ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

TIK Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Student Thesis - Winter Term 2006/2007

Robustness and Uncertainty in Multiobjective Search

Marianne M. Michel
mmm@ee.ethz.ch

January 31, 2006

Professor:	Eckart Zitzler	zitzler@tik.ee.ethz.ch
Tutors:	Eckart Zitzler,	
	Tim Hohm	hohm@tik.ee.ethz.ch

Abstract

For many multi-objective optimization scenarios, the goal is usually to identify the set of Pareto-optimal solutions. In practice however, some problems might arise in this context. In many cases, design variables can be prone to variation, which may cause the focus to shift to finding so-called robust solutions - solutions which are the least sensitive to slight changes in variables. In addition, real-world optimization problems are often subject to a wide range of uncertainties, which can be caused by stochastic models, for example.

In this study, these aspects are discussed and an approach is presented which allows for both robustness and uncertainty to be taken into account. A concept is developed, building upon existing studies and is implemented in the form of an evolutionary algorithm situated in the PISA framework, which is adapted to allow the handling of robustness and uncertainty.

In order to assess the developed algorithm, a testproblem was implemented which allows the effects of robustness to be visualized. The simulation showed that the optimizer successfully finds the robust solutions if the according parameters are chosen wisely.

Contents

List of Figures

List of Tables

1

Introduction

This chapter provides an introduction to this thesis. Section 1.1 introduces and motivates the topics and concepts which are central to this study and, in Section 1.2, related approaches are presented. Section 1.3 sets the focus for this thesis and states the official task description of the project. Finally, in Section 1.4, the structure of this report is explained.

1.1 Motivation

Regarding many optimization problems, one will find that in most cases it is not one, but multiple objectives which want to be optimized. Usually these criteria compete against each other, which makes you need a multi-objective optimization procedure in order to find the optima.
If the goal is to design an embedded system, for example, one will want to keep the costs and size low while pushing efficiency and reliability to its limits. Evolutionary Algorithms have proven to be a good choice to find the set of optimal solutions.

There are, however, a few problems one can encounter. One aspect which has to be regarded is robustness. In many fields, certain variables can be prone to variation. E.g. in production the properties of a product's components might vary in size or weight as a consequence of a tolerance in precision. Here one would primarily be interested in finding those compositions, whose quality is the least sensitive to this variation, which would be the most robust solutions.

An other aspect that can come to one's attention is uncertainty. Optimization problems are often subject to uncertainties, due to stochastic objective functions, for example.
These aspects of robustness and uncertainty, and how they can be handled in multi-objective optimization, are central to this thesis.

This chapter will give you an overview of these topics, providing the motivation for this project.

1.1.1 Optimization

Nowadays, real-world problems most often involve the need to simultaneously optimize several, often conflicting, objectives. In this text, we assume that 'to optimize' means to *minimize* the values of these objective functions.

The goal in such scenarios, is usually to identify the set of Pareto-optimal solutions: the Pareto front. A solution $\mathbf{x}^*$ is said to be Pareto-optimal, if all other vectors $\mathbf{x} \in X$ have a higher value for at least one of the objective functions $f_i(\mathbf{x})$, or else have the same value for all objectives.

In this context, the *decision space*, containing the elements among which the best is sought, is denoted by X. These elements are called solutions or decision vectors. The *objective space*, denoted by Z, contains the objective vectors.

There are several different techniques which can be applied to an optimization problem. Almost all of them combine the multiple objectives into one scalar objective, whose solution is a Pareto-optimal point for the original multi-objective optimization problem. In the following text, this single objective is denoted as the *fitness function* $F(\mathbf{x})$.

What makes some optimization problems hard to solve, is that they can be too complex to be solved using deterministic techniques, or that the objective functions are highly complex or poorly understood. Another possibility is that the objective functions are not given in closed form, but have to be determined by simulation or experiment.

1.1.2 Evolutionary Algorithms

In the late 50s, an idea started to evolve that was totally new in the field of optimization. Researchers began to realize that, as the process of natural evolution can be seen as a procedure to optimize species in order to

increase their expectation of survival, one can apply this concept to any other optimization problem.

In the mid 60s, Ingo Rechenberg and Hans-Paul Schwefel were working on the development of a two phase flashing nozzle at the Technical University of Berlin [6]. When they couldn't satisfactorily optimize its aerodynamic properties in a mathematical way, they chose an alternative approach based on selection and random variation (achieved by throwing dice to generate potential new solutions), thereby borrowing the approach of evolutionary optimization from nature. The outcome exceeded their expectations and the concept has been subject to extensive further research ever since.

The Basic Idea

The workflow of an example Evolutionary Algorithm is shown in Table 1.1.

One starts by randomly choosing a set of solutions and adding them to an initial population. Then, the fitness values are calculated and assigned to every solution in the population.

The actual generation cycle starts when a set of parents is chosen from the current population, creating a mating pool. When mutation and recombination are performed on this set of parents, a new pool is created consisting of the offspring. The next step is to evaluate the newly generated solutions and to exchange the best individuals in the offspring with the worst solutions in the population. The described cycle is repeatedly performed, until a stopping criterion is reached.

Evolutionary Algorithms exist in many different variations but the underlying scheme is always the same. These algorithms have proven to be very effective when applied to optimization problems, especially to those that are too complex to be solved using deterministic techniques. With the presented workflow (Table 1.1) though, it is not possible to consider the mentioned aspects of robustness and uncertainty, which will be covered in the next paragraph.

1.1.3 Robustness

In practice, the properties of a solution may be subject to a certain amount of variation because its implementation cannot be realized with arbitrary

Table 1.1: An Evolutionary Algorithm

Evolutionary Algorithm

Step 1:	**Initialization:** *Randomly choose initial solutions* $x_1, x_2, ..., x_N$ *from* X *Set iteration counter* t *to* 0
Step 2:	**Fitness assignment:** *Calculate the fitness values for all solutions* **x** *in the population* P
Step 3:	`loop` *Increment iteration counter* t **Mating selection:** *perform tournament on* M *and select parents*
Step 4:	**Variation:** *Apply recombination and mutation operators to the mating pool, assign fitness to the resulting offspring and add these to* P

Step 5:	**Termination:**
	`if` $t \geq t_{max}$
	output best solution in p *and STOP*
	`end if`
	`end loop`

precision. This can occur due to various reasons, such as precision tolerances in production or natural variation (e.g. a banana's weight and size will differ from one banana to an other).

Now, let's consider instances of a certain solution which have slightly differing decision variable values. If these instances are evaluated, the corresponding objective vectors might differ widely, even though the variation in the decision space is not huge. In this case, we would say that the solution is sensitive to the variation in decision variables.

Figure 1.1 shows two solutions in the decision and objective space, where solution A is less sensitive to variable perturbations than solution B (source: [4]).

In such scenarios, it may be desirable to pick only those solutions, which are the least sensitive to these perturbations - robust solutions, so to say. This concept is best illustrated applied to an example, as in Table 1.2.

1.1.4 Uncertainty

Many real-world optimization problems are subject to a wide range of uncertainties. Possible reasons for these uncertainties could be noisy objective functions or approximation errors.

