances discussed in Section 5.2.3.1 are shown graphically in the following figures. Also shown are the subsets of populations that were archived during the execution of the MOO CEM algorithm.

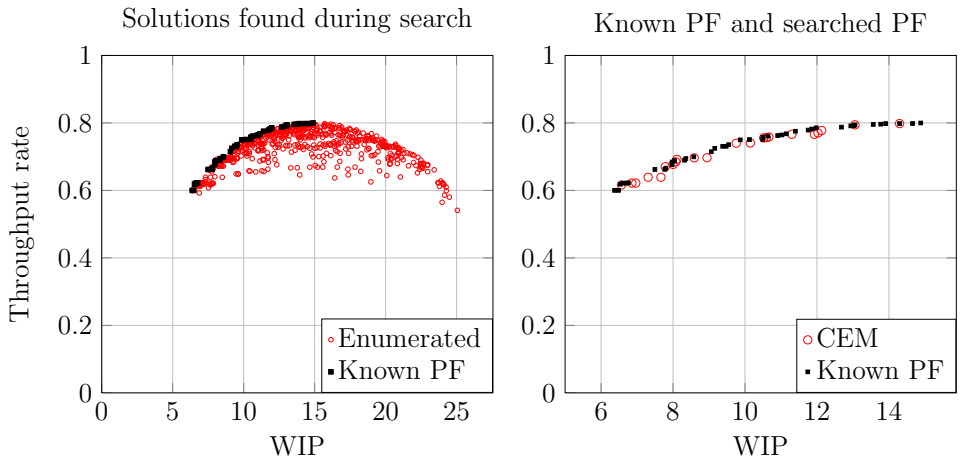

Figure A.1: Graphic results for $m = 5$ machines and $n = 20$ niches, exponential processing times.

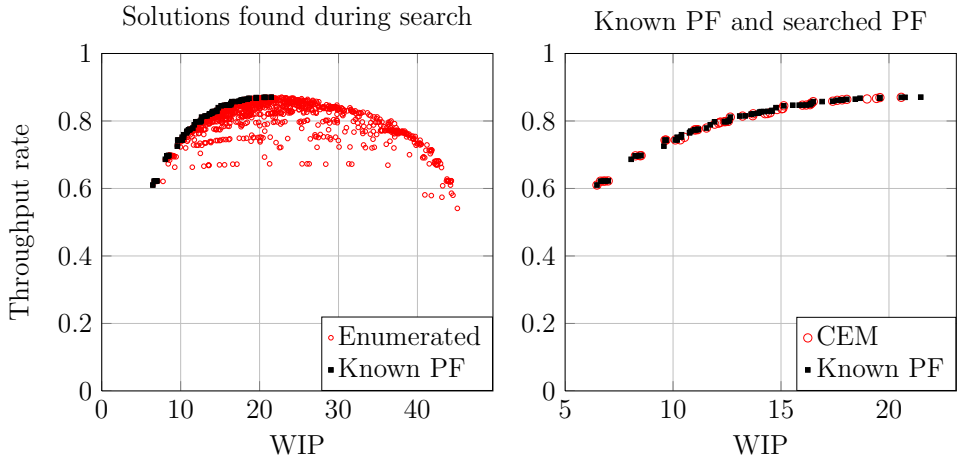

Figure A.2: Graphic results for $m = 5$ machines and $n = 40$ niches, exponential processing times.

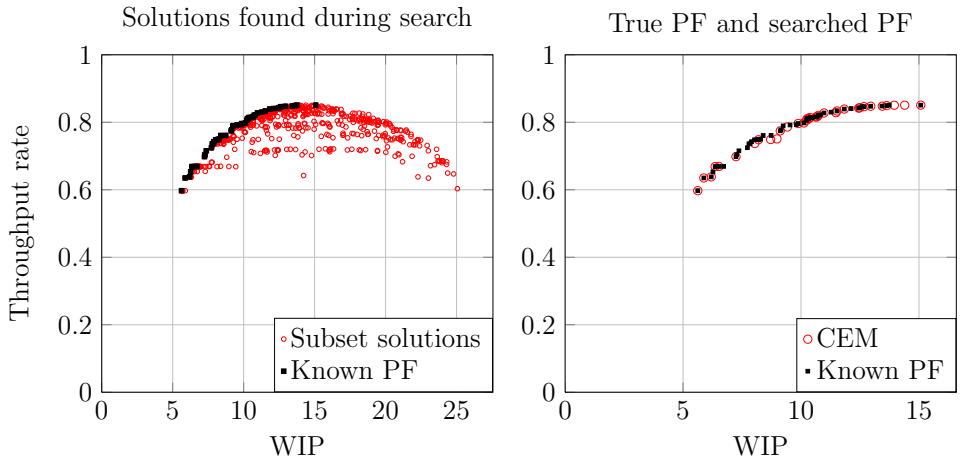

Figure A.4: Graphic results for $m = 5$ machines and $n = 20$ niches, Erlang_2 processing times.

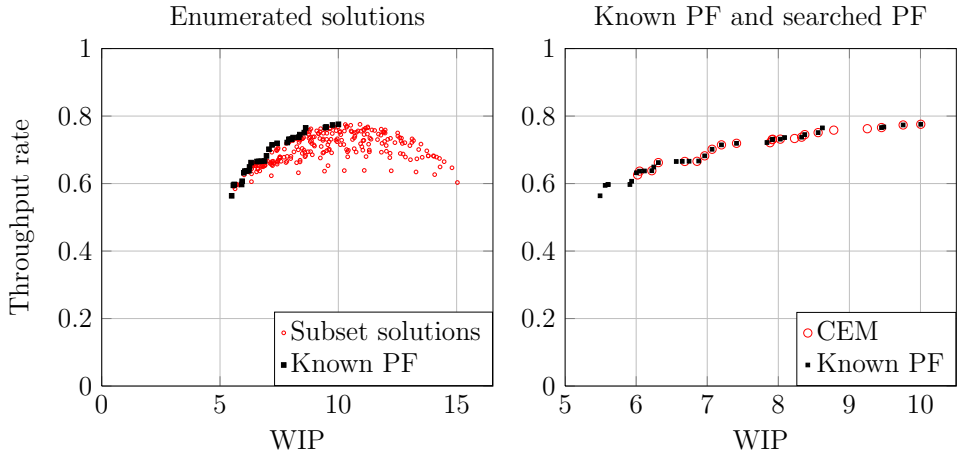

Figure A.3: Graphic results for $m = 5$ machines and $n = 10$ niches, Erlang_2 processing times.

The approximate Pareto sets for the non-serial problem ($m = 16$ machines, various n) are shown in the following figures. Again, the archived solutions are shown.

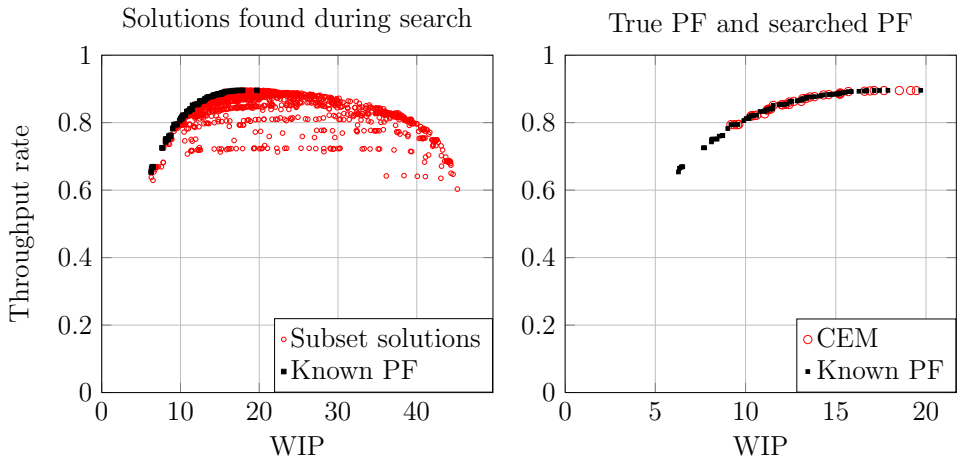

Figure A.5: Graphic results for $m = 5$ machines and $n = 40$ niches, Erlang_2 processing times.

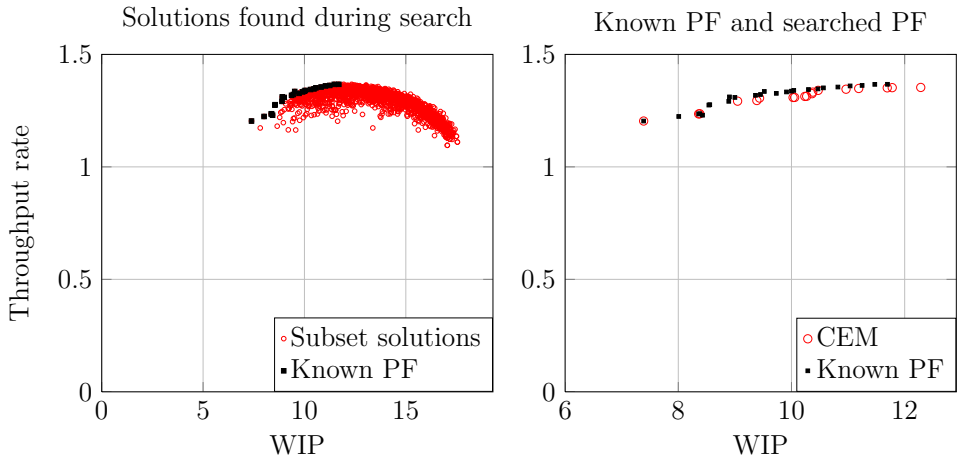

Figure A.6: Graphic results for $m = 10$ machines and $n = 10$ niches, exponential processing times.

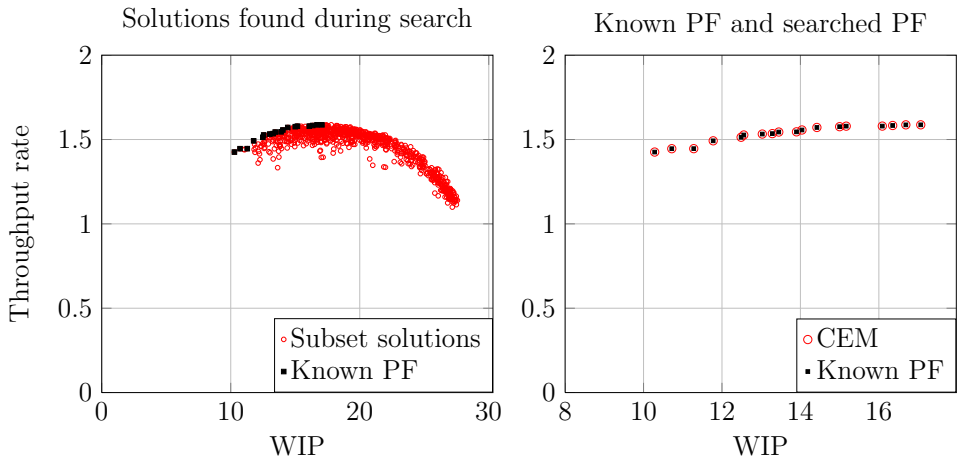

Figure A.7: Graphic results for $m = 10$ machines and $n = 20$ niches, exponential processing times.

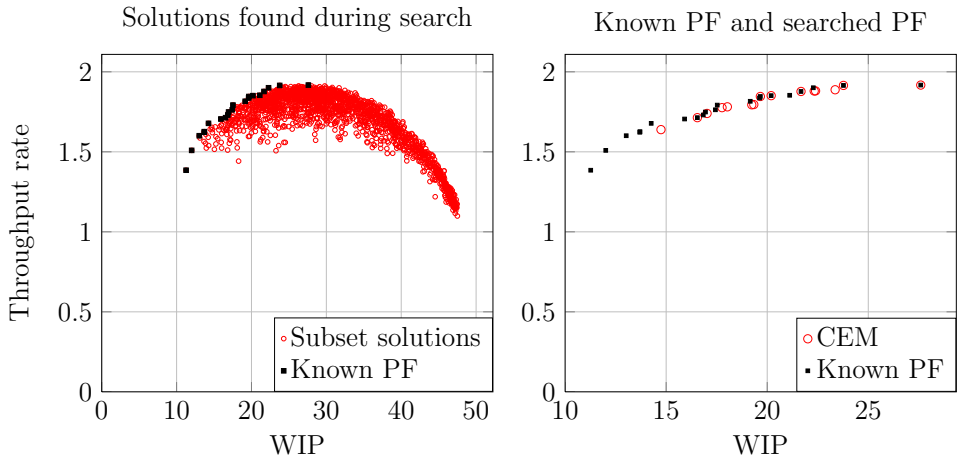

Figure A.8: Graphic results for $m = 10$ machines and $n = 40$ niches, exponential processing times.

