CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Civil Engineering

Ph.D. Programme: Civil Engineering

Branch of study: Physical and Material Engineering

Ing. Zuzana Vitingerová

EVOLUTIONARY ALGORITHMS FOR MULTI-OBJECTIVE PARAMETER ESTIMATION

Evoluční algoritmy pro vícekriteriální identifikaci hodnot parametrů

DOCTORAL THESIS FOR OBTAINING THE DEGREE OF Ph.D.

Supervisor: Ing. Matěj Lepš, Ph.D.

Prague, June 2010

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my supervisor Ing. Matěj Lepš, Ph.D. for his support and patience not only during the whole course of the work on this thesis but also during my whole studies.

I also wish to thank other colleagues who have given me valuable comments and advice, namely Ing. Anička Kučerová, Ph.D. and Doc. Ing. Vít Šmilauer, Ph.D.

Also, I am very grateful to colleagues from the next door office for their friendly support and access to the coffee maker, without them, the last days of thesis writing would be unbearable.

Most importantly, I would like to thank my parents, my brothers and friends for their never ending encouragement and support that help me to attain the goal I have set for myself and the opportunity to fully concentrate on my study.

Last but not least, the financial support provided by the MPO research project FT-TA4/100 and by GACR grant, GACR 103/09/H078 is gratefully acknowledged.

ABSTRACT

The determination of input parameters for any model is crucial for its use. However, the nowadays used models are very complex and their parameters do not have an easily interpreted relation to the modelled material or experiment; therefore, this task can be very difficult. The parameter estimation task is a special type of inverse analysis, i.e. the task of obtaining unknown inputs (values of model input parameters in this case) from known outputs (experimental or model data). This thesis deals with a possible solution of such a problem, particularly with a multi-objective approach based on evolutionary algorithms.

The thesis briefly introduces methods for the parameter estimation. Particularly: a *hand fitting method* and a *trial and error method*, which represent easy but still very often used methods. Next, an *inverse mode* is presented; this method assumes the existence of an inverse relation between outputs and inputs and searches for this relation or its approximation. The last method is a *forward mode* based on the minimization of a function determining the difference between desired and estimated outputs. This method can be used universally, but the optimization process must be repeated for each experiment; therefore, the applied algorithm should be efficient. Traditional gradient based methods cannot usually be used, because the mathematical description is unknown, moreover, the defined error function is very often multi-modal. Therefore, artificial intelligence methods, e.g. evolutionary algorithms, which do not require the knowledge of mathematical formulae or the differentiability of the error function, are used instead. However, these algorithms tend to converge to the local optima. One possibility to tackle this obstacle is determining additional error functions and transforming the originally single-objective optimization to the multi-objective one.

The next part of the thesis presents the basic principles of multi-objective optimization and algorithms used for its solution. NSGA-II, SPEA2 algorithms, the Weighted Sum Method and the Average Ranking method are introduced in details; these algorithms are later used for testing the proposed methodology. The properties of these algorithms are demonstrated on a set of test problems commonly used in multi-objective literature. The performance of individual algorithms is studied regarding the number of iterations as well as the population size. These two indicators are important, because the models which are to be estimated can be very time consuming; therefore, the algorithm capable of finding a good solution with a low number of function evaluations is preferred.

In the last part of the thesis, these algorithms are used for the parameter estimation of three real engineering applications. The first one is a model of cement paste hydration, the next is a model of an infiltration experiment in an environment with preferential flow and the last case is a finite element model of a nanoindentation experiment.

ABSTRAKT

Určení hodnot vstupních parametrů jakéhokoliv modelu je zásadním úkolem pro jeho použití. Ovšem vzhledem k tomu, že v současnosti používané modely jsou velmi komplexní a navíc jejich vstupní parametry často nemají snadno interpretovatelný vztah ke skutečnému materiálu či experimentu (což je dáno obtížným fyzickým popisem experimentu), není tato úloha nikterak jednoduchá. Identifikace hodnot parametrů modelu je úlohou inverzní analýzy, tedy úlohou, kde ze známých výstupů (z experimentu či z numerického modelu) chceme získat neznámé vstupy (v tomto případě hodnoty vstupních parametrů pro model). Disertační práce se zabývá jedním z možných řešení takto definovaného problému, konkrétně vícekriteriálním postupem identifikace s využitím evolučních algoritmů.