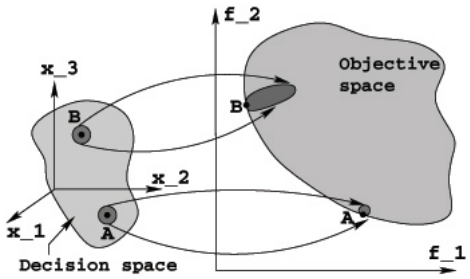

Figure 1.1: Robust and non-robust solutions

Table 1.2: Robustness in a car tyre

Problem:	You want to produce a car tyre which consists of two types of rubber. The tyre should be durable and have a good adhesive strength. To achieve this goal, you need to know in which ratio the two rubber types should be mixed.
Decision variables:	Rubber types A and B.
Objectives:	Durability and Adhesion.
Solution:	In this case, a robust solution (or tyre) would be one, whose quality, consisting of durability and adhesion, does not differ greatly if the ratio of rubber types A and B varies slightly.

In such cases, these uncertainties have to be considered and additional measures taken, in order to make sure that the set of optimal solutions is found. This is essential, to be able to make a prediction on the effective quality of solutions, and to make sure evolutionary optimizers work correctly. For example, if a solution is evaluated only once and the assigned fitness value is bad, this solution will probably be discarded in an optimization run, in which uncertainty is not considered. This means that the solution is lost, even though the same solution might have had excellent values in ninety-nine evaluations out of a hundred - which, in most cases, would make it a good solution.

Uncertainty can in this context be described as a variation in objective space. Figure 1.2 shows a solution in the decision and objective space. The variance in objective function values emerges due to uncertainty.

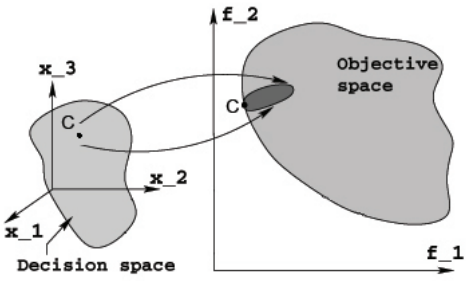

Figure 1.2: Uncertainty: variation in the objective space

1.2 Related Work

While some studies have addressed the mentioned topics, a great deal of this research was situated in the context of single-objective optimization.

Jin and Branke [5] published a study on uncertainty, providing an overview of the related work and discussing existing approaches. They also investigated the relationship between different categories of uncertainties.

Most of the published approaches concerning robustness propose the averaging of objective values. An example for this approach is the first of two robust multi-objective optimization techniques proposed by Deb and Gupta [4]. In order to consider robustness, they generate a set of neighboring solutions for every initial individual, distributed according to a predefined grid. Then, the mean effective objective vector is calculated and used for the fitness assignment.

Recently, Basseur and Zitzler [1] presented a technique to handle uncertainty in multi-objective optimization, which does not make any assumptions on the type of distribution representing the uncertainty. In order to find a Pareto set approximation, their approach includes the comparison of sets of solutions, instead of comparing single solutions. This approach was based on the work of Zitzler and Künzli [2], in which a quality indicator was introduced, allowing preference information of the decision maker to be integrated in the search. These two studies mentioned last, provide the basis for the approach which was chosen in this thesis, and they will be described in more detail in Section 2.2.

The approaches mentioned above leave some questions open, however.

a) The technique of averaging objective vectors, which is used in many approaches considering robustness, causes the quality of the solutions to depend on the distribution of the neighborhood in the decision space. If one wants to search for robust solutions which are subject to a variation in decision space that is distributed in a way so that the average of objective vectors is not a meaningful measure, the optimization procedure will not be successful.

b) While the approach which Basseur and Zitzler [1] used to account for uncertainty was very successful and flexible, the developed algorithm was never implemented in PISA (see Section 2.3). Since this technique represents the basis for the part of the approach chosen in this thesis involving uncertainty, its implementation was one of the tasks that had to be tackled in this study.

c) To this date, no studies have come to the authors attention which discuss a way to handle both the aspects of robustness and uncertainty in multi-objective optimization.

1.3 Focus of this Thesis

In the previous sections the concept of robustness and uncertainty was described. Especially for real-world optimization problems and applications which often deal with a certain variation induced by measurement tolerances or inaccuracies in production for example, the benefit of an optimization procedure which is able to deal with these aspects seems huge.

Now the question which is addressed in this study is exactly how to handle robustness and uncertainty in multi-objective optimization. In the context of this thesis the following tasks had to be tackled in order to find the answers to this question.

1. **Concept**
   On the basis of [1] and [2] a concept had to be developed which allows for the integration of robustness and uncertainty in evolutionary multi-objective search. Different approaches should be considered and a procedure picked which unites both aspects.

2. **Adaptation of PISA**
   PISA [3] defines a standardized interface allowing a separation of the problem-specific part (variator) of an optimizer from the problem-independent part (selector). So far, this interface had not yet supported the handling of robustness or uncertainty, which is why it should be adapted accordingly, thereby preserving downward compatibility.

3. **Implementation**
   An algorithm which incorporates the developed concept should be implemented in the PISA framework, which included creating a selector and a variator on the basis of existing modules.

4. **Testfunction**
   In order to allow for the visualization of the effect robustness and uncertainty have on the outcome of an optimization run, a testfunction had to be developed, which was to be integrated in the developed variator.

5. **Evaluation**
   The designed concept, the efficiency of the algorithm and the correct working of the PISA interface had to be tested and the value of the chosen approach verified. Further, a simulation for different parameter values should be conducted, including an evaluation of the gained results. Possible issues should be identified and described in order to formulate possible further goals in research.

The mentioned PISA framework, as well as the previous work the developed concept is based upon, are addressed in more detail in Chapter 2.

1.4 Overview

The remaining chapters of this report are designed as follows.

Chapter 2 provides the background for this thesis. Previous work, the developed concepts are based upon, is presented and here you will also find a description of the notation used throughout this report.

In Chapter 3 the actual approach that was taken to achieve the formulated goal is described, as well as the implementation in PISA. The adaptation of the PISA framework, which was needed in order to allow support of processes which provide the handling of robustness and uncertainty, is also discussed here.

Chapter 4 presents the developed testfunction which is used to simulate the behaviour of the implemented algorithm, when searching for robust solutions. Here, you will also find detailed plots produced during simulation and a discussion of the results.

Finally, a conclusion of this study is presented in Chapter 5, including the outcome of the study, as well as the drawbacks of the applied approach, followed by a section in which several interesting possibilities of further research based on the presented work are described.

2

Background

This chapter provides the essential background information for this thesis. In Section 2.1 you will find a description of the notation which is used throughout this report. Sections 2.2 and 2.3 will then present the former approaches which the concept and algorithm developed in this study are based upon, in addition to the framework in which the actual algorithm was implemented.

2.1 Notation

In the following text, the decision space and objective space are denoted by X and Z, respectively. An element $\mathbf{x} \in X$ is called a solution or individual and $Z = \mathbb{R}^n$ contains the objective vectors $\mathbf{f}(\mathbf{x})$, which are to be minimized.

The population size is denoted by α, the number of selected parents and the size of the offspring by μ and λ, respectively. S is the set of Pareto-optimal solutions, which is to be identified.

2.2 IBEA and the introduction of uncertainty

In 2002, Zitzler and Künzli [2] presented an approach which allows the decision maker's preference information to be directly integrated into the search, by defining the optimization goal in terms of a quality indicator. They proposed an indicator-based evolutionary algorithm (IBEA) that does not require any additional diversity preservation techniques to be employed. The algorithm outline can be found in Table 2.1.