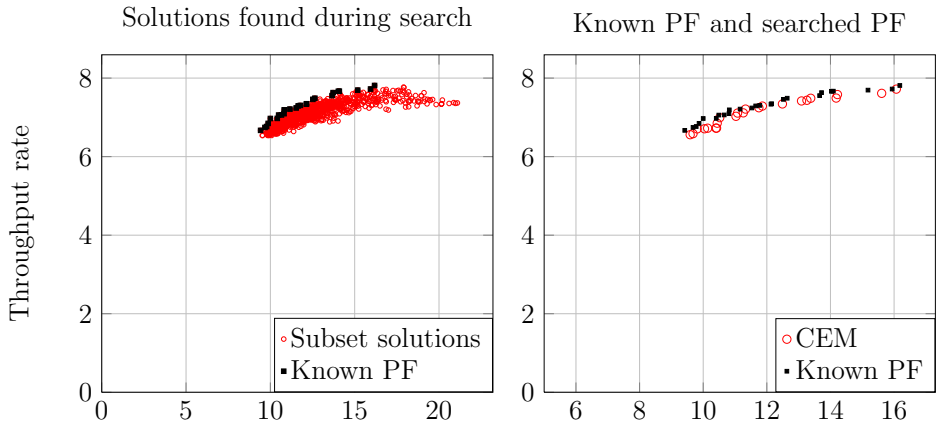

Figure A.9: Graphic results for $m = 16$ machines and $n = 10$ niches, exponential processing times.

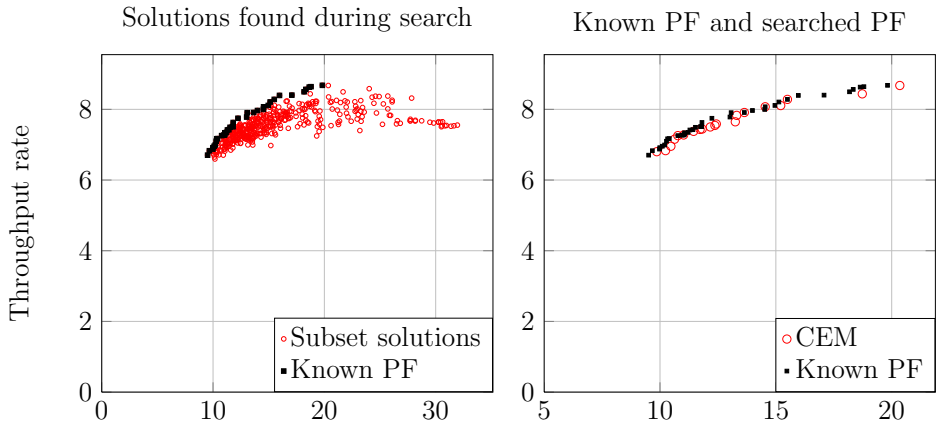

Figure A.10: Graphic results for $m = 16$ machines and $n = 20$ niches, exponential processing times.

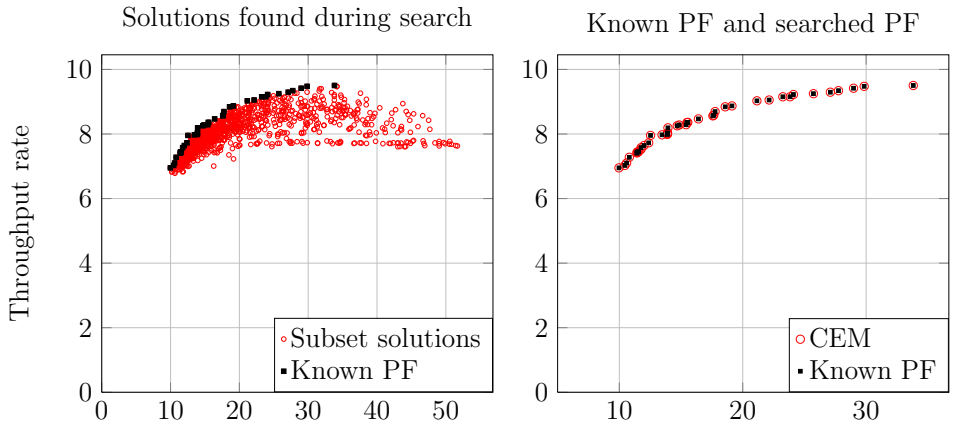

Figure A.11: Graphic results for $m = 16$ machines and $n = 40$ niches, exponential processing times.

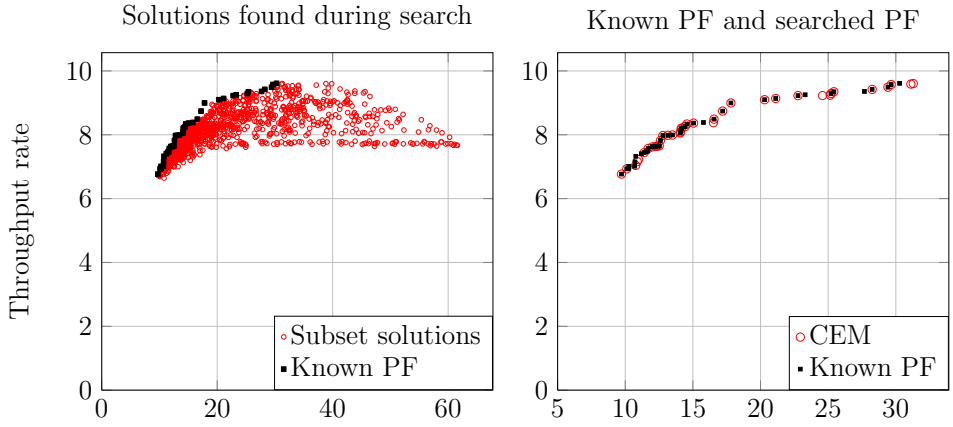

Figure A.12: Graphic results for $m = 16$ machines and $n = 50$ niches, exponential processing times.

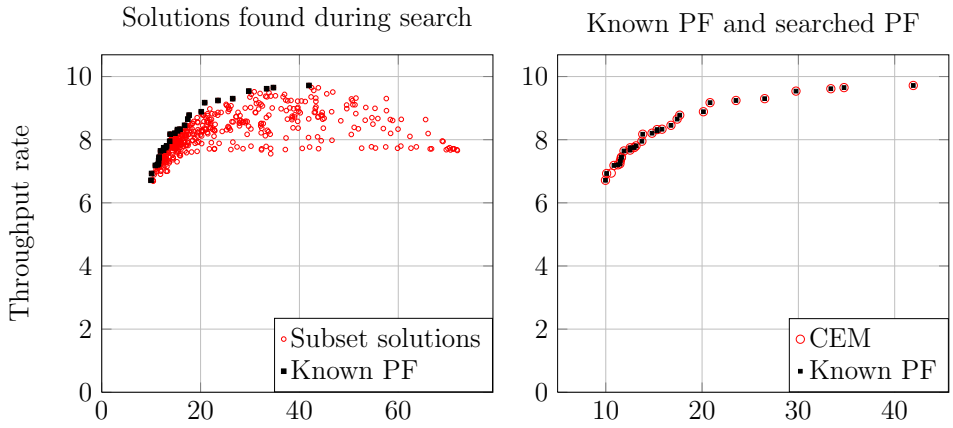

Figure A.13: Graphic results for $m = 16$ machines and $n = 60$ niches, exponential processing times.

APPENDIX *B*

SOLUTIONS FOR THE VEHICLE ROUTING PROBLEM

V1	V2	V3	V4	V5	V6
0	0	0	0	0	0
1 813	1 714	1 588	430 378	430 471	1 725
2 104	649	430 761	856	1 721	1 870
1 173	430 030	2 107	907	2 152	1 389
1 384	1 703	974	486	430 625	2 044
1 678	1 777	1 781	1 509	0	2 121
661	1 203	430 804	948		1 875
106	2 138	482	2 007		1 235
2 073	0	0	1 463		2 149
1 897			0		669
2 000					2 003
1 362					0
1 686					
430 148					
1 888					
0					

Table B.1: Routes of solution C in Figure B.2, 50_d1_tw4 ($Z2$ vs $Z3$).

V1	V2	V3	V4	V5	V6
0	0	0	0	0	0
1 725	1 870	1 714	2 104	856	1 588
2 107	2 044	430 378	1 813	1 173	1 463
430 761	2 121	106	1 384	649	2 073
1 781	1 389	661	1 897	430 030	430 625
669	1 235	486	1 888	1 703	2 000
482	1 875	907	430 148	1 777	0
0	974	1 509	1 686	1 203	
	2 149	1 678	2 152	2 138	
	430 804	2 007	1 721	0	
	2 003	948	1 362		
	0	430 471	0		
		0			

Table B.2: Routes of solution D in Figure B.4, 50_d1_tw4 ($Z2$ vs $Z5$).

A number of VRPSTW cases were discussed in Subsection 4.3.4, and further results for the MOO of these cases (50_d1_tw4 and 250_d1_tw4) are presented in this appendix. A complete results set is included in Hauman (2012).

B.1 Results for VRP 50_d1_tw4

The results for case 50_d1_tw4 are presented here, and the structure is similar to that of Subsection 4.3.5. The front progression is shown as a plot, with the final approximate front in an adjacent graph. The route mapping for an arbitrary selected solution from the final front is also shown.

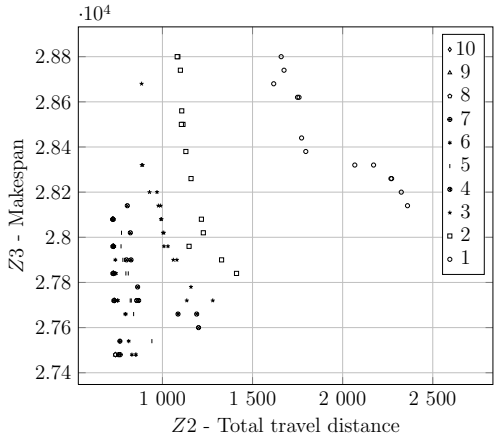

Figure B.1: Front progression of 50_d1_tw4 for $Z2$ vs $Z3$.

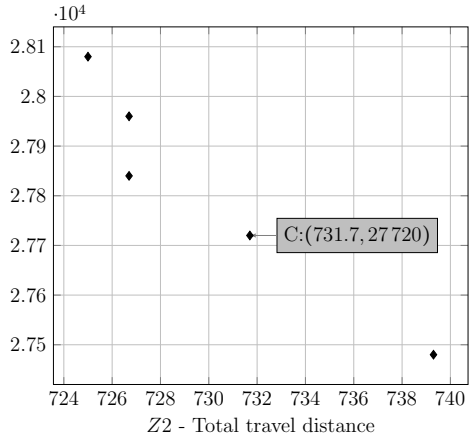

Figure B.2: Final approximate front of 50_d1_tw4 for $Z2$ vs $Z3$.

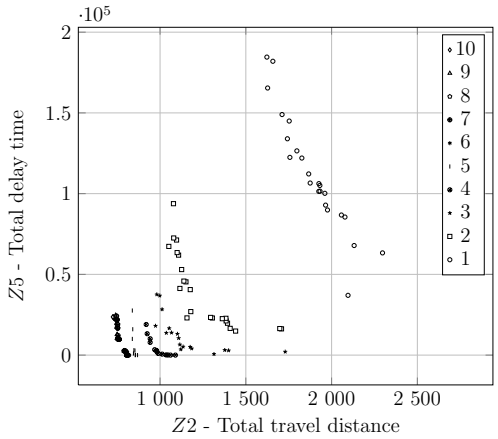

Figure B.3: Front progression of 50_d1_tw4 for $Z2$ vs $Z5$.

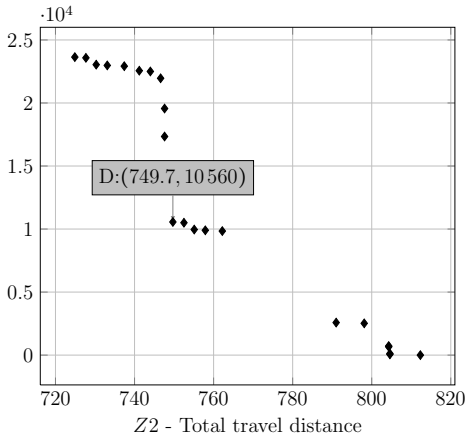

Figure B.4: Final approximate front of 50_d1_tw4 for $Z2$ vs $Z5$.

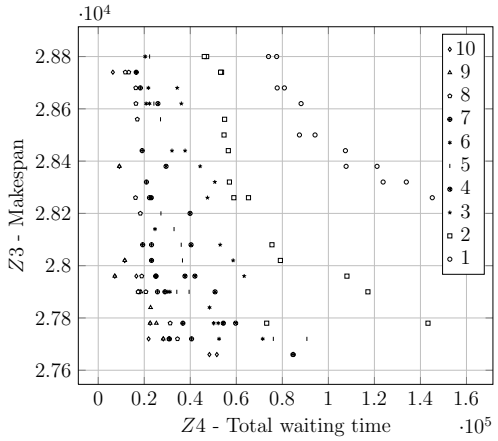

Figure B.5: Front progression of 50_d1_tw4 for $Z4$ vs $Z3$.