V práci jsou stručně představeny možné způsoby určování hodnot vstupních parametrů: určení parametrů *„od oka“* a metoda *pokus - omyl*, které zastupují jednoduché, ale stále velmi často používané metody. Dále pak *zpětný postup*, který předpokládá existenci inverzního vztahu mezi výstupy a vstupy a snaží se takovýto vztah najít, resp. aproximovat. Posledním způsobem je *přímý postup*, který je založen na minimalizaci cílové funkce určující rozdíl mezi výstupem z modelu a experimentem (případně známou simulací modelu). Hlavní výhodou tohoto postupu je jeho univerzálnost. Naopak nevýhodou je nutnost provádět optimalizaci znova pro každý nový experiment, což je samozřejmě výpočetně velmi náročné a je proto důležité nalézt velmi efektivní optimalizační algoritmus. Tradiční gradientní metody v tomto případně většinou nelze aplikovat, protože matematický předpis cílové funkce je neznámý, a funkce jsou velmi často multimodální. Proto se pro takto definovanou úlohu často využívají metody umělé inteligence, např. evoluční algoritmy, které nevyžadují znalost matematického předpisu, diferencovatelnost ani spojitost funkcí. Nicméně evoluční algoritmy mají tendenci konvergovat do lokálního optima. Jedním z možných řešení k odstranění tohoto problému je definování doplňkových chybových funkcí a převedení jednokriteriální optimalizace na vícekriteriální.

Další část disertační práce představuje základní principy vícekriteriální optimalizace a některé algoritmy používané pro její řešení. Podrobně jsou představeny algoritmy NSGA-II, SPEA2, Weighted Sum Method a metoda Average Ranking, které jsou později použity pro testování navrhovaného přístupu k identifikaci parametrů. Vlastnosti těchto algoritmů jsou prezentovány na sadě testovacích funkcí používaných v literatuře zabývající se vícekriteriální optimalizací. Úspěšnost algoritmů je sledována jak v závislosti na velikosti populace tak v závislosti na počtu proběhlých iterací. Tyto dva ukazatele jsou sledovány proto, že modely, jejichž parametry je třeba identifikovat, bývají časově velmi náročné a je proto nutné nalézt algoritmus, který k dosažení dostatečně přesného výsledku nepotřebuje mnoho ohodnocení cílových funkcí.

Na závěr jsou pak tyto algoritmy použity k odhadu parametrů tří skutečných materiálových modelů. Prvním z nich je model hydratace cementové pasty, dále pak model infiltračního experimentu v prostředí s preferenčním prouděním a posledním případem je konečně-prvkový model nanoindentace cementu.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Motivation and objectives

Model parameter estimation is a frequent engineering task that occurs every time an input of a model is required and this input cannot be measured or does not have a clear relation with the modelled situation. Many scientific areas deal with parameter estimation, besides the civil engineering field it can be found in electronic engineering, astrophysics or control systems tasks. Also, different names can be found in literature for the same task, e.g. parameter identification, model calibration, model fitting, etc.

Parameter estimation (**PE**) can be understood as a special type of an inverse analysis problem. Generally, the aim of inverse analysis is to rediscover unknown inputs from known outputs, particularly the aim of PE is to find a set of parameters for a numerical model describing the experiment. A PE problem can be solved by many distinct methods. This thesis concerns one of these methods, a not so common approach based on multi-objective optimization. Although multi-objective optimization is very popular nowadays, it is not frequently used for parameter estimation. However, based on some publications and results, which are presented later, one can assume that the multi-objective approach can be worthwhile for parameter estimation as well.

Despite the existence of some papers dealing with multi-objective parameter estimation and some papers about re-formulating the single-objective optimization problem (**SOP**) as a multi-objective one, to the best author's knowledge, there is no publication about „multi-objectivized“ parameter estimation. In other words, no one have analysed the specifics of the following phenomena:

- the selection of proper objectives;
- the probable existence of a so-called ideal point[1] and therefore, the situation of non-conflicting objectives[2] (sometimes called correlated objectives);
- the fact that we are not interested in the whole Pareto front, whereas the commonly used multi-objective optimization algorithms are designed to approximate the front as wide as possible;
- the selection of a final solution; it seems to be appropriate to choose the one with the best value of original objective function, but is it really true?

Therefore, the most important output of this thesis is a general recommendation when and how to use the multi-objective approach for parameter estimation. Particularly, the detailed objectives of the thesis are:

1. test the proposed estimation methods in the framework of several constitutive models;
2. provide a guide for the best choice of:
    (a) the type of algorithm most suitable for a particular application;
    (b) additional objectives;
    (c) final solutions.