A binary quality indicator is used to compare Pareto set approximations, which allows a measure for the 'loss in quality' if a solution $\mathbf{x}$ is removed from the population to be defined. The fitness assignment used in IBEA is based on this measure and is given by

$$F(\mathbf{x}^*) = \sum_{\mathbf{x} \in S \setminus \{\mathbf{x}^*\}} -e^{-I(\{\mathbf{x}\},\{\mathbf{x}^*\})/\kappa}.$$

The binary additive ϵ-indicator - an example quality indicator that is also used in the approach presented in this thesis - is defined as follows.

$$I_{\epsilon^+}(A,B) = \min_\epsilon \{\forall \mathbf{x}^2 \in B \quad \exists \mathbf{x}^1 \in A : f_i(\mathbf{x}^1) - \epsilon \leq f_i(\mathbf{x}^2) \text{ for } i \in \{1,2,...,n\}\}.$$

It defines the minimum distance, by which a Pareto set approximation can or must be translated in each dimension such that another set is weakly dominated.

Table 2.1: The Basic IBEA algorithm

Basic IBEA Algorithm

Input:	N	*(population size)*
	G	*(maximum number of generations)*
	κ	*(fitness scaling factor)*
Output:	S	*(Pareto set approximation)*

Step 1: **Initialization:** *Generate an initial population S of size α; set the generation counter g to 0.*

Step 2: **Fitness assignment:** *Calculate the fitness values of all $\mathbf{x}$ in S, i.e., $Fit(\mathbf{x}^*) = \sum_{\mathbf{x} \in S \setminus \{\mathbf{x}^*\}} -e^{-I(\{\mathbf{x}\},\{\mathbf{x}^*\})/\kappa}$.*

Step 3: **Environmental selection:** *While $|S| > \alpha$, remove the individual $\mathbf{x}^* \in S$ with the worst fitness value (i.e. $Fit(\mathbf{x}^*) \leq Fit(\mathbf{x}) \ \forall \ \mathbf{x} \in S$) from S and update the fitness values of the remaining individuals (i.e. $Fit(\mathbf{x}) = Fit(\mathbf{x}) + e^{-I(\{\mathbf{x}^*\},\{\mathbf{x}\})/\kappa} \ \forall \ \mathbf{x} \in S$).*

Step 4: **Termination:** *If $g \geq G$ return S.*

Step 5: **Mating selection:** *Perform binary tournament selection with replacement on S in order to fill the temporary mating pool.*

Step 6: **Variation:** *Apply recombination and mutation operators to the mating pool and add the resulting offspring to S. Increment the generation counter ($g = g + 1$) and go to Step 2.*

2.2.1 An Indicator-Based Model for Uncertain Environments

Basseur and Zitzler [1] recently extended the above-mentioned approach, providing the ability to handle uncertainty. A solution is considered to be associated with an unknown probability distribution in the objective space, which makes this approach very flexible. An objective function is regarded as a randomized procedure, which means that every evaluation of a certain solution $\mathbf{x}$ returns a different objective vector.

The fitness of an individual is defined as the expected indicator value which corresponds to the expected loss in quality, if the individual is removed from the population. Since the probability distributions are assumed to be unknown, this value cannot be calculated directly, but has to be estimated instead. To this end, several techniques were presented, three of which are also used in this thesis and are therefore described in Section 3.1.2.

In order to estimate the probability of a solution $\mathbf{x}$ to be mapped to any objective vector in Z, a finite sample of objective vectors is considered for every $\mathbf{x}$.

2.3 PISA

The interface specification PISA (**P**latform and Programming Language Independent **I**nterface for **S**earch **A**lgorithms) was introduced in 2004 by Bleuler, Laumanns, Thiele and Zitzler [3]. The goal they wanted to achieve was to design a "standarized, extendible and easy to use framework for the implementation of multi-objective optimization algorithms" ([3]), which separates the application specific and the optimizing part of a multi-objective optimization procedure.

The idea behind this concept is that the optimizing side does not need any problem specific information (such as the representation of the solutions) to function correctly and vice versa. The result is a simple and flexible solution, which allows for optimizers to be tested on different testproblems and for the application of different strategies on one testproblem. In this context, the problem specific part is called a *variator* and the optimizing part a *selector*.

The availability of such variator and selector modules, implemented for PISA, was crucial for the success of this project. To this end, a Website

is maintained at `http://www.tik.ee.ethz.ch/pisa/` where implemented modules are available for download.

The control flow and data flow specification of PISA is shown in Figure 2.2 (source: [3]). The transitions (rectangular boxes) represent the operations and the circles correspond to the control and data flow, the files which both modules read and write.

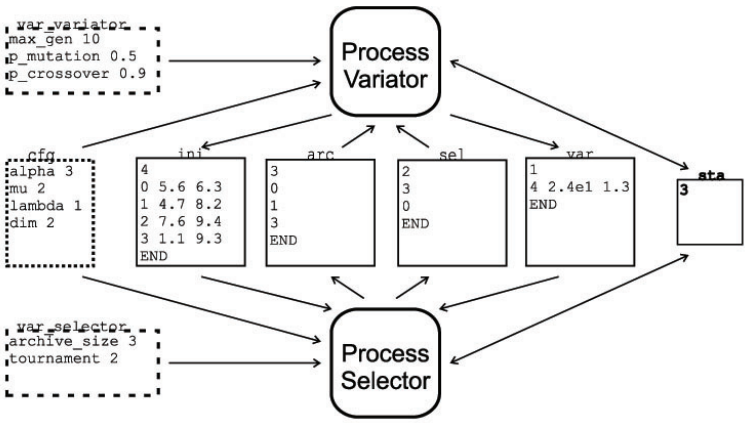

Figure 2.1: PISA: data files

Figure 2.1 shows the data files which are needed for the communication of the modules (source: [3]). The data flow consists of the initial population in `ini`, the archive of the selector in `arc`, the sample of parent individuals in `sel` and the offspring in `var`. The `cfg` file contains the common parameters and `sta` contains the state variable.

The files containing the information on the individuals (`ini` and `var`) consist of following entries. The first element specifies the number of data elements following. Then, the properties of each individual are added on a separate line: the index and objective function values. The last entry is an `END` tag, which is needed to detect file corruption.

In the specification of PISA [3], handling of robustness and uncertainty was not yet supported, so, in order to introduce these aspects and implement our approach, the specification of the interface had to be adapted as described in Section 3.2.