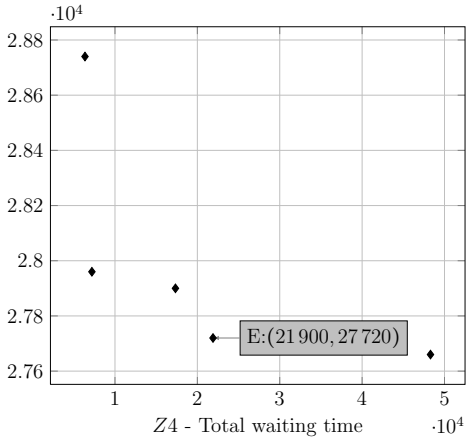

Figure B.6: Final approximate front of 50_d1_tw4 for $Z4$ vs $Z3$.

V1	V2	V3	V4	V5	V6
0	0	0	0	0	0
2 107	2 104	1 725	1 777	649	1 870
1 897	430 378	1 888	2 000	1 714	430 030
1 703	2 121	974	2 044	1 678	1 389
1 235	907	1 588	1 686	1 362	1 509
2 149	856	1 781	2 152	1 384	1 721
430 804	669	2 138	106	486	430 625
482	2 003	0	0	2 073	430 471
0	0			948	0
				661	
				2 007	
				1 463	
				0	

Table B.3: Routes of solution E in Figure B.6, 50_d1_tw4 ($Z4$ vs $Z3$).

V1	V2	V3	V4	V5	V6
0	0	0	0	0	0
649	2 107	1 725	1 870	1 897	1 173
430 378	1 777	1 384	2 000	1 813	856
1 203	106	2 104	1 875	1 714	1 888
661	1 463	1 703	2 044	430 030	430 761
2 149	1 509	1 588	2 121	974	1 235
430 471	907	669	1 686	1 389	2 073
430 625	486	2 003	2 152	1 721	430 148
0	1 781	0	1 362	2 007	2 138
	430 804		0	948	0
	482			1 678	
	0			0	

Table B.4: Routes of solution F in Figure B.8, 50_d1_tw4 ($Z4$ vs $Z5$).

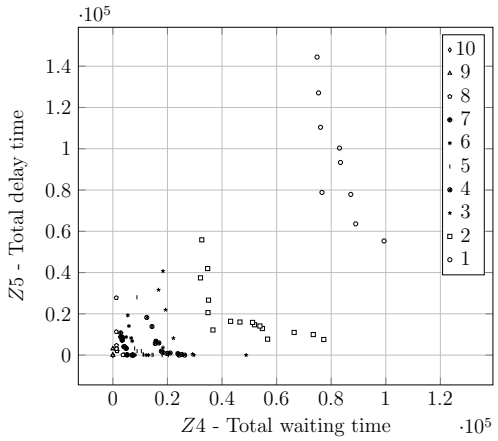

Figure B.7: Front progression of 50_d1_tw4 for $Z4$ vs $Z5$.

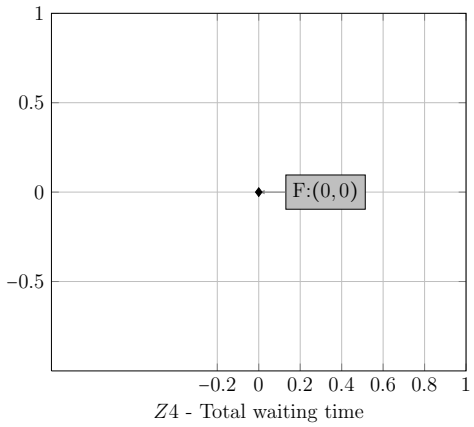

Figure B.8: Final approximate front of 50_d1_tw4 for $Z4$ vs $Z5$.

B.2 Results for VRP 250_d1_tw4

The results for case 250_d1_tw4 are presented next. The front progression and approximate front for $Z2$ vs $Z3$ are shown in Figure B.9 and Figure B.10, respectively. Note that this case has 250 visiting points.

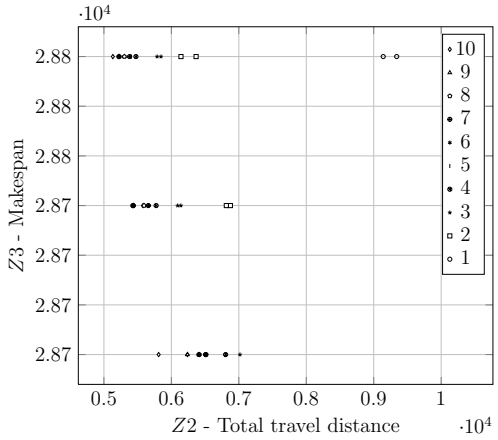

Figure B.9: Front progression of 250_d2_tw1 for $Z2$ vs $Z3$.

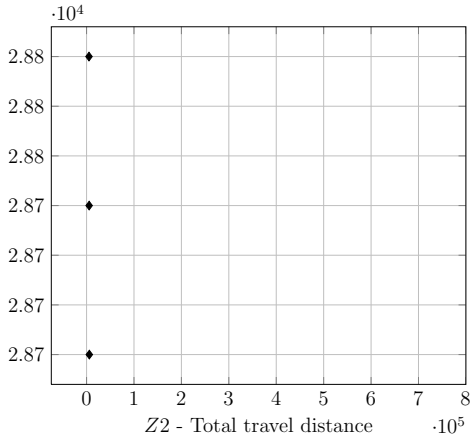

Figure B.10: Final approximate front of 250_d2_tw1 for $Z2$ vs $Z3$.

The front progression and approximate front for $Z4$ vs $Z5$ are shown in Figure B.11 and Figure B.12, respectively.

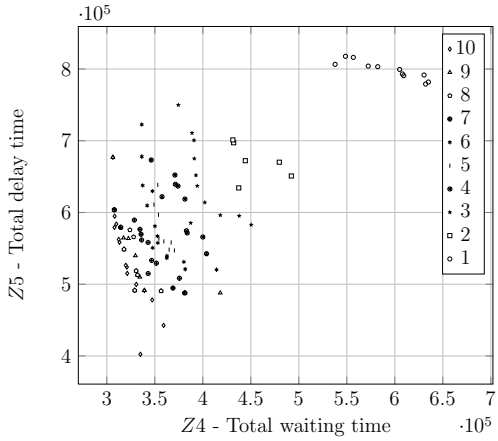

Figure B.11: Front progression of 250_d2_tw1 for $Z4$ vs $Z5$.

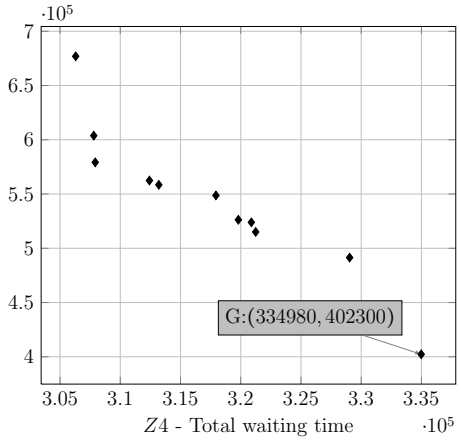

Figure B.12: Final approximate front of 250_d2_tw1 for $Z4$ vs $Z5$.

There are 39 routes for the solution with label "G" in Figure B.12 (case 250_d2_tw1 for $(Z4, Z5)$). The routes set was separated into four groups and these are presented in Figures B.13 – B.16.

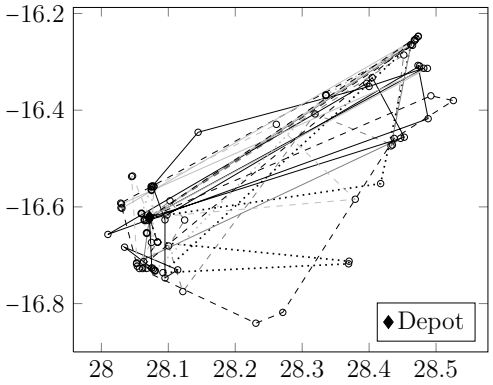

Figure B.13: Map of routes of solution G, 250_d2_tw1 for $Z4$ vs $Z5$, Part 1.

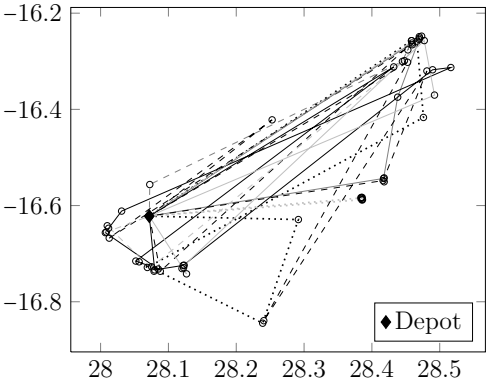

Figure B.14: Map of routes of solution G, 250_d2_tw1 for $Z4$ vs $Z5$, Part 2.

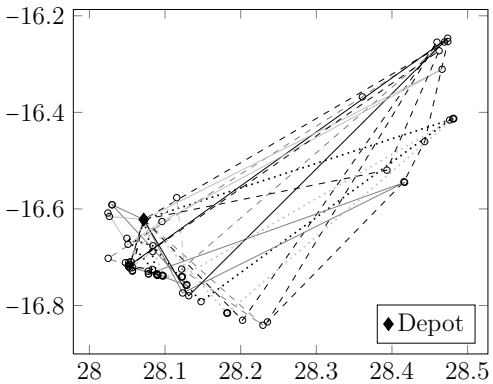

Figure B.15: Map of routes of solution G, 250_d2_tw1 for $Z4$ vs $Z5$, Part 3.

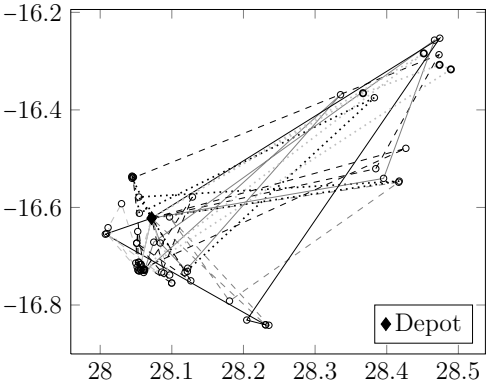

Figure B.16: Map of routes of solution G, 250_d2_tw1 for $Z4$ vs $Z5$, Part 4.

APPENDIX *C*

BOX-WHISKER PLOTS FOR HYPERAREA AND THE EPSILON QUALITY INDICATORS

The box-whisker plots for the benchmark problems discussed in Subsection 2.3.3 are shown in Figure C.1 to Figure C.8.

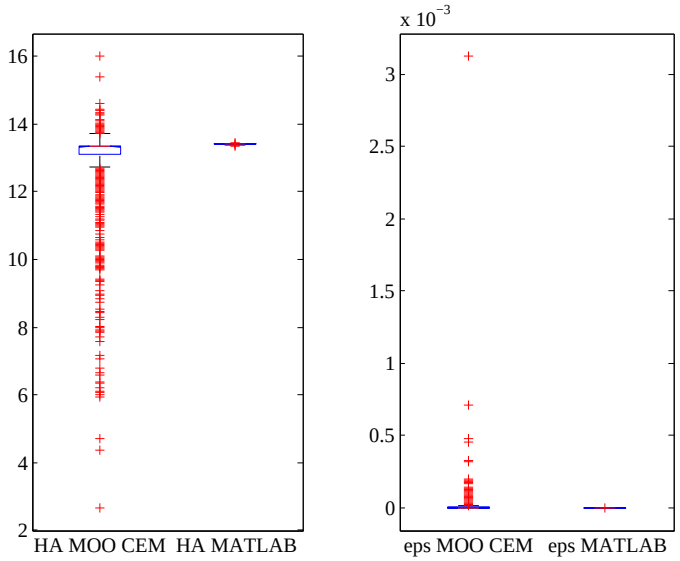

Figure C.1: Box-whisker plot for MOP1.

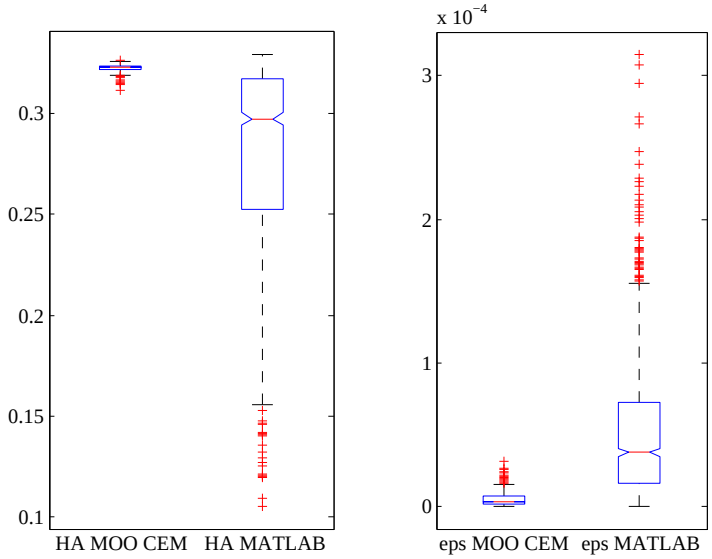

Figure C.2: Box-whisker plot for MOP2.