The thesis is organized as follows: This chapter presents an introduction to the parameter estimation theory. The remaining text is divided into two parts. The first one is more theoretical and contains the description of algorithms and their performance in standard test problems. Chapter 2, describing in detail an GRADE algorithm, presents the standard approach to which the proposed methodology will be compared. Chapter 3 deals with a multi-objective approach to parameter estimation. At first, the idea of *multi-objectivization* and

[1] the point where all objectives have the minimal values
[2] at least not in the entire feasible space

objective helpers, i.e. the idea of adding more objective functions to a single-optimization problem, is presented. Then, basic concepts of the multi-objective optimization and a brief review of existing algorithms, with an emphasis on multi-objective evolutionary algorithms (**MOEA**s), are presented. The algorithms tested in this thesis, i.e. Weighted Sum Method (**WSM**), Average Ranking (**AR**), Nondominated Sorting Genetic Algorithm II (**NSGA-II**) and Strength Pareto Evolutionary Algorithm 2 (**SPEA2**) are introduced. Chapter 4 provides a brief introduction to performance assessment for multi-objective problems, and results of chosen algorithms in standard multi-objective test problems are presented to show the advantages and disadvantages of individual methods.

The second part deals with applications of presented methodology in real engineering tasks. At the beginning, some existing applications of multi-objective parameter estimation are presented. Because of the absence of test problems for this type of tasks, the next chapters serve as benchmarks for the proposed methodology. Chapter 6 discusses parameter estimation for the affinity model of cement paste hydration. Chapter 7 shows results for PE for the numerical solver of a dual porosity model of Richard's equation. In the next chapter, parameters for the finite element model of cement paste nanoindentation are estimated. Chapter 9 discusses all results and gives general recommendations for multi-objective parameter estimation.

1.1 Parameter estimation theory

For the introductory part, we follow the description presented in [Mahnken, 2004] and notation used in [Lepš, 2008], later extended in [Kučerová, 2007]. Here, the inverse analysis is based on the existence of an experiment E, which, physically or virtually, connects the known inputs (parameters) $\mathbf{x}^E$ to the desired outputs (measurements) $\mathbf{y}^E$. Formally, this can be written as

$$\mathbf{y}^E = E(\mathbf{x}^E). \tag{1.1}$$

The problem of inverse analysis is defined as a search for unknown inputs $\mathbf{x}^E$ from the known outputs $\mathbf{y}^E$, i.e. inversely to the experiment E. In engineering applications, the experiment E can be simulated by some virtual model M. Usually, the model is a program based on numerical methods, for example the finite element method. Such a model M usually does not describe a real experiment E exactly, but it is considered as a „good“ approximation:

$$M \approx E. \tag{1.2}$$

The use of the model instead of the real experiment is important in terms of economy, where the cost of the evaluation of the model M is assumed to be by an order of magnitude smaller than the cost of the physical experiment E.

Theoretical models are constructed to describe real experiments in order to obtain equivalent outputs (measurements). Therefore, the *output parameters* $\mathbf{y}^M$ of a theoretical model should correspond to those from the experiment $\mathbf{y}^E$. On the other hand, this type of models often uses parameters without physical interpretation. It means, that in general, *input parameters* $\mathbf{x}^M$ of a theoretical model can be different from physical parameters $\mathbf{x}^E$. It is caused by difficulty in the determination of physical parameters or description of physical phenomena. Therefore, in the case of a perfect fit, we can write

$$\mathbf{y}^E = \mathbf{y}^M = M(\mathbf{x}^M). \tag{1.3}$$

Hence, the goal of a parameter estimation problem is to find the input model parameters using the response of the model. This process is necessary in two phases of the model lifecycle. At first, when a new model is created, its validation has to be done to prove the model is able to describe the physical experiment with sufficient accuracy. Second, another use of the estimation method is on demand when new values of model parameters should be found to fit experimental measurements on a new material. Because the estimation process is supposed to be used repeatedly for any new measurement, not only the accuracy but also the efficiency of the estimation method is essential for its choice.

In general, four main possible solutions of the estimation problem can be described: a *hand fitting*, a *trial and error*, an *inverse* and a *forward* method.

1.2 Methods

1.2.1 Hand Fitting Method

In the case, that the input parameters are directly related to experimental results or the user is experienced enough the fitting procedure can be performed „by hand“ this method is also called the *guru method*. In complex cases *hand fitting* can be divided into more steps: the effects from single parameters are separated, certain parameters are estimated from one test or a part of an experimental curve and then, the already obtained parameter is used along with other data for the estimation of next parameters.