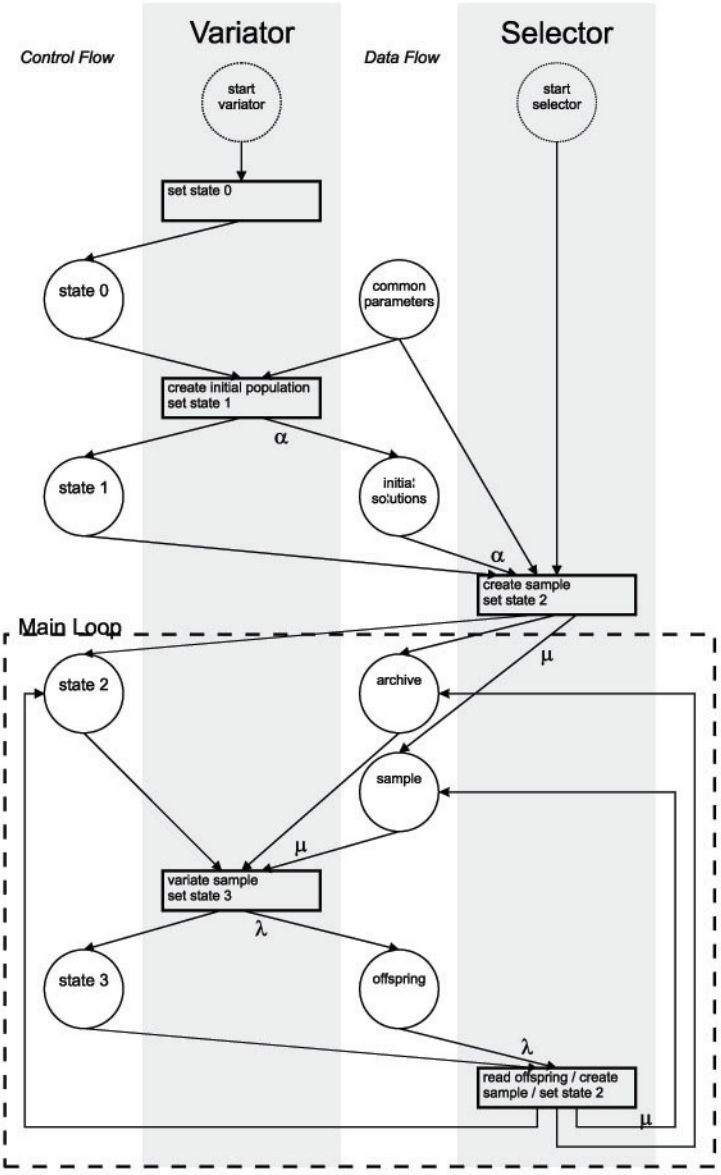

Figure 2.2: PISA: control flow and data flow specification

3

Approach

The following chapter describes the approach that was taken in order to achieve the goals of this study, as formulated in Chapter 1. In addition, the implementation of the developed concept is presented - an algorithm which was implemented in the PISA framework. Section 3.2 will then focus on describing the mentioned framework, as well as the adaptation which had to be performed, in order to allow the handling of robustness and uncertainty in PISA.

3.1 Handling Robustness

After considering different approaches, an elegant combination was chosen which allows robustness and uncertainty to be handled similarly.

3.1.1 Basic Concept

For the introduction of robustness, an approach is taken which is similar to the one described in [4].

In order to describe the robustness of an individual $\mathbf{x}$, a sample of possible values for $\mathbf{x}$ can be regarded in the decision space and evaluated in the objective space. The assigned objective vectors then represent a set of possible values for the individual, and the variation in the objective space corresponds to the solution's degree of robustness.

A set of possible decision vectors can be represented by a cloud composed of an individual and its neighbors in decision space. In the following

text, this cloud will be denoted by $C(\mathbf{x})$ and c will be the size of the cloud (including the initial individual). The generation of these clouds can be described as follows.

1. For every individual in the initial population, a cloud $C(\mathbf{x})$ is generated by subsequently picking neighbors on a specified interval cp around $\mathbf{x}$, according to a specified distribution (for the implementation in this study a uniform distribution was chosen).

2. Each cloud member is then evaluated and assigned an objective vector. The result is yet another cloud in the objective space, containing a set of possible objective vectors for every initial individual $\mathbf{x}$.

3.1.2 Combining Robustness and Uncertainty

In order to introduce uncertainty, much the same approach was chosen as presented in [1], where uncertainty is described by considering a sample of objective vectors that is drawn for every solution.

After executing the two steps described above, our approach therefore continues as follows.

3. For every individual and its neighbors, a sample $S(\mathbf{x})$ of objective vectors is drawn. The number of evaluations per solution is given by the sample size s. The resulting objective vectors can be seen as a double cloud in the objective space

Figure 3.1: Cloud of samples in objective space

What remains to be done is to rate a solution's quality according to its robustness.

4. For every dimension and every cloud: if the diameter of the sample cloud exceeds the value specified for `delta` in that dimension, the cloud is scaled, so that the values that are 'too good' are assigned the worst value minus delta for the considered dimension. This provides a sort of lower bound on the quality of a solution (see Figure 3.2).

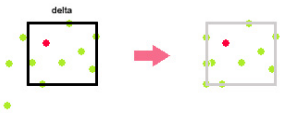

Figure 3.2: Scaling sample clouds according to delta

Figure 3.3 shows (from top left to bottom right): (a) one initial individual in the decision space, (b) the resulting cloud for 3 additionally sampled neighbors, (c) the corresponding sample cloud in objective space, when each cloud member is evaluated once, and (d) the resulting sample cloud if uncertainty is considered and each cloud member is evaluated four times.

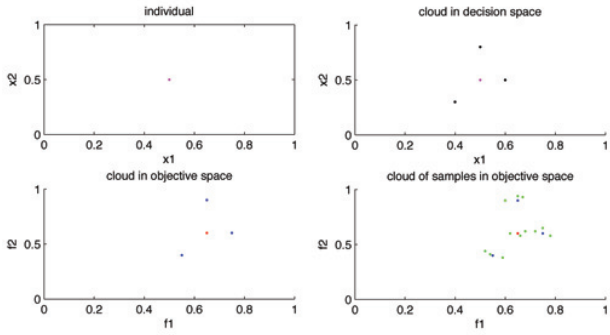

Figure 3.3: Clouds in decision and objective space

The implemented algorithm is shown in Table 3.1. The approach chosen for the handling of uncertainty is the same as Basseur and Zitzler [1] proposed. Instead of comparing single individuals, whole sets of solutions are compared, and the fitness value assigned to a solution is defined as the estimated expected loss in quality, if $\mathbf{x}$ were to be removed from the population S.

For the actual fitness calculation, several different methods were proposed in [1]. In our algorithm, one can choose between three of them: `avg`, which averages the objective vectors, `exp`, which directly computes the

fitness values, and `bck`, which approximates the ranking of the individuals induced by the 'estimated expected loss in quality' values. For details, please refer to [1].

3.2 Adaptation of PISA

Since the algorithm was to be implemented in PISA, and the specification did not yet include support for the handling of robustness or uncertainty, these aspects had to be introduced and the interface adapted.

In order to do so, the simplest approach was chosen, so as to best preserve downward compatibility. The states and state transitions were left exactly as they were. Instead, only the operations that are executed and the specification of the data that is exchanged, were extended for modules which support the consideration of robustness and uncertainty.

The control flow and data flow specification of PISA, including the added operations highlighted in red, is shown in Figure 3.4. In addition to creating the initial population, the variator will now also generate a cloud of neighbors for every solution in the population in state 0, if robustness is considered. If uncertainty is considered, all solutions in the population (including the cloud members) are evaluated multiple times, according to the specified value for the sample size `s`. The module then writes the generated population and its clouds to the `ini` file and set's state 1.

The selector reads the values from the `ini` file and scales the objective function values according to delta, as described in Section 3.1.2, before performing the fitness assignment. It then selects the parents and sets the state variable to 2.

The main loop is executed until a predefined stopping criteria is reached. Along with applying the mutation and recombination operators to the mating pool and generating the offspring, the operations in state 2 now include adding decision space clouds for each newly created individual. Subsequently, every offspring and each of its neighbors is evaluated `s` times. The offspring and the corresponding cloud members are then written to the `var` file and state 3 is set.