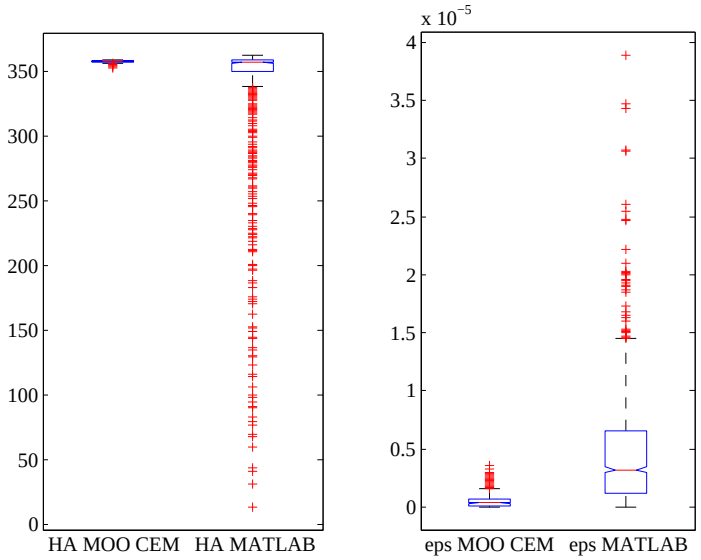

Figure C.3: Box-whisker plot for MOP3.

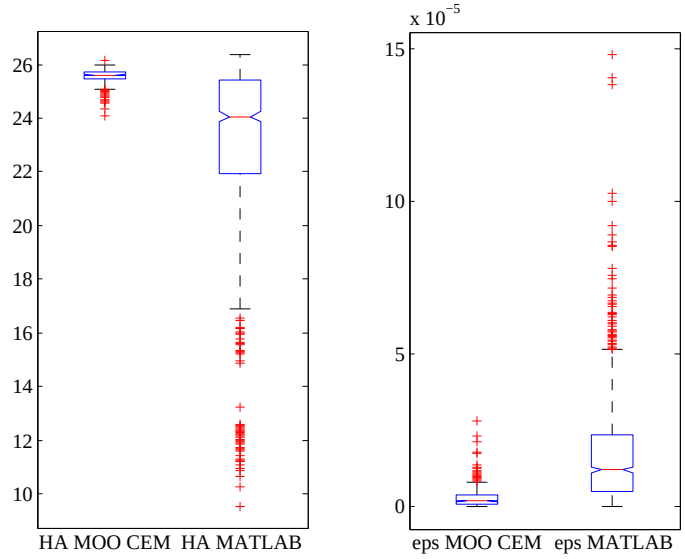

Figure C.4: Box-whisker plot for MOP4.

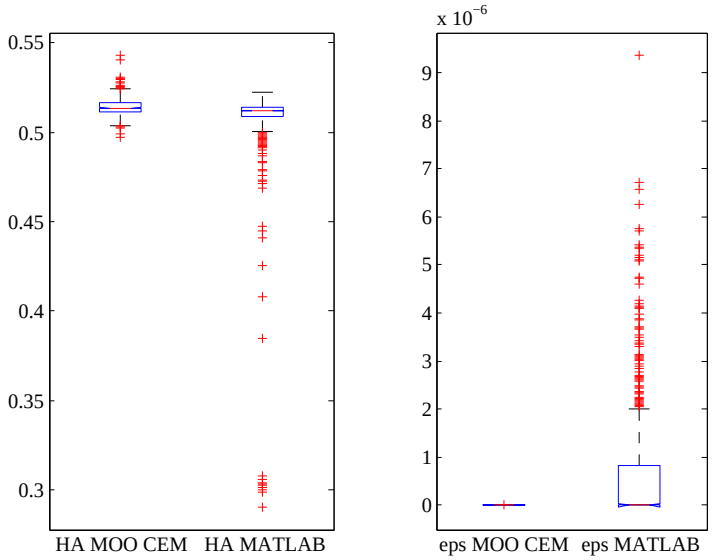

Figure C.5: Box-whisker plot for MOP6.

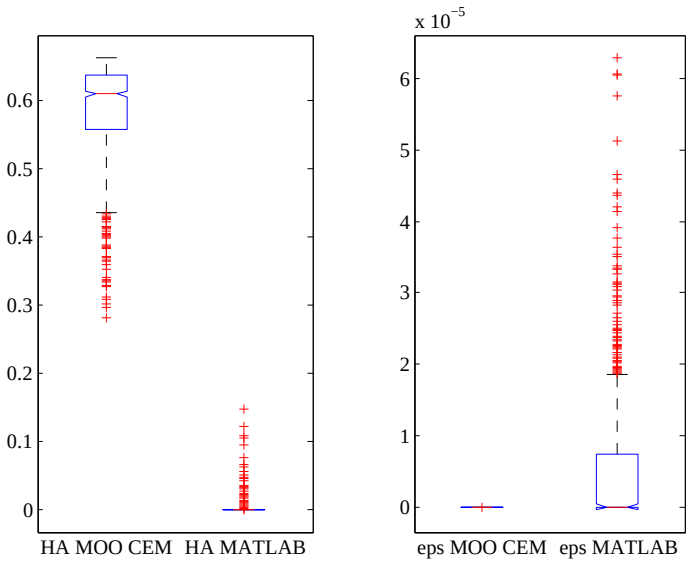

Figure C.6: Box-whisker plot for ZDT1.

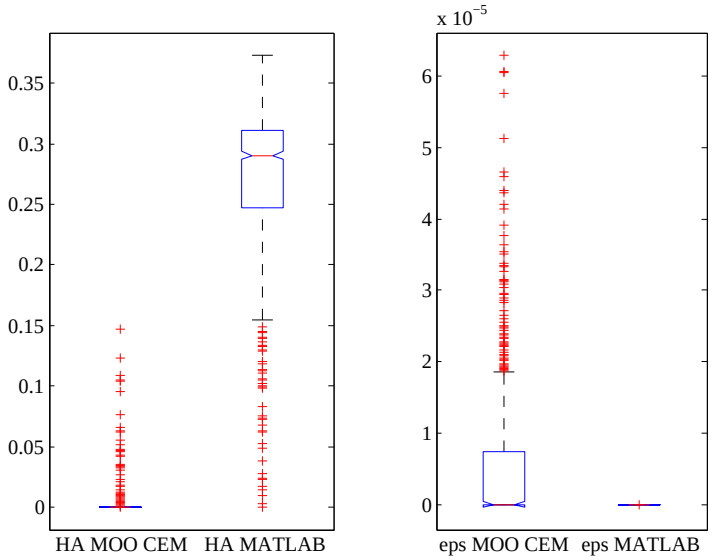

Figure C.7: Box-whisker plot for ZDT2.

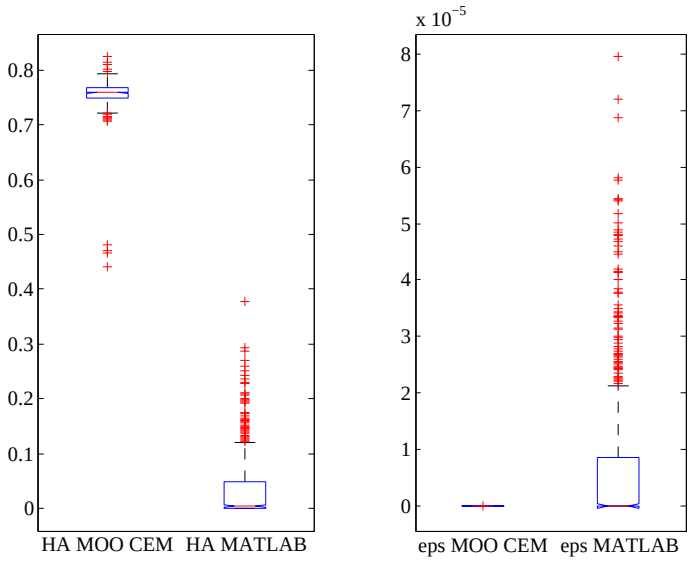

Figure C.8: Box-whisker plot for ZDT3.

The box-whisker plots for the buffer allocation problems discussed in Section 5.2 are shown in Figure C.9 to Figure C.24.

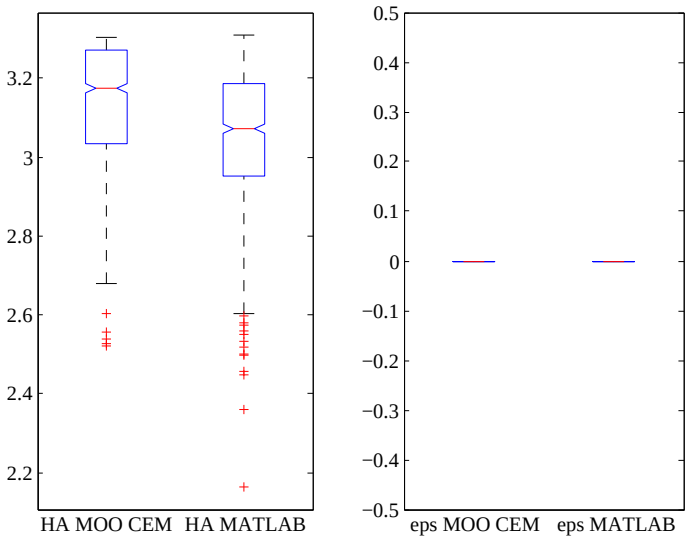

Figure C.9: Box-whisker plot for BAP1.

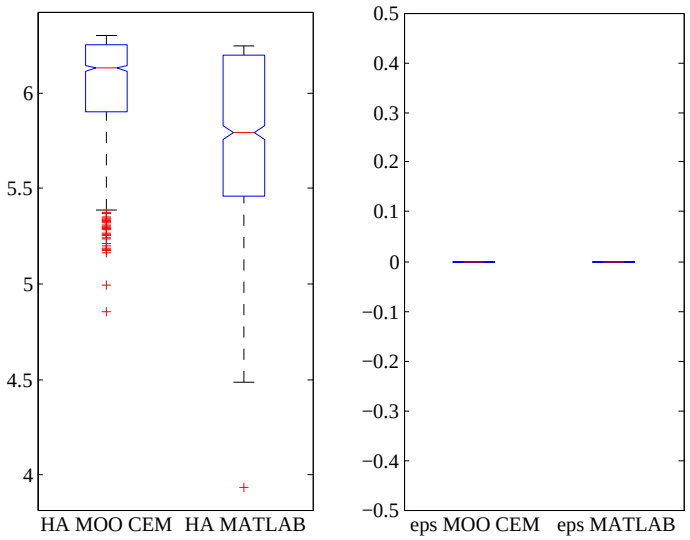

Figure C.10: Box-whisker plot for BAP2.

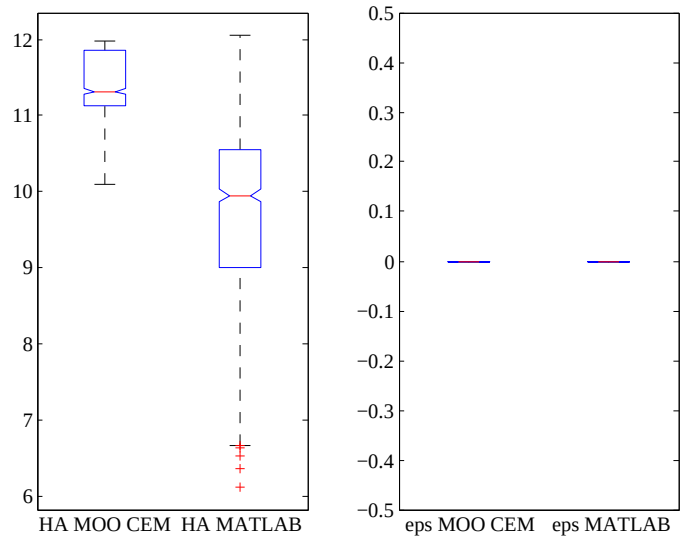

Figure C.11: Box-whisker plot for BAP3.

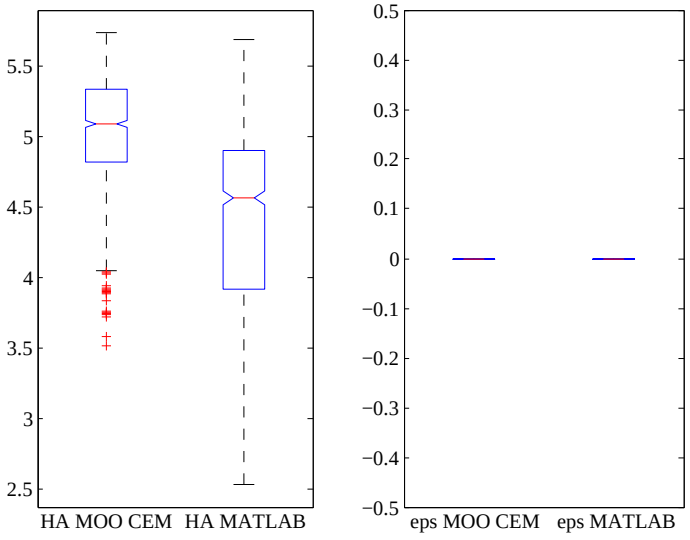

Figure C.12: Box-whisker plot for BAP4.