The advantage of this method is clear, it is fast and easy. However, it can be difficult to distinguish the effect of input parameters for modern complex models (as mentioned above, input parameters are very often without any physical relationship to the real experiment). Moreover, even when the relationship is direct, experimental measurements are always affected by some noise; therefore, the accuracy of the obtained results is usually very low.

1.2.2 Trial and Error Method

Trial and error is another simple estimation method. The only requirement is an algorithm that solves Equation (1.3) for any $\mathbf{x}$. The iteration steps of the method are as follows:

```
1.  estimate starting values of $\mathbf{x}$;
2.  calculate $\mathbf{y}$;
3.  compare $\mathbf{y}$ with $\mathbf{y^E}$;
4.  if the result is satisfactory, finish;
5.  else estimate new $\mathbf{x}$, go to 2.
```

This method can be used for any inverse problem and no development of a special estimation procedure is needed. On the other hand, its use is very computationally demanding and there is no objective criterion which specifies the accuracy, only the subjective feeling of „satisfaction“.

1.2.3 Inverse Mode

The *inverse* mode assumes the existence of an inverse relationship between outputs and inputs, i.e. the existence of an inverse model M^{INV} connecting outputs $\mathbf{y}$ from the model M with its inputs $\mathbf{x}$:

$$\mathbf{x} = M^{INV}(\mathbf{y}) \tag{1.4}$$

for all possible $\mathbf{y}$. If such a relationship exists and is established, then the desired inputs $\mathbf{x}^M$ are obtained easily by simply inserting $\mathbf{y}^E$ into Equation (1.4). In engineering applications, it is not essential to find the exact description of this relationship, but an approximation is sufficient. The quality of this approximation is easy to measure since a pair $\mathbf{x}$, $\mathbf{y}$ obtained using Equation (1.4) should also fulfil Equation (1.3).

The main advantage is obvious, if an inverse relationship is found, then the retrieval of the desired inputs is very fast even if executed repeatedly. This can be utilized for frequent identifications of one model.

On the other hand, the main disadvantage is an exhausting (and often unsuccessful) search for the inverse relationship. This is probably the main reason why this approach is not so popular as the others. A further obstacle is the inability to solve the problem of the same value of outputs $\mathbf{y}$ for different inputs $\mathbf{x}$, i.e. the existence of several global optima. The opposite, i.e. the existence of different outputs $\mathbf{y}$ related to one input $\mathbf{x}$ introduced by stochastic and probability calculations or by experiments polluted with a noise or an experimental error, can be tackled e.g. by the introduction of stochastic parameters for outputs [Fairbairn et al., 2000; Lehký and Novák, 2005]. Another case, when there is more than one experiment for one material, can be handled by sequential, cascade or iterative processes.

As a solution, different approximation tools are applied. Nowadays, artificial neural networks have become the most frequently used methods, see e.g. [Kučerová, 2007] for more references and applications.