After reading the `var` file in state 3, the selector scales the objective vectors of the received offspring, assigns the fitness values and performs the

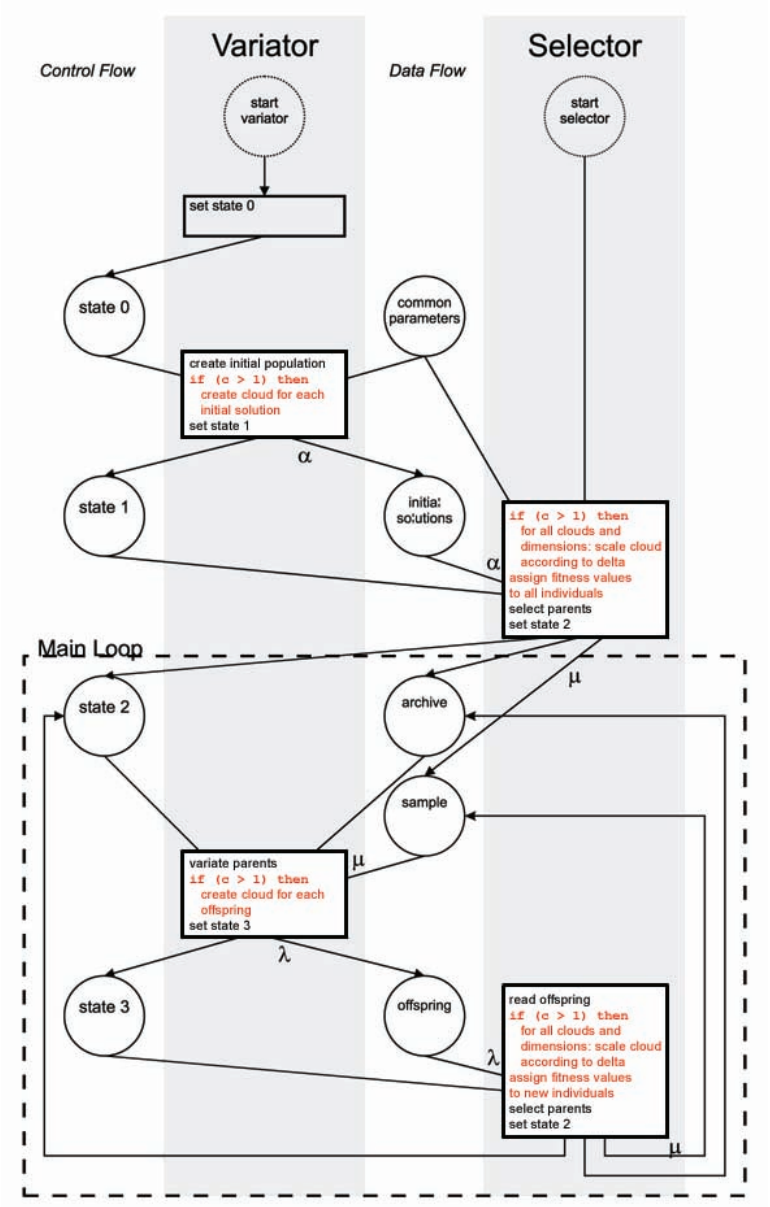

Figure 3.4: Adapted PISA: extended control flow and data flow specification

mating selection, before writing the `arc` and `sel` files and setting the state variable.

Figure 3.5 shows the data files that are exchanged. The additional data, that is needed by procedures which consider robustness and/or uncertainty, is highlighted in red.

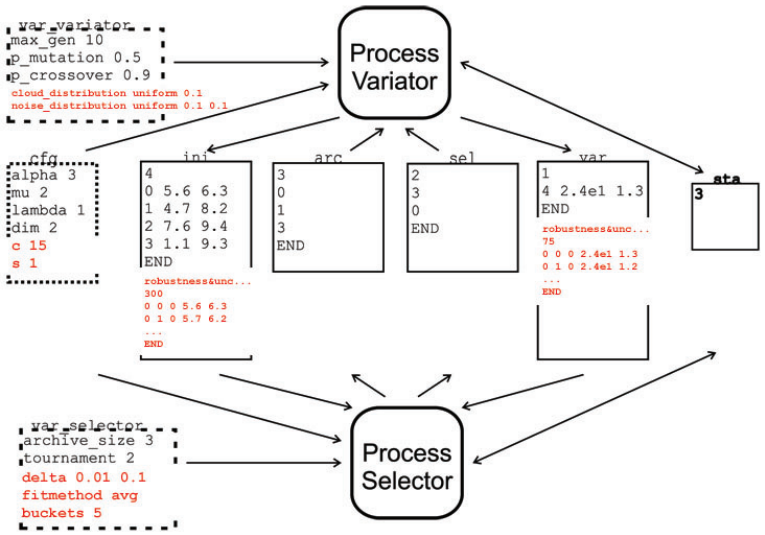

Figure 3.5: Adapted PISA: data files including additional data

The new parameters which are only needed by either the selector or the variator, are added to their respective parameter files. In our implementation, the testfunction needs to know how to distribute the neighbors when creating clouds and in which interval these values have to lie (specified after `cloud_distribution`), for example.

The `cfg` file now also includes the values for the cloud size `c` and sample size `s`, which are needed by both modules.

The files containing the information on the individuals (`ini` and `var`) are extended as follows. The first section remains exactly the same, as it will still be used by all modules that do not support the handling of robustness or uncertainty. After the `END` tag, a new section is added, which starts with the title "robustness&uncertainty". A module that is looking for this data can scan the file for this title tag and ignore the preceeding data. The next entry specifies the number of data elements following, so the reading module knows how much it has to read. Then, the properties of each individual are added in the following order (each on a separate line): the index of the individual, the index of the cloud member (0 stands for the individual itself), the index of the sample and objective function values of the considered sample. The last entry is again an `END` tag, which is needed to detect file corruption. You will find a close-up of an example `var` file (for a cloud size of $c = 2$ and a sample size of $s = 2$) in Figure 3.6.

var

```
6
4 2.3 5.7
6 3.1 2.4
END
robustness&uncertainty
40
4 0 0 2.3 5.7
4 0 1 2.4 5.7
4 1 0 2.3 5.6
4 1 1 2.5 5.8
6 0 0 3.1 2.4
6 0 1 3.1 2.5
6 1 0 3.1 2.7
6 1 1 3.4 5.8
END
```

Figure 3.6: Adapted PISA: example `var` file with additional data

Table 3.1: The implemented algorithm

Algorithm for handling of robustness & uncertainty

Input:	α	*(population size)*
	μ	*(number of selected parents)*
	λ	*(size of offspring)*
	G	*(maximum number of generations)*
	c	*(cloud size; number of solutions in a decision space cloud)*
	s	*(sample size; number of objective vector samples drawn per solution)*
	δ	*(limit on the diameter of a cloud in the objective space)*
Output:	S	*(Pareto set approximation)*

Step 1: **Initialization:** *Generate an initial population S of size α. Generate clouds $\mathbf{C}(\mathbf{x})$ by adding $c-1$ neighboring solutions around each initial individual $\mathbf{x} \in S$ according to a specified probability distribution. Draw a sample of objective function vectors of size s for each individual $\mathbf{x} \in S$ and for every neighboring solution in the surrounding cloud, and add these to the sample cloud $\mathbf{S}(\mathbf{x})$. Set the generation counter g to 0.*