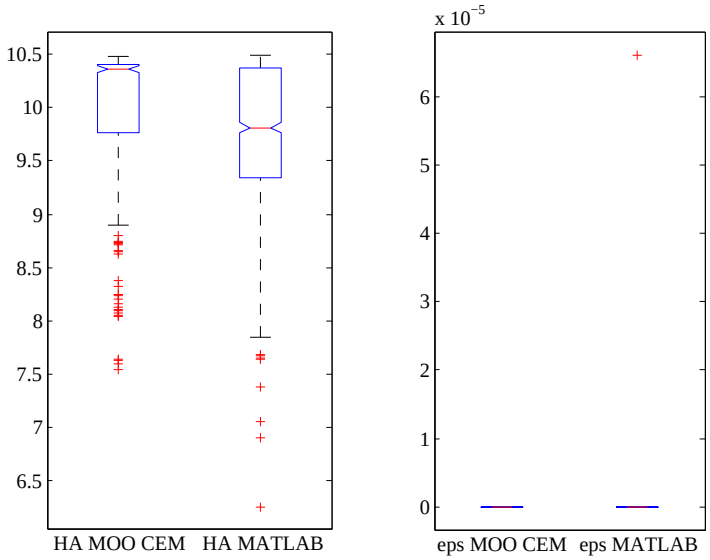

Figure C.13: Box-whisker plot for BAP5.

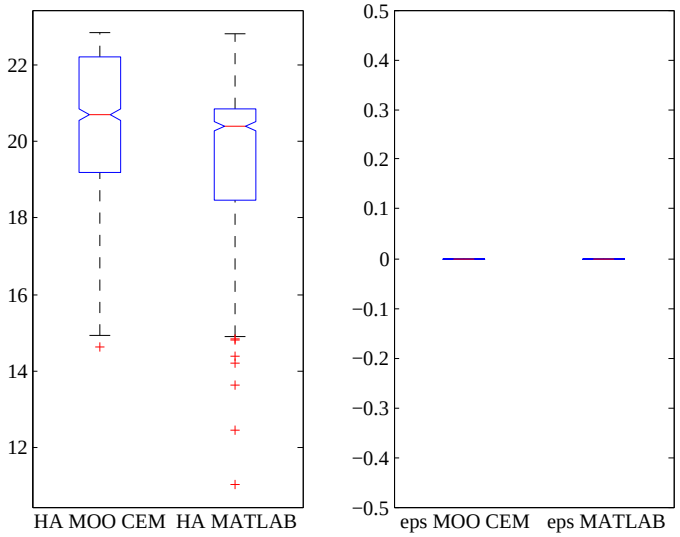

Figure C.14: Box-whisker plot for BAP6.

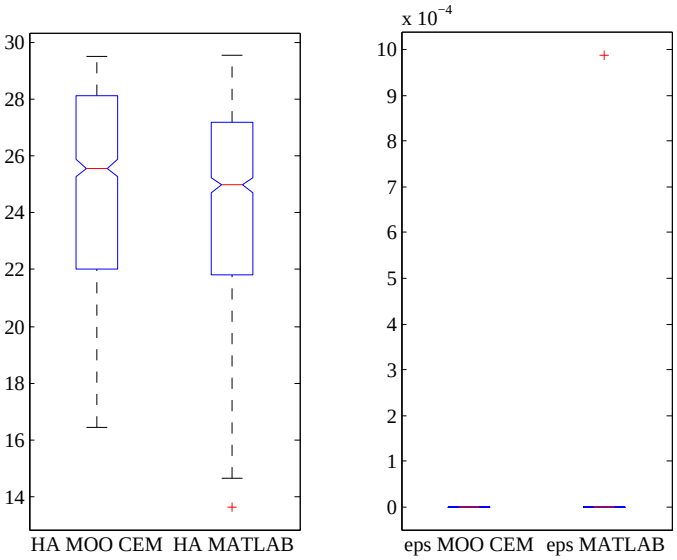

Figure C.15: Box-whisker plot for BAP7.

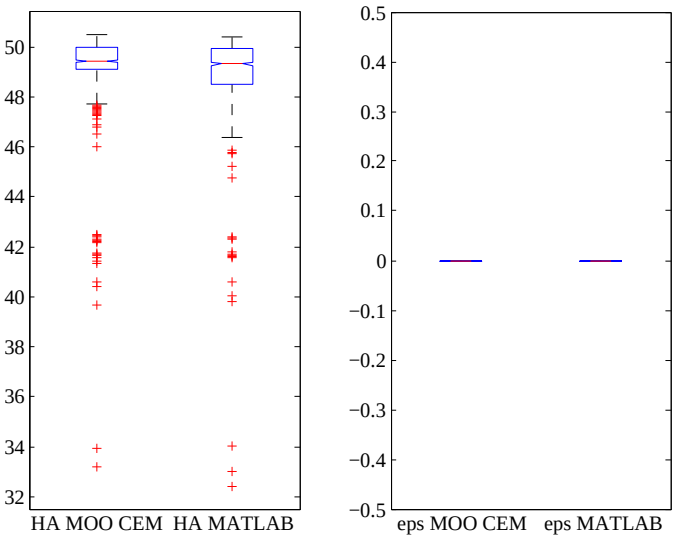

Figure C.16: Box-whisker plot for BAP8.

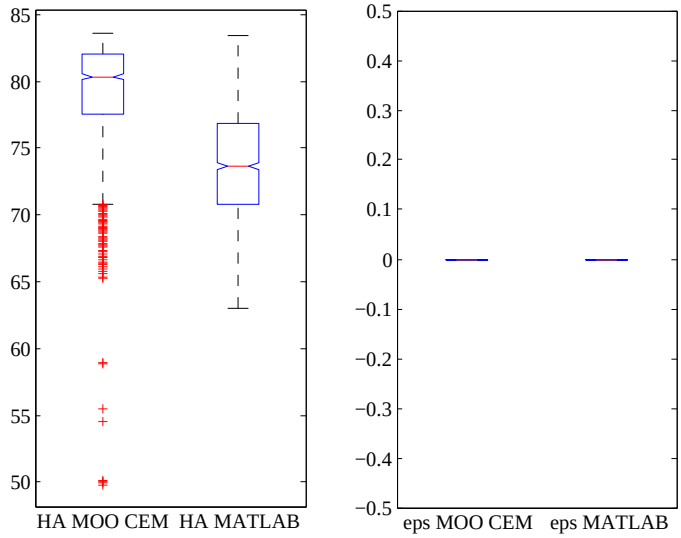

Figure C.17: Box-whisker plot for BAP9.

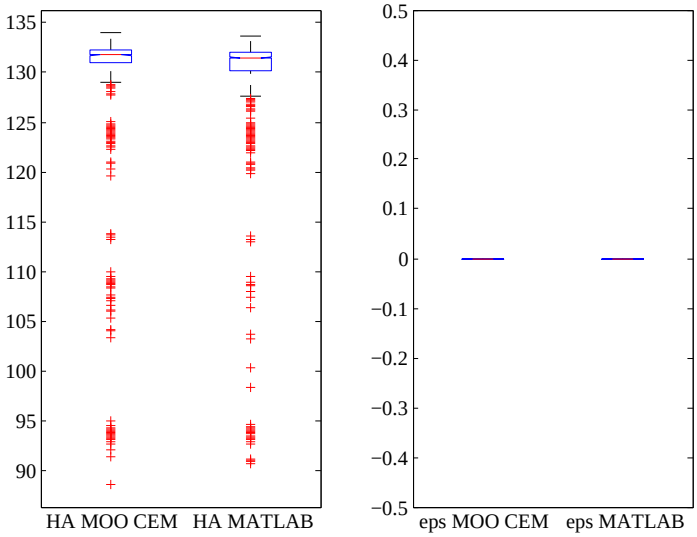

Figure C.18: Box-whisker plot for BAP10.

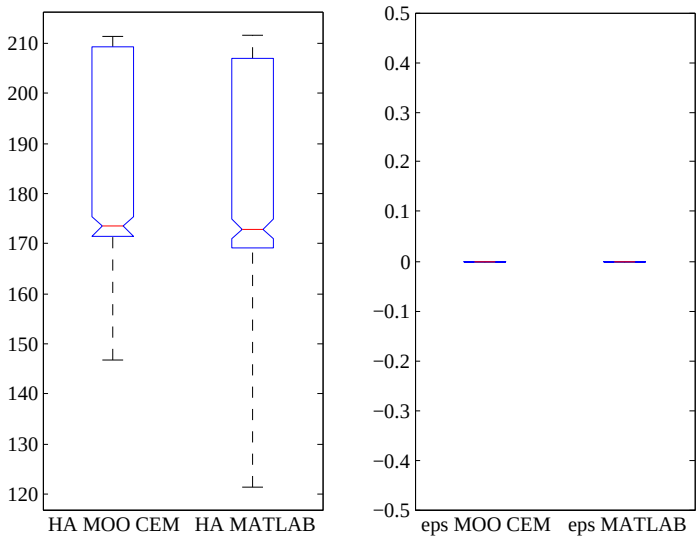

Figure C.19: Box-whisker plot for BAP11.

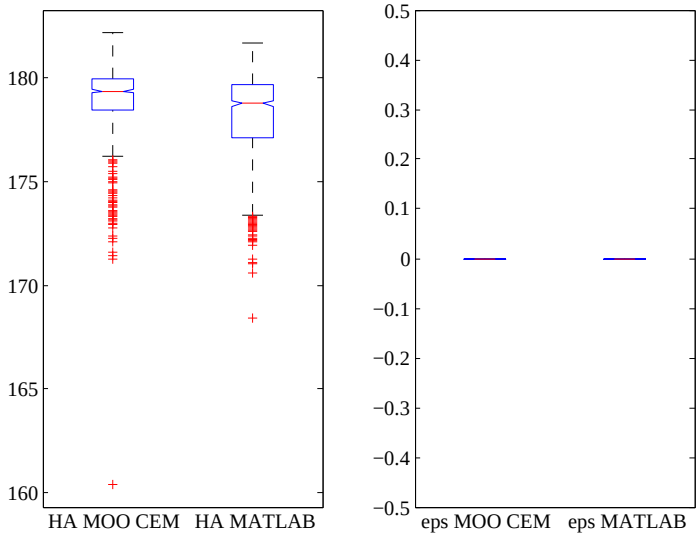

Figure C.20: Box-whisker plot for BAP12.

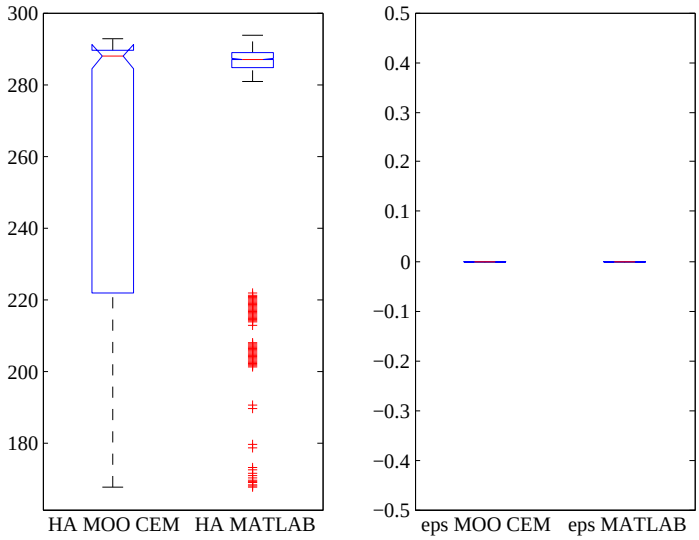

Figure C.21: Box-whisker plot for BAP13.

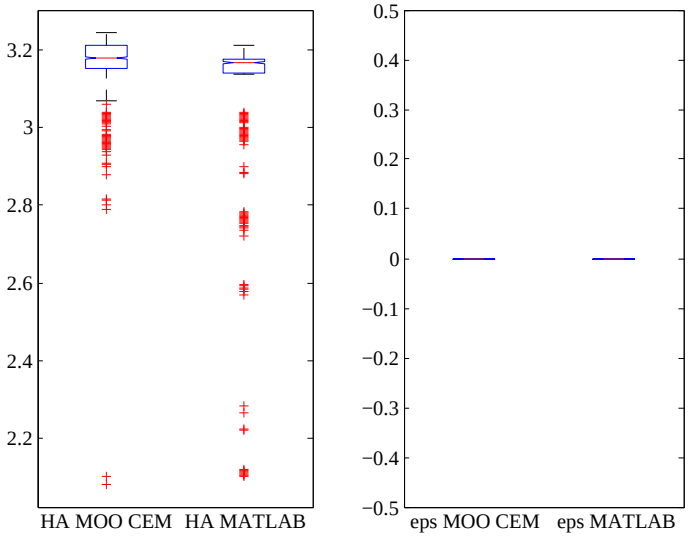

Figure C.22: Box-whisker plot for BAP14.