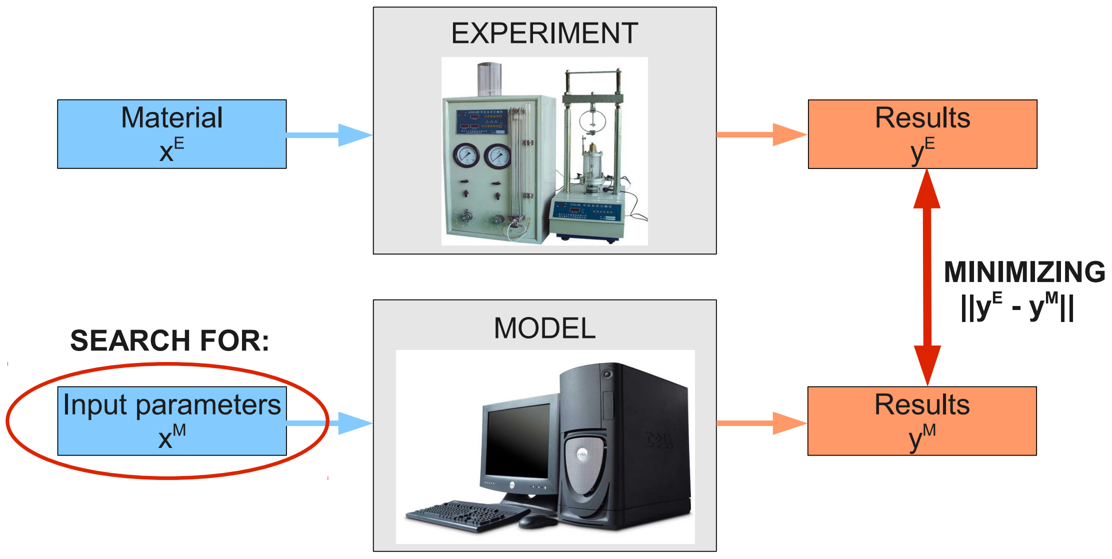

Figure 1.1: Illustration of forward mode of parameter estimation.

1.2.4 Forward Mode

With the above-mentioned statements, the *forward* mode is based on the definition of an error function $F(\mathbf{x})$ of the difference between outputs of the model $\mathbf{y}^M$ and experimental measurements $\mathbf{y}^E$, see Figure 1.1. To find a solution this error function is to be minimized, i.e.

$$\text{minimize } F(\mathbf{x}) = \|\mathbf{y}^E - \mathbf{y}^M)\| = \|\mathbf{y}^E - M(\mathbf{x})\|. \tag{1.5}$$

The forward mode can be understood as a sophisticated version of the trial and error method, where the subjective “satisfaction” is replaced by minimizing the error function and the new $\mathbf{x}$ is created in some defined way. The problem (1.5) has been classically solved by *gradient-based optimization methods*. Nowadays, the model M is often created in a program where the code visibility is limited by license conditions, etc. and, therefore, the knowledge of derivatives is missing even if the function is differentiable. Hence, soft-computing methods can successfully be applied here. *Evolutionary algorithms (**EAs**)* [Goldberg, 1989] or [Michalewicz, 1999] with a “population” of solutions or *the simulated annealing method* [Ingber, 1993; Vidal, 1993] with one solution in time are popular.

The forward mode is general in all possible aspects and (with enough time and a good algorithm) is able to find an appropriate solution if such exists. The method is successful even in special cases like:

a) There are different outputs ($\mathbf{y}$) for one input ($\mathbf{x}$). This is the already mentioned case of stochastic and probabilistic calculations as well as experiments polluted with a noise. This obstacle can be solved by the introduction of stochastic parameters for outputs or by the regularization of the objective function, see [Mahnken and Stein, 1996; Maier et al., 2006].

b) A problem of the same value of outputs ($\mathbf{y}$) for different inputs ($\mathbf{x}$), i.e. the existence of several global optima. This case leads to a multi-modal optimization [Mahfoud, 1995] but it can be solved by an appropriate modification of the optimization algorithm or by a modification of the error function. Both approaches are discussed in more detail in the next chapters. One example of an algorithm created to overcome the multi-modality is presented in Chapter 2. The multi-objective approach proposed in this thesis can be understood as an example of the error function modification, see Chapter 3.

c) There is more than one experiment for one material. This can be solved by a multi-objective formulation of a problem, see e.g. [Lepš, 2007].

One disadvantage of the forward mode, following the definition, is the fact that the computationally expensive search should be repeated for any change in data, e.g. even for a small change in the experimental setup. This feature handicaps the forward mode from an automatic and frequent usage. Moreover, the forward mode usually requires a huge number of error function evaluations. This problem can be managed by two approaches which are based on: (i) parallel decomposition and parallel implementation or (ii) computationally inexpensive approximation or interpolation method, see again [Kučerová, 2007] for an extensive review.

Part I

Algorithms

Chapter 2

GRADE + CERAF ALGORITHM

2.1 Introduction to Evolutionary algorithms

Before the description of individual algorithms is presented, evolutionary algorithms (**EA**s) are briefly introduced.

EAs do not usually use gradient information in their search process, therefore they can be applied in many optimization problems, although, in some optimization problems such as linear or quadratic programming, they will be inevitably beaten by traditional gradient-based optimization procedures. In contrary to most classical optimization methods, EAs use stochastic operators instead of deterministic ones.

EAs belong to the group of bio-inspired algorithms, in this particular case the algorithms are inspired by the Darwinian survival-of-the-fittest evolutionary theory to iteratively create new and better solutions. The algorithm uses more than one solution in an iteration (so called *population*), unlike most classical optimization algorithms that update one solution in each iteration. This feature provides a number of advantages such as:

- a parallel searching power for exploring a bigger area of a search space,
- a possibility of finding multiple optimal solutions