Step 2: **Fitness assignment:** *For every $\mathbf{x} \in S$ and every dimension in the objective space, calculate the diameter of $\mathbf{S}(\mathbf{x})$ (i.e. the difference between the worst and best objective function value). For clouds with a diameter $> \delta$, scale the related objective function values for the corresponding objective function, so that a lower bound on the quality of the solutions in a cloud can be guaranteed.*
Calculate the fitness values of all $\mathbf{x} \in S$ according to the specified method (i.e. avg, exp or bck).
Avg: *calculates the average of the vectors in a sample cloud $\mathbf{S}(\mathbf{x})$ and uses these to determine the fitness of each individual $\mathbf{x} \in S$: $Fit(\mathbf{x}^*) = \sum_{\mathbf{x} \in S \setminus \{\mathbf{x}^*\}} -e^{-I(\{av(\mathbf{x})\},\{av(\mathbf{x}^*)\})/\kappa}$.*
Exp: *directly uses all sample objective vectors to calculate fitness: $Fit(\mathbf{x}^*) = \sum_{\mathbf{z}^* \in \mathbf{S}(\mathbf{x}^*)} \sum_{\mathbf{x} \in S \setminus \{\mathbf{x}^*\}} \sum_{\mathbf{z} \in \mathbf{S}(\mathbf{x})} -e^{-I(\{\mathbf{z}\},\{\mathbf{z}^*\})/\kappa}$.*
Bck: *uses bucket sort and estimates expected loss in quality: $Fit(\mathbf{x}) = \sum_{\mathbf{z} \in \mathbf{S}(\mathbf{x})} \hat{E}(I(F(S \setminus \{\mathbf{x}\}), \{\mathbf{z}\}))/|\mathbf{S}(\mathbf{x})|$ for all $\mathbf{x} \in S$.*

Step 3: **Environmental selection:** *While $|S| > \alpha$, remove the individual $\mathbf{x}^* \in S$ with the worst fitness value (i.e. $Fit(\mathbf{x}^*) \leq Fit(\mathbf{x}) \; \forall \; \mathbf{x} \in S$) from S and update the fitness values of the remaining individuals.*

Step 4: **Termination:** *If $g \geq G$ return S.*

Step 5: **Mating selection:** *Perform tournament selection on S to select μ parents.*

Step 6: **Variation:** *Apply recombination and mutation operators to the selected parents. Insert each offspring $\mathbf{x}$ into the population S, generate a cloud $\mathbf{C}(\mathbf{x})$ surrounding the newly created individual and draw its sample cloud $\mathbf{S}(\mathbf{x})$. Increment the generation counter ($g = g + 1$) and go to Step 2.*

4

Results and Discussion

This chapter provides a description of the testfunction developed and used to test the implemented algorithm. The focus is laid mainly upon visualizing the program's efficiency in finding the more robust fraction of the solutions. In addition, the simulation is presented and the results and their implications are discussed in detail.

4.1 Testfunctions

Since the approach taken to consider uncertainty was not greatly modified compared to its description in [1], the focus was laid on simulating the influence and handling of robustness.

In order to assess the algorithm which was developed and implemented in this study, a testproblem had to be found, which allows the introduction of robustness and uncertainty. To this end, the existing testfunction ZDT1 was extended. ZDT1 is part of a rich collection of testproblems, the DTLZ test suite [7], which is available for download on the Website `http://www.tik.ee.ethz.ch/pisa`.

The ZDT1 testproblem is given as follows.

$$
\begin{array}{ll}
Minimize & f_1(\mathbf{x}) = x_1, \\
Minimize & f_2(\mathbf{x}) = g(\mathbf{x}) * h(\mathbf{x}). \\
where & 0 \leq x_i \leq 1, \quad \text{for } i = 1, 2, ..., n, \\
 & g(\mathbf{x}) = 1 + 9 * \frac{1}{n-1} * \sum_{j=2}^{n} x_j, \\
 & h(\mathbf{x}) = 1 - \sqrt{x_1/g}.
\end{array}
$$

Figure 4.2 shows the objective space of ZDT1 for a grid of solutions, evenly distributed on $[0, 1] \times [0, 1]$ in the decision space. The lowest border in this plot is the Pareto-optimal front, which to find is the ultimate goal.

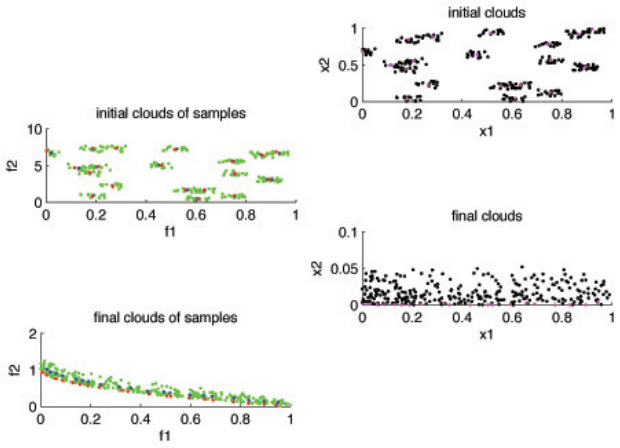

Figure 4.1: ZDT1: plot of pareto-optimal front (`c`: 15, `s`: 1, `delta`: 1000, fitmethod: `avg`)

Since there are no regions, where a variation in the decision space leads to more variation in objective values than elsewhere, there are no robust nor non-robust solutions in this scenario and even if the variable perturbation is large (e.g. $cp = 0.1$), the same Pareto-optimal front is found (see Figure 4.1). In order to introduce this characteristic, the ZDT1 function was adapted, creating the new testproblem, ZDT1ru.

The ZDT1ru testproblem is given as follows.

$$
\begin{aligned}
&\textit{Minimize} \quad f_1(\mathbf{x}) = x_1, \\
&\textit{Minimize} \quad f_2(\mathbf{x}) = \begin{cases} g(\mathbf{x}) * h(\mathbf{x}) + (x_1 - 0.7 + 1)^2 - 1 \\ \qquad\qquad \text{if } s(\mathbf{x}) < 0.05 \text{ and } x_1 > 0.7 \\ g(\mathbf{x}) * h(\mathbf{x}) \qquad \text{else.} \end{cases} \\
&\textit{where} \quad 0 \leq x_i \leq 1, \quad \text{for } i = 1, 2, ..., n, \\
&\qquad g(\mathbf{x}) = 1 + 9 * \tfrac{1}{n-1} * \textstyle\sum_{j=2}^{n} x_j, \\
&\qquad h(\mathbf{x}) = 1 - \sqrt{x_1 / g}.
\end{aligned}
$$

The two testproblems, ZDT1 and ZDT1ru, differ only for values of $f_1 > 0.75$, which can be seen in Figure 4.2. The lowest border is, again, the Pareto-optimal front. For values of $f_1 > 0.75$ the curve is perturbed and its

steepness ascends quickly, introducing a larger variation in objective function values for solutions in this region. What results, is a region of non-robust solutions and it is this feature, which allows us to visualize the influence of robustness when optimizing on this testfunction.