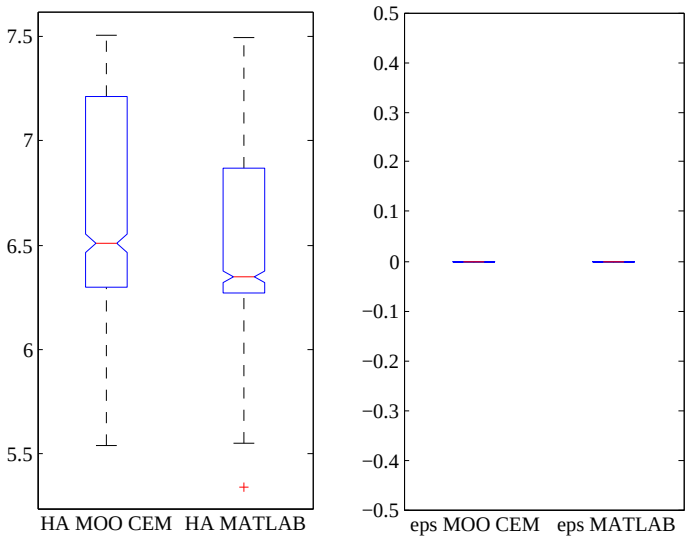

Figure C.23: Box-whisker plot for BAP15.

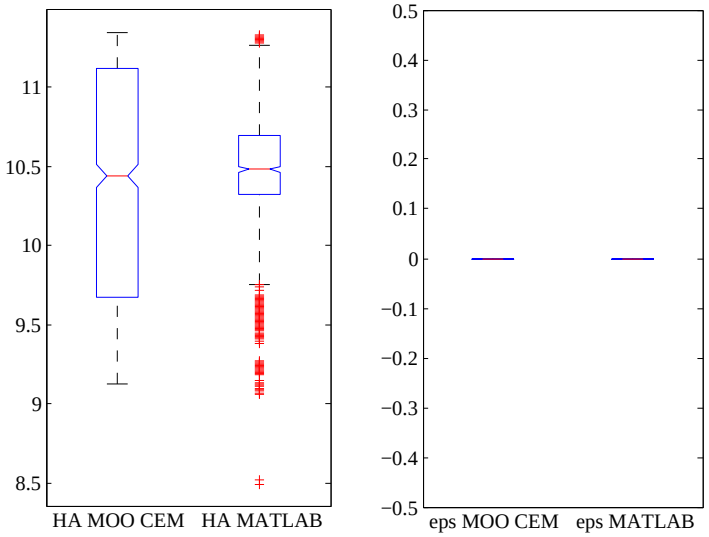

Figure C.24: Box-whisker plot for BAP16.

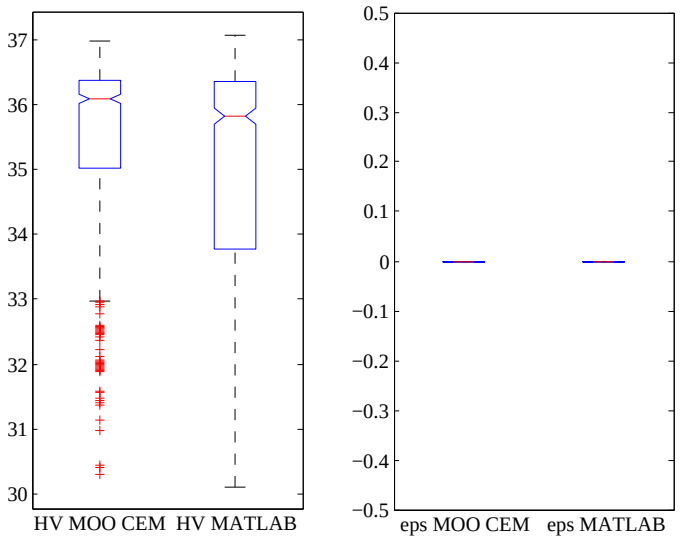

Figure C.25: Box-whisker plot for BAP17.

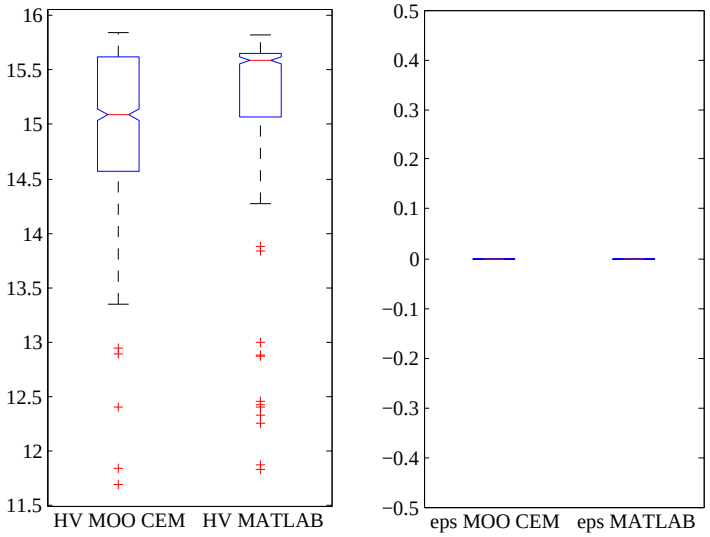

Figure C.26: Box-whisker plot for BAP18.

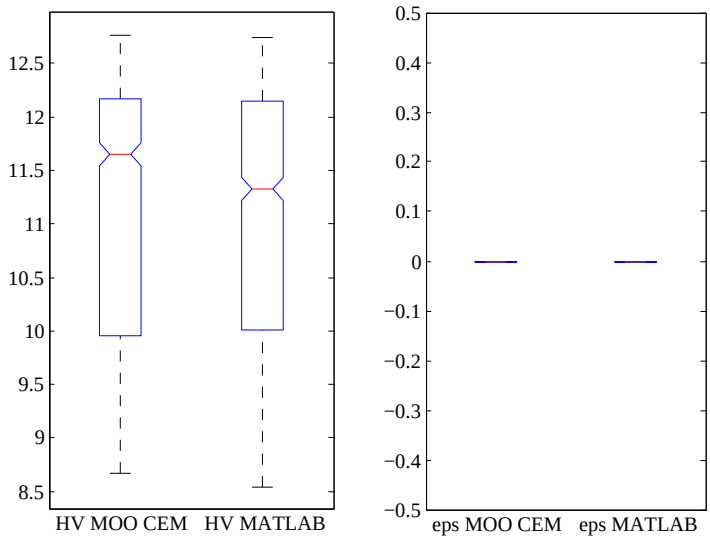

Figure C.27: Box-whisker plot for BAP19.

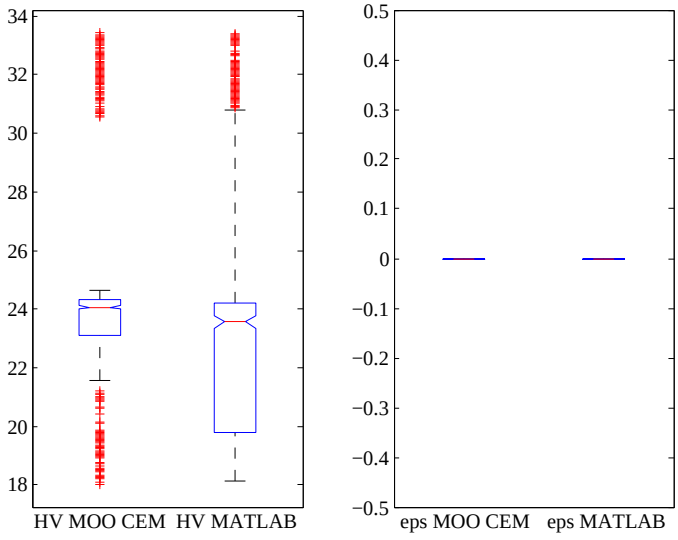

Figure C.28: Box-whisker plot for BAP20.

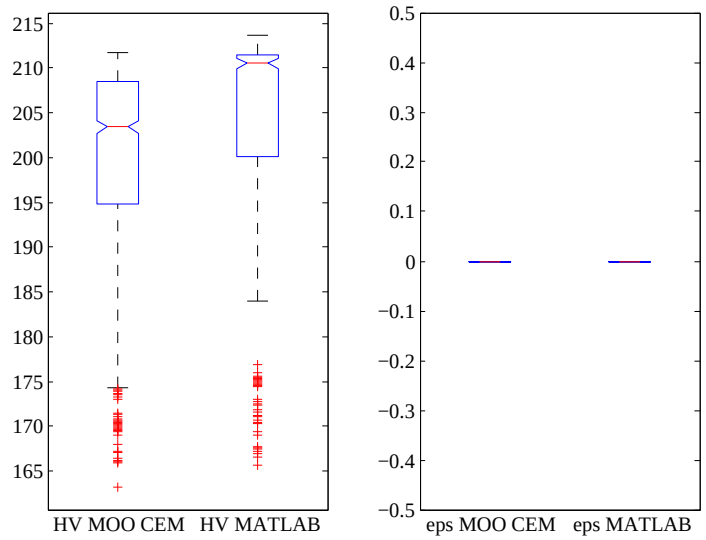

Figure C.29: Box-whisker plot for BAP21.

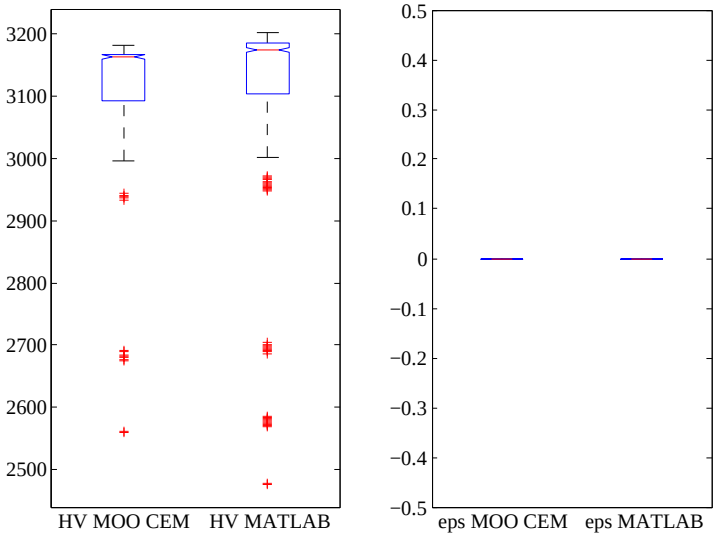

Figure C.30: Box-whisker plot for BAP22.

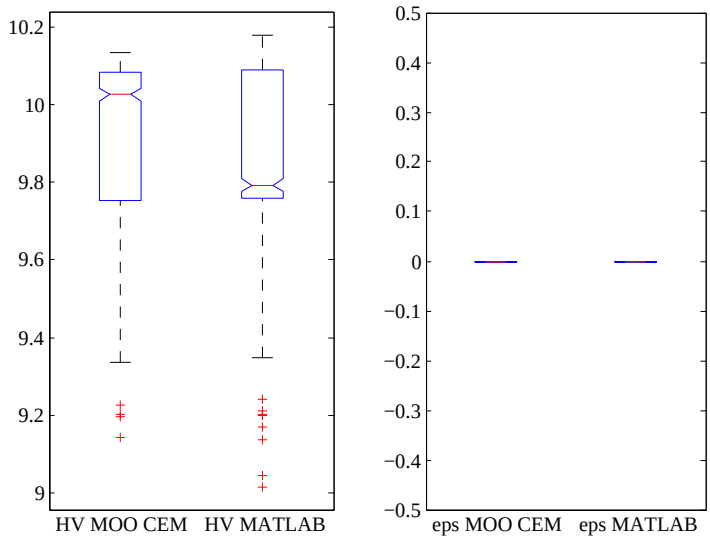

Figure C.31: Box-whisker plot for BAP23.

The box-whisker plot for the (s, S) inventory problem discussed in Section 5.1 is shown in Figure C.32.

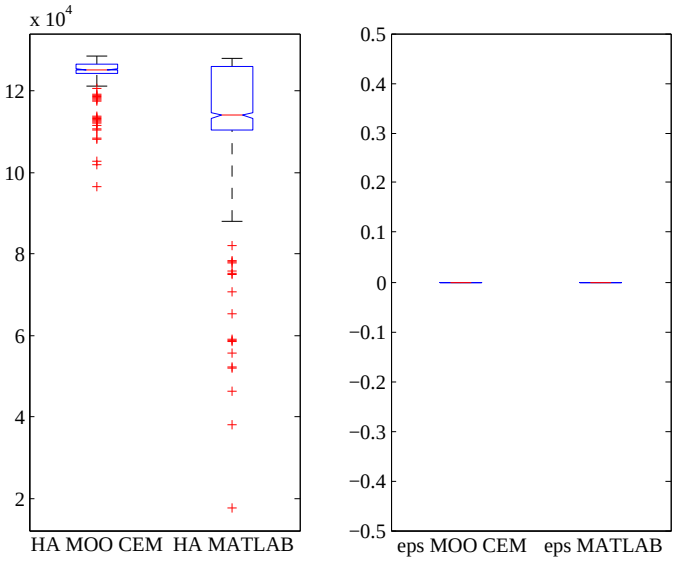

Figure C.32: Box-whisker plot for the (s, S) inventory problem.