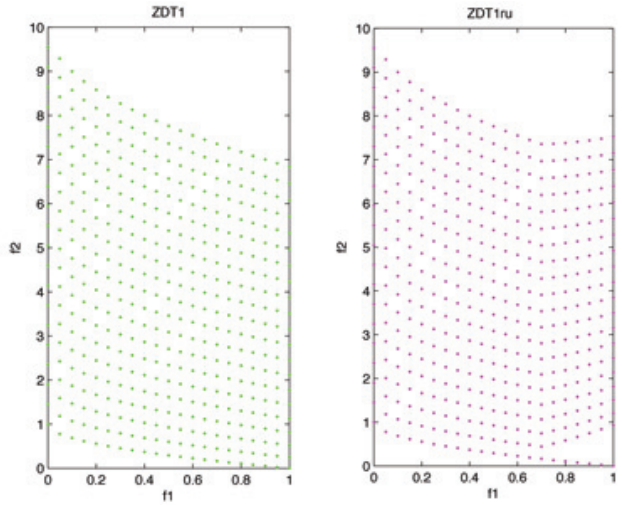

Figure 4.2: Plot of TODO testfunctions ZDT1 and ZDT1ru

4.2 Simulation

For all simulation runs, following values were chosen: the population size was set to 20 and 100 generations were evaluated. The tests were executed for 2 decision variables and 2 objectives, for reasons of visualization.

First the algorithm was tested on ZDT1, for a cloud size of $c = 1$ and sample size of $s = 1$, to assure the correct functioning of the algorithm. This test was successful and the Pareto-optimal front was easily found (see Figure 4.3). Next, the same test was executed, this time for our new testfunction ZDT1ru, and again the Pareto-optimal front was found.

In order to test our algorithm's ability to find robust solutions, the following setup was used. For increasing values of the interval cp, from which the cloud members are sampled in the decision space, a set of 30 runs was executed using the fitness assignment method `avg`. After each set, the largest values of f_1 that were found in each run were plotted, in order to visualize

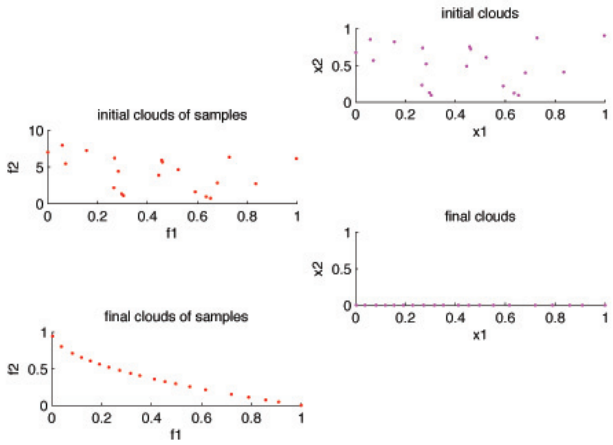

Figure 4.3: ZDT1: plot of pareto-optimal front (`c`: 1, `s`: 1, `delta`: 1000, fitmethod: `avg`)

the increasing influence of the variation in decision space. This was first done without scaling the objective function values ($delta = 1000$).

The result is shown in Figure 4.4. The values of the interval cp are: 0.02, 0.025, 0.03, 0.04, 0.05 and 0.1, from top to bottom. The effect is clear: with increasing values of cp the largest found values of f_1 decrease and for $cp = 0.05$ only solutions on the Pareto-optimal front are found whose f_1 values are less than 0.75.

The explication for this phenomenon is simple. Due to the averaging of the objective vectors, robust solutions are implicitly rated better than non-robust solutions. Therefore, the optimizer only finds the robust solutions, which have an f_1 value of less than 0.75 if the interval cp is chosen large enough.

The next step was to repeat this setup, this time with the additional scaling of objective function values, setting $delta$ to 0.01 for the first objective and 0.1 for the second. The result is shown in Figure 4.5. Again, we see the same scenario, but this time the amount of variation in decision space has a greater effect on the performance of the optimizer. Now, a value of $cp = 0.03$ suffices to find only solutions on the Pareto-optimal front whose f_1 values are less than 0.75.

Further tests were conducted for varying cloud sizes ($c = 1, 5, 10$ etc.) and the results again reflected our expectations: the larger the cloud sizes

were chosen, the smaller a value for *cp* was needed in order to find only solutions to the left of the point where $f_1 = 0.75$. The less neighbors are considered, the less variation will probably become obvious and the higher the the probability will be, that solutions are rated better (or in our case: more robust) than they really are.

Simulation runs using the same setup for different fitness assignment methods (`exp`, `bck`) showed similar results, although the influence of robustness was even greater here. One simulation run using fitmethod `exp` without scaling the objectives (`delta` = 1000), for example, showed much the same performance as the same run using fitmethod `avg` *with* scaling (`delta`: 0.01, 0.1)

Unfortunately, the ZDT1ru testfunction does not help to visualize the effects of uncertainty, so that the combined effects of robustness and uncertainty could not yet be thoroughly tested.

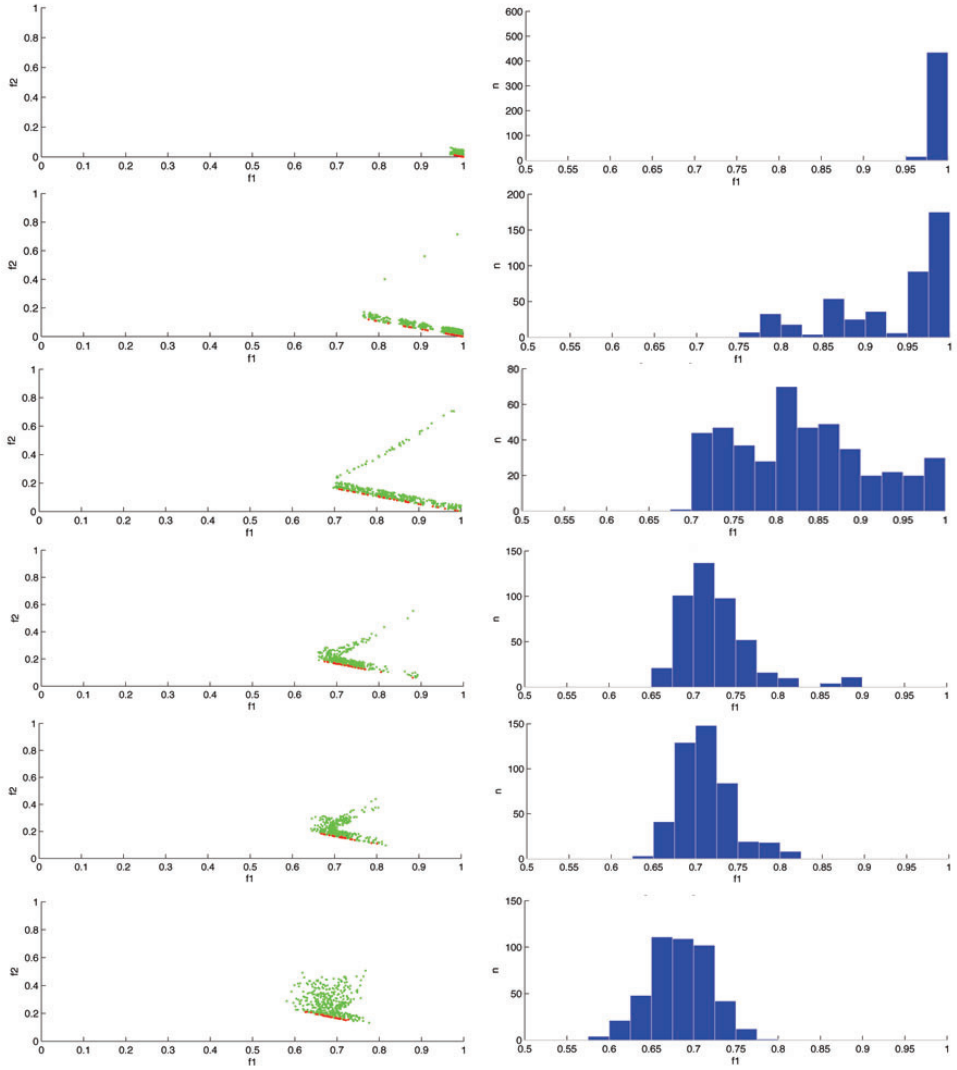

Figure 4.4: ZDT1ru: plots of largest values of f1 found for `delta`: 1000 and different values of `cp` over 30 runs using fitmethod `avg`