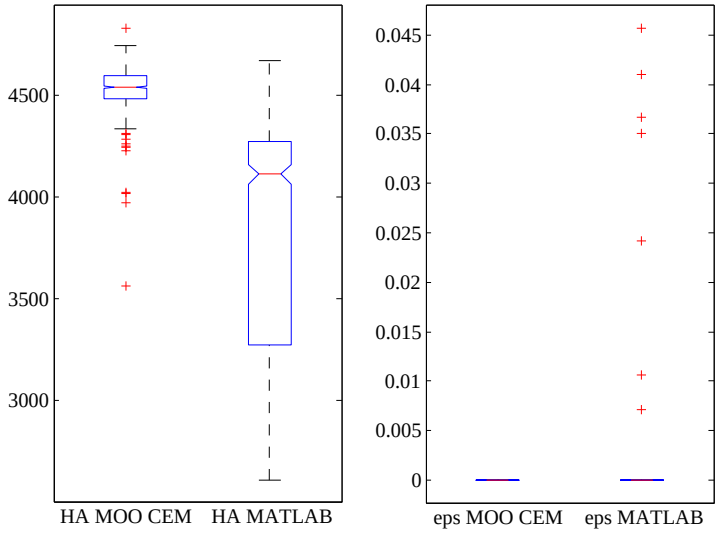

Figure C.33: Box-whisker plot for the reconfigurable manufacturing problem.

The box-whisker plot for the reconfigurable manufacturing system discussed in Section 5.3 is shown in Figure C.33.

The box-whisker plot for the CO gas problem discussed in Section 5.5 is shown in Figure C.34.

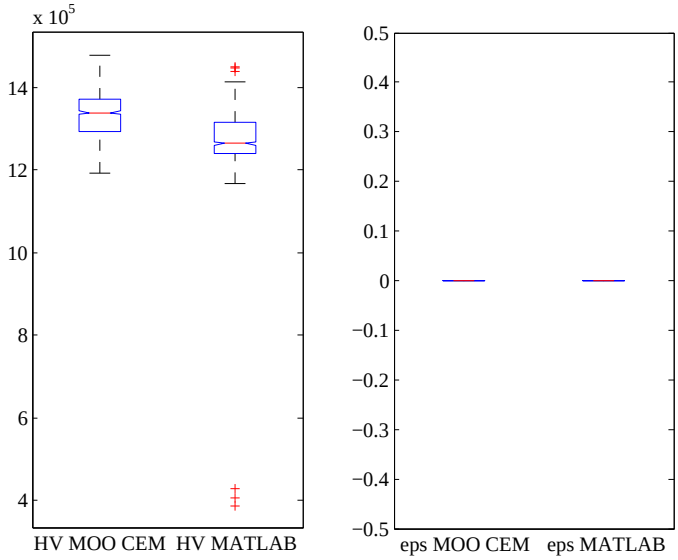

Figure C.34: Box-whisker plot for the CO gas problem.

APPENDIX *D*

IMPLEMENTATION GUIDELINES

Specific aspects of implementation are discussed in this appendix, with the focus on the integration of the two software packages Matlab® and Arena®. It is assumed that the reader has a working knowledge of both, but this is not essential for understanding the implementation concepts. Specific guidelines for running the MOO CEM algorithm in Matlab® are given and the setup in Arena is also explained. In the last part of this appendix, some limitations of the optimisation implementation are explained.

D.1 Integration of Matlab® and Arena®

The integration architecture of Matlab® and Arena® with respect to executing the MOO CEM algorithm for stochastic optimisation is shown schematically in Figure D.1. Matlab® executes the primary process of the MOO CEM algorithm, while Arena is used to evaluate candidate solutions by taking values of decision variables proposed by the MOO CEM algorithm. These values are input to the simulation model in Arena, which return point estimators for the objectives to Matlab®. Data exchange is achieved via simple text files. The Matlab® code to activate Arena is shown below.

```
function Arena
    % Define a connection to Arena from Matlab.
    h = actxserver('arena.application');
    s = h.activemodel %Model must be loaded in Arena and ready to run.
    s.invoke('go');  %Invoke the Arena method 'go'.
    s.invoke('end'); %Invoke the Arena method 'end', after the required
                     %number of replications was executed.
end
```

Some requirements must be met in order to execute the two integrated packages. These are discussed next.

D.2 Requirements for executing the optimisation

Before the optimisation model can be executed, some specific requirements must be met. These are:

1. The Arena model must be verified and validated, and set up to run sufficient replications (including a warm-up period, if necessary), for a given set of decision variables. These variables are defined in the usual way in Arena under its Variables data sheet.

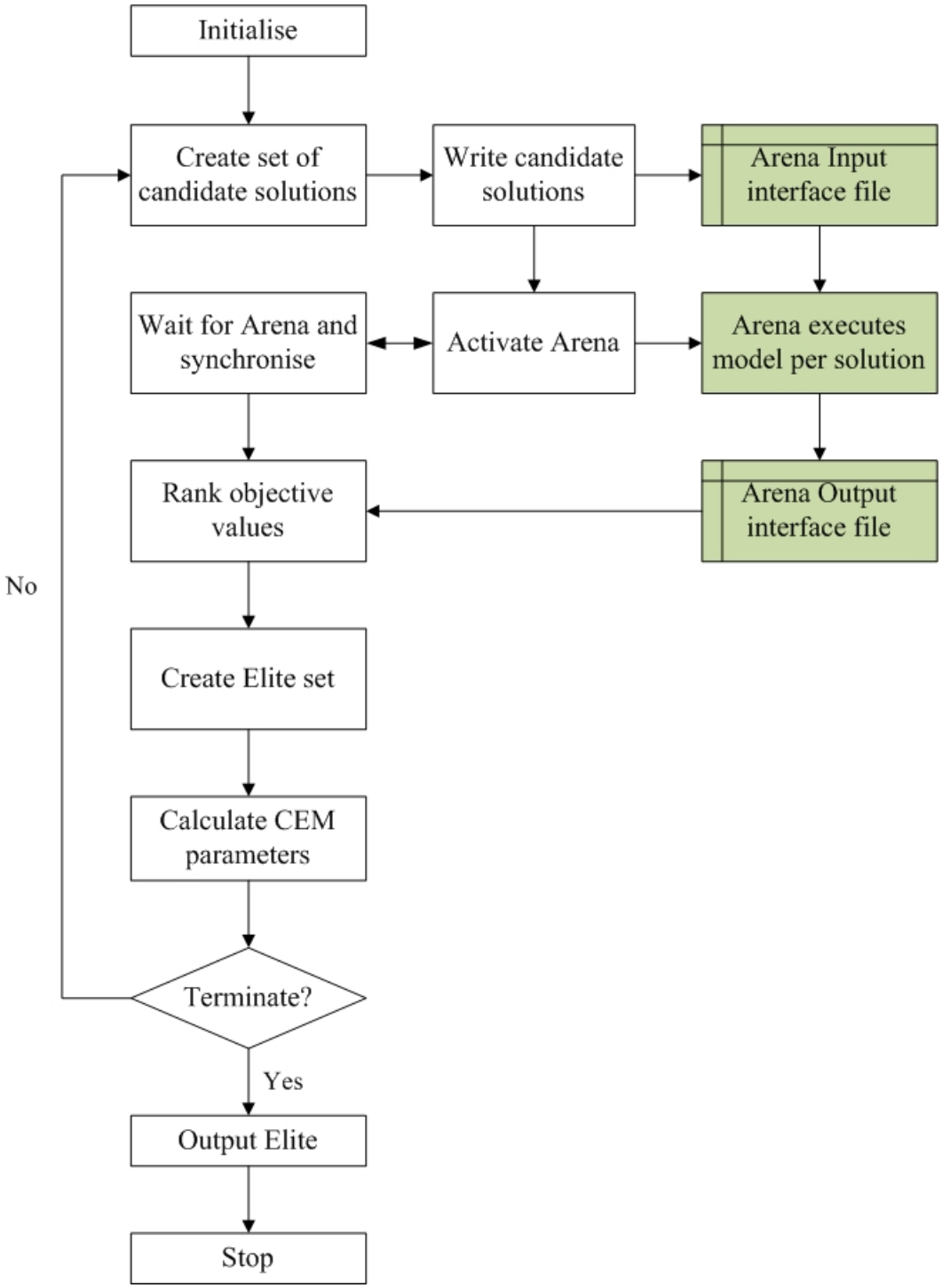

Figure D.1: Schematic architecture: Matlab® and Arena® integration.

2. In Matlab®, the algorithm settings must be defined, or default values can be accepted. The smoothing parameter α is set equal to 0.7, but the value can be changed. Also, the maximum number of evaluations is set to 10 000, but the number can be changed, while the termination value ϵ_c is typically set to 0.1. The probability to invert a histogram is set to $p_h = 0.3$.

3. In the Matlab® function `InitializeProblem`, the problem is initialised by defining a vector consisting of the problem number (which can be 0), the number of decision variables D, the number of objectives K, and finally default values for the lower limit and upper limit of the decision variables. The vector is named `MOP_Config`, and a typical setting would be: [1 2 2 −4 4], which means that Problem Number 1 has two decision variables, two objectives, and both decision variables are defined on the range $[-4, 4]$. If the decision variables have different ranges, the values must be assigned via Matlab® code to the array variable $L(D, 2)$. This variable has D rows (the number of decision variables), and two columns, with column 1 containing the lower limit of decision variable i and column 2 containing the upper limit. A problem with two decision variables with different ranges will thus have a 2×2 array to define its limits, e.g. for $-2 \leq x_1 \leq 3$ and $-1 \leq x_2 \leq 1$:
$$\begin{bmatrix} -2 & 3 \\ -1 & 1 \end{bmatrix}$$

4. The result (the approximate Pareto set) is collected in the Matlab® variable `Elite`, and can be written via code to Microsoft Excel, for example, using the Matlab® command `xlswrite`. `Elite` contains a column with values for each decision variable and a column each for the values of the objectives. The number of rows in `Elite` represents the size of the approximate Pareto set.

Only two objectives can be optimised, since the Pareto sort algorithm was initially programmed to only handle two vectors. The number of decision variables is hypothetically unlimited. When large discrete solution spaces are explored, it is recommended that the continuous version of the MOO CEM algorithm be used, because the probability vector P in Algorithm 2 may otherwise become very large.