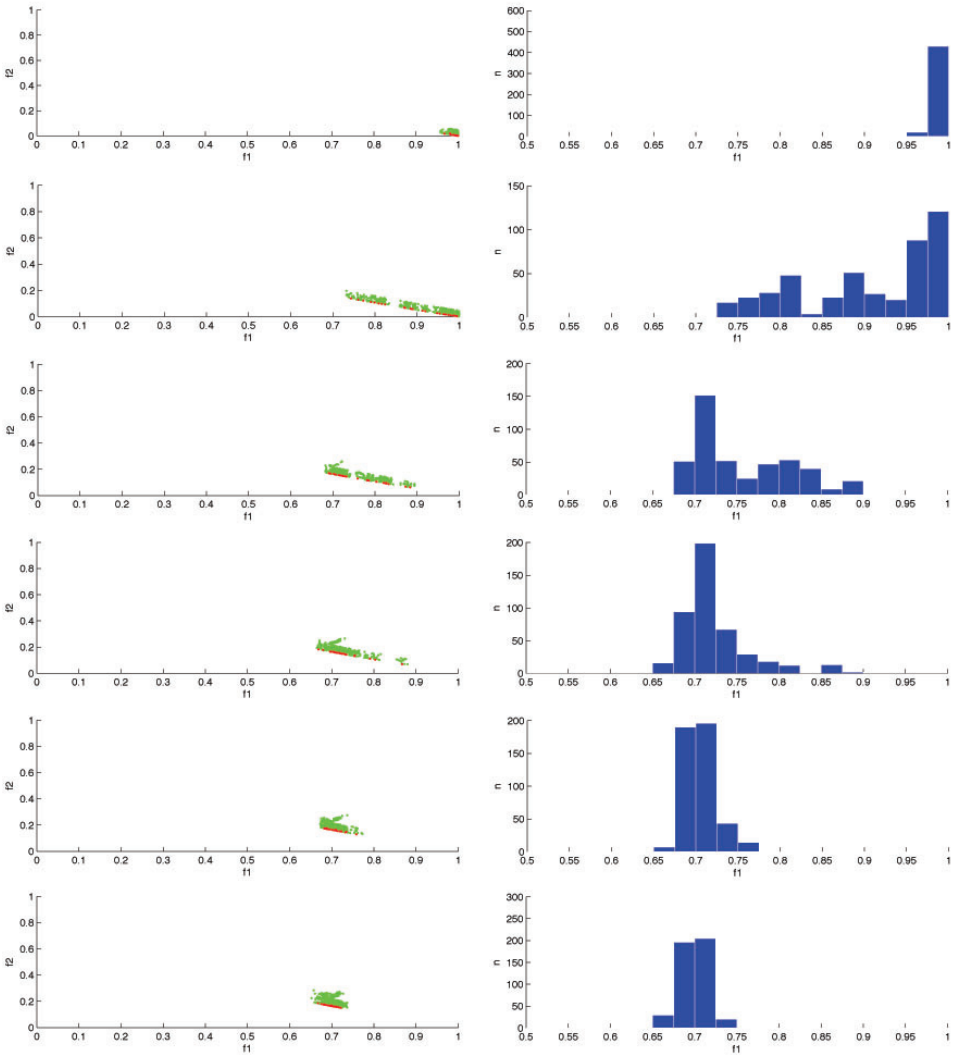

Figure 4.5: ZDT1ru: plots of largest values of f1 found for `delta`: $0.01, 0.1$ and different values of `cp` over 30 runs using fitmethod `avg`

5

Conclusion and Outlook

The following chapter concludes this thesis by giving a short overview of the work and the achieved results in Section 5.1 and presenting some ideas for future work and following projects in Section 5.2.

5.1 Conclusion

In this thesis, the main goal was to develop an approach which provides a way to consider robustness and uncertainty in multi-objective optimization. To this end, the following tasks were dealt with.

1. On the basis of [1] and [2] a concept was developed which allows the handling of robustness and uncertainty in evolutionary multi-objective search.

2. The PISA interface was successfully adapted, in order to provide support for modules which consider robustness and/or uncertainty, thereby preserving downward compatibility.

3. An algorithm which incorporates the developed concept was implemented in the PISA framework, which included creating a selector and a variator on the basis of existing modules.

4. In order to verify the algorithm, a testfunction was generated and integrated in the implemented variator.

5. The designed algorithm was tested and the value of the chosen approach verified, which included the simulation of its performance for different parameter values and the subsequent evaluation of the gained results.

The goals formulated in Section 1.3 were successfully achieved and the ability of the algorithm to handle robustness and uncertainty verified in the simulation process.

One thing to note is, that the simulation phase was kept rather short, and only the most important tests conducted, due to a lack of time. Therefore, it is necessary that future tests are performed, in order to assess the approach and implemented algorithm in more detail.

One drawback of the presented approach is the static operational sequence of the algorithm, which does not yet provide the ability to dynamically adapt the size of clouds and samples during an optimization run. In order to provide such a feature, the PISA protocol would have to undergo major changes.

5.2 Outlook

There are several interesting possibilities of projects and further research, based on the presented work.

First of all, the presented algorithm still needs extensive testing, in order to be verified and assessed properly. Eventually, these tests would want to include the application of the selector on real-world problems, where the significance of the approach would have to be assessed. The performance of the algorithm must also be tested on different testfunctions, including benchmark problems, and a comparison with other optimizers conducted.

Furthermore, a future study could include the adaptation of the presented variator, so as to provide a decision space topology with regions of differing variation distributions. The described selector could then be extended with the ability to dynamically focus on regions with a greater possibility to contain predominantly robust solutions.

Bibliography

[1] Matthieu Basseur, Eckart Zitzler. *Handling Uncertainty in Indicator-Based Multiobjective Optimization.* International Journal of Computational Intelligence Research, Research India Publications, 2(3), pages 255–272, 2006.

[2] Eckart Zitzler, Simon Künzli. *Indicator-Based Selection in Multiobjective Search.* 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII), Springer-Verlag, Berlin, Germany, pages 832–842, 2004.

[3] Stefan Bleuler, Marco Laumanns, Lothar Thiele and Eckart Zitzler. *PISA – A Platform and Programming Language Independent Interface for Search Algorithms.* Evolutionary Multi-Criterion Optimization, Second International Conference, EMO 2005, pages 494–508, 2003.

[4] Kalyanmoy Deb, Himanshu Gupta. *Searching for Robust Pareto-Optimal Solutions in Multi-objective Optimization.* Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005, pages 150–164, 2005.

[5] Yaochu Jin, Jürgen Branke. *Evolutionary Optimization in Uncertain Environments – a survey.* IEEE Transactions on Evolutionary Computation, 9(3), pages 303–317, 2005.

[6] David Fogel. *EvolutionaryComputationTheFossil Record.* IEEE Press, Piscataway, 1998.

[7] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns and Eckart Zitzler. *Scalable Test Problems for Evolutionary Multi-Objective Optimization.*

Evolutionary Computation: Theoretical Advances and Applications, Springer, 2004.