D.3 Matlab® code for the MOO CEM algorithm

The Matlab® code for the basic MOO CEM algorithm is shown below. This can be used for the continuous benchmark problems, while the objective functions can be adapted for other problems.

```
function MOO_CEM

%The workarea for the algorithm, including population of decision values,
%objective values and ranking
global WorkArea
%Elite vector preserving good solutions and driving the CEM
global Elite;
%The means vector of the CEM; the standard deviation vector of the CEM
global mu; global sigma;
%Variable to collect each Elite mean and std dev after an iteration.
%For info purposes, e.g. in thesis.
global SetMu; global SetSigma;

%The researcher can set this value to solve some standard MOO problems:
%1,2,3,4,6 in Coello Coello, and ZDT1, 2, 3.
MOP = 2;

%Problem parameters were coded in InitializeProblem to make the code
%more generic. "Limits" is used as a varying limit vector in the histogram
%assignments, while "L" is kept fixed. These initially have the same values
%(the limits of the decision variables).
[NumVars, NumObjectives, Limits, L, SheetName, ProblemN] = ...
    InitializeProblem(MOP);

%Clear all vectors:
WorkArea = []; mu=[]; Elite=[]; sigma=[]; SetMu=[]; SetSigma=[];

%The epsilon of the CEM. Set at 0.1-1.
eps = 0.1;

%Alpha of the CEM. Set at 0.7-0.8
alpha=0.7; %0.7

%Maximum number of outer loops. This loop supports the multi-objective
%search. In single-objective search, it would be one loop only.
NoOfLoops = 100;

%Population size. Set at 30-100.
N = 100;

%Probability to invert histograms to diversify search and to avoid trapping
Prob = 0.3;
```

```
MaxEvaluations = 10000;  %Try to achieve good solutions with "few" tries.

%If the researcher wants to try different random numbers.
%rand('twister', 156089);
%rand('state', 34089);
%rand('seed', 14089);

NumEvaluations = 0;
Elite=[];
tic  %Start clock, not necessary.

% ~~~~~~~~~~  MAIN LOOP ~~~~~~~~~~
for k=1:NoOfLoops
    k
    %Init the CE vectors. sigmaeps keeps track of the consecutive sigma
    %differences for convergence checking.
    sigmaeps(1:NumVars)=Inf;
    %CEM requires an arbitrary large initial sigma
    sigma(1:NumVars) = 10*(L(1:NumVars,2) - L(1:NumVars,1));
    %Select a random mu vlaue for each DV in its definition range
    mu(1:NumVars) = (L(1:NumVars,1) + (L(1:NumVars,2) - ...
        L(1:NumVars,1)).*rand(NumVars,1))';

    t=0;
    WorkArea = []; %Must clear WorkArea in each main loop

    NotTerminate = true;

    while NotTerminate  %Will be governed by convergence
        t=t+1;
        bin_freq = [];
        bin_edges = [];
        if size(Elite,1) > 0 && k>1
           r = k + 2;
           for i=1:NumVars
               bin_edges(1:r+1) = 0;
               bin_edges(1) = L(i, 1);
               bin_edges(r+1) = L(i, 2);
               bin_edges(2) = min(Elite(:, i));
               bin_edges(r) = max(Elite(:, i));
               bin_edges(2:r)= bin_edges(2):(bin_edges(r) - ...
                   bin_edges(2))/(r-2):bin_edges(r);
               bin_freq(1:r)   = histc(Elite(:,i), bin_edges(1:r));
               if rand<Prob
                  bin_freq(1:r) = max(bin_freq) - bin_freq(1:r);
               end
               bin_freq = floor(N*bin_freq./sum(bin_freq));
               s = sum(bin_freq(1:r));
               bin_freq(r) = N - sum(bin_freq(1:r)) + bin_freq(r);
               s=sum(bin_freq);
               UpTo=0;  %Indices into WorkArea
               %Now sample on each histogram bin range, with the number of
```

```
        %observations proportional to the histogram count
        for a=1:r
            Start = UpTo + 1;
            UpTo  = UpTo + bin_freq(a);
            Limits(i, 1) = bin_edges(a);
            Limits(i, 2) = bin_edges(a+1);
            if Start <= UpTo
               %This is not the same mu as mu(i).
               %It moves with the histogram range
               temp_mu = Limits(i,1) + rand*(Limits(i,2) - Limits(i,1));
               temp_s  = Limits(i,2) - Limits(i,1);
               WorkArea(Start:UpTo, i) = ...
                   rand(UpTo-Start+1, 1)*(normcdf(Limits(i,2), ...
                  temp_mu, temp_s) - normcdf(Limits(i,1), ...
                  temp_mu, temp_s)) + normcdf(Limits(i,1), ...
                  temp_mu, temp_s);
              %WorkArea(Start:UpTo,i) = log2(WorkArea(Start:UpTo,i));
               WorkArea(Start:UpTo, i) = ...
               norminv(WorkArea(Start:UpTo, i),...
                   temp_mu, temp_s);
            end
        end   %For each bin of the histogram of this i-th DV
    end       %For each DV
else          % Build initial population
    for i=1:NumVars
        %Select random values from the definition ranges of the DVs
        WorkArea(1:N, i) = rand(N, 1)*(normcdf(L(i, 2), ...
            mu(i), sigma(i)) - normcdf(L(i,1), mu(i), ...
                           sigma(i))) + normcdf(L(i,1),...
                           mu(i), sigma(i));
        WorkArea(1:N, i) = norminv(WorkArea(1:N, i), ...
            mu(i), sigma(i));
    end
end    %Elite exists or initial population must be formed

s = size(WorkArea, 1);
% +2 is for assistance with ranking
WorkArea(1:s, NumVars+NumObjectives+2) = zeros(s, 1);
%Call function "f1" to evaluate the DV values:
WorkArea(1:s, NumVars+1) = f1(WorkArea, NumVars, MOP);
%Call function "f2" to evaluate the DV values:
WorkArea(1:s, NumVars+2) = f2(WorkArea, NumVars, MOP);

%Rank the solutions and remove any "good" solutions.
%Here, the threshold is th=2 (see text)
Temp = Rank(WorkArea, 2, NumVars, NumObjectives);
%Add to Elite:
Elite = vertcat(Elite, Temp);

%It's nice to see the progress, but it can slow down the algorithm:
PlotDetailProgress(NumVars, WorkArea, ProblemN, NoOfLoops, t, k, N);
```

```
        %If good solutions were identified, Elite can be exploited.
        if size(Elite,1) > 1
            AllEps = 0;
            for i=1:NumVars
                %Adjust the mu's according to the CEM:
                mu(i)       = (1-alpha)*mu(i)     + alpha*mean(Elite(:,i));
                sigmaeps(i) = sigma(i);
                %Adjust the sigmas according to the CEM:
                sigma(i)    = (1-alpha)*sigma(i) + alpha*std(Elite(:,i));
                %Calculate the consecutive changes, to determine
                %convergence:
                sigmaeps(i) = abs(sigma(i) - sigmaeps(i));
                %If this i-th DV has converged, mark it:
                AllEps = AllEps + (sigmaeps(i) > eps);
            end
            %If they all converged, terminate inner loop:
            if AllEps == 0
                NotTerminate = false;
            end
            SetMu = vertcat(SetMu, mu);  %Record the current mu
            SetSigma = vertcat(SetSigma, sigma);  %and sigma for information
        end
        %Avoid trapping of the algorithm: allow no more than half the
        %maximum number of evaluations for the inner loop:
        NotTerminate = (NotTerminate && (N*t <= MaxEvaluations/2));
        if ~NotTerminate, break, end

        NumEvaluations = NumEvaluations + N;
        if  (NumEvaluations >= MaxEvaluations), break, end
    end  %while not Terminate

    NumEvaluations;
    %Rank again with th=1:
    Elite=Rank(Elite, 1, NumVars, NumObjectives);
    %Plot the intermediate Elite vector (approximation set), for
    %information purposes only:
    i = subplot(2,2,2);
    hold on
    scatter(Elite(:,NumVars+1), Elite(:, NumVars+2), 3, '*');
    xlabel('f1')
    ylabel('f2')
    title(['Current Pareto front, after iteration ' int2str(t)...
        '  ' int2str(NumEvaluations)])
    drawnow
    grid on
    hold off

    %Terminate outer loop if NumEvaluations was reached before NoOfLoops
    %was executed:
    if (NumEvaluations >= MaxEvaluations),  break, end
end % for k
```

```
%Final ranking, non-dominated values, th=0:
Elite=Rank(Elite, 0, NumVars, NumObjectives);
size(Elite)

toc
%Show the known Pareto front and the approximate front:
Plot_WorkArea(Elite(:,NumVars+1), Elite(:,NumVars+2),...
    MOP, SheetName, NumEvaluations)

end  % Main function MOO_CEM
%--------------------------------------------------------------------------


function Do_f1 = f1(X, NVars, MOP)
    if MOP == 1
       Do_f1=X(:,1).^2;
    elseif MOP == 2
       rt = 1/sqrt(NVars);
       Do_f1=1-exp(-((X(:,1)-rt).^2+(X(:,2)-rt).^2+(X(:,3)-rt).^2));
    elseif MOP == 3
       A1 = 0.5*sin(1)-2*cos(1)+  sin(2) - 1.5*cos(2);
       A2 = 1.5*sin(1)-  cos(1)+2*sin(2) - 0.5*cos(2);
       B1 = 0.5*sin(X(:,1)) - 2* cos(X(:,1)) + sin(X(:,2)) - 1.5*cos(X(:,2));
       B2 = 1.5*sin(X(:,1)) -    cos(X(:,1)) + 2*sin(X(:,2)) - 0.5*cos(X(:,2));
       Do_f1 = -(1 + (A1 - B1).^2 + (A2 - B2).^2);
    elseif MOP == 4
       Do_f1 = -10*(exp(-0.2*sqrt(X(:,1).^2 + X(:,2).^2)) +...
           exp(-0.2*sqrt(X(:,2).^2 + X(:,3).^2)));
    elseif MOP == 6
       Do_f1=X(:,1);
    elseif MOP >= 8
       Do_f1 = X(:,1);  %ZDT1, 2 & 3
    end
end
function Do_f2 = f2(X, NVars, MOP)
    if MOP == 1
       Do_f2=(X(:,1)-2).^2;
    elseif MOP == 2
        rt = 1/sqrt(NVars);
        Do_f2=1-exp(-((X(:,1)+rt).^2+(X(:,2)+rt).^2+(X(:,3)+rt).^2));
    elseif MOP == 3
       Do_f2 = -((X(:,1) + 3).^2 + (X(:,2) + 1).^2);
    elseif MOP == 4
       Do_f2   = abs(X(:,1)).^(0.8)+ 5*sin((X(:,1)).^3) + abs(X(:,2)).^(0.8)...
           + 5*sin((X(:,2)).^3) + abs(X(:,3)).^(0.8) + 5*sin((X(:,3)).^3);
    elseif MOP == 6
        x=X(:,1)./(1+10*X(:,2));
        y=x;
        x=x.^2;
        x=1-x;
        Do_f2=(1+10*X(:,2)).*(x - y.*sin(12*pi*X(:,1)));
    elseif MOP >= 8 && MOP <= 10 %ZDT1-3
```

```
        c = 9/(NVars-1);
        x = transpose(sum(transpose(X(:,2:NVars))));
        gx = 1 + x.*c;
        gx_inv = 1./gx;
        if MOP == 8     %ZDT1
            Do_f2 = gx.*(1 - sqrt(gx_inv.*X(:,NVars+1)));
        elseif MOP == 9 %ZDT2
            Do_f2 = gx.*(1 - (gx_inv.*X(:,NVars+1)).^2);
        elseif MOP == 10 %ZDT3
            Ten_Pi = 10*pi;
            Do_f2 = gx.*(1 - sqrt(gx_inv.*X(:,1)) - ...
                gx_inv.*X(:,1).*sin(Ten_Pi*X(:,1)));
        end
    end
end

function PPF =Plot_WorkArea(x,y, MOP, xlSheetName, NEval)
subplot(2,2,4);
hold on;
if MOP <= 4 || (MOP >= 6 && MOP < 11)
    z=[]; %The true Pareto fronts provided by Coello Coello are in Excel
    z = xlsread('True_PFs_Coello.xls', xlSheetName);
    scatter(z(:,1), z(:,2), 5, 'v', 'filled');
end
scatter(x, y, 3, 'o');
hold off
grid on
xlabel('f1');
ylabel('f2');
title(['Final Pareto Front of MOP', int2str(MOP), ...
    ' after ', int2str(NEval), ' evaluations']);
end
function RankIt = Rank(Pop, Threshold, NVars, NObj)

K = NVars+NObj;
Pop(:,K+1)=0;

F=[];
Sp=[];
signe = -1; %+1 for MOP3

for z=NVars+1:K - 1
    Pop=sortrows(Pop, signe*(z));
    for p=1:size(Pop,1)-1
        for q=p+1:size(Pop,1)
            if Pop(p,z+1) >  Pop(q, z+1)  %Turn around for MOP3, maxim.
               %Rank is in last col.
               Pop(p, K+1) = Pop(p, K+1) + 1;
                %No need to look further, this candidate is not making it
               if Pop(p, K+1) > (NObj-1)*Threshold, break, end
            end
        end
```

```
        if Pop(p, K+1) <= Threshold
           F = vertcat(F, Pop(p, :));
        end
    end
    F = vertcat(F, Pop(size(Pop,1), :));   %Add the last vector to the set
end
RankIt  = F;

end

function [NumVars, NumObjectives, Limits, L, SheetName, ProblemN] = ...
    InitializeProblem(MOP)

MOP_Config = [1 1 2 -1E5 1E5, %MOP1
              2 3 2 -4 4,     %MOP2
              3 2 2 -pi pi,   %MOP3
              4 3 2 -4 4,     %MOP4
              5 0 0 0 0,
              6 2 2 0 1,      %MOP6
              7 0 0 0 0,
              8 30 2 0 1,     %ZDT1
              9 30 2 0 1,     %ZDT2
              10 30 2 0 1     %ZDT3
              ];

NumVars       = MOP_Config(MOP, 2);
NumObjectives = MOP_Config(MOP, 3);

for i=1:NumVars    %Set problem boundaries
    L(i,1) = MOP_Config(MOP, 4);
    L(i,2) = MOP_Config(MOP, 5);
end

ProblemName = ['MOP1 ', 'MOP2 ', 'MOP3 ', 'MOP4 ', 'MOP5 ', 'MOP6 ',...
    'MOP7 ', 'ZDT1 ', 'ZDT2 ', 'ZDT3 '];

for i=1:NumVars
    Limits(i,1) = L(i,1);
    Limits(i,2) = L(i,2);
end;

ProblemN = ProblemName((MOP-1)*5+1:5*MOP);
SheetName = ProblemN(1:4);
end %function InitializeProblem

%PlotDetailProgress
function PlotDetailProgress(NumVars, WorkArea, ProblemN, NoOfLoops, t, ...
    k, N)
  h = subplot(2,2,[1 3]);
  hold on
  scatter(WorkArea(1:N, NumVars+1), WorkArea(1:N, NumVars+2), 3, '*');
  xlabel('f1')
```

```
  ylabel('f2')
  title(['Search space for ' ProblemN ': k=' int2str(k)...
      ' of ' int2str(NoOfLoops) ', t=' int2str(t)]);
  drawnow
  grid on
  hold off
end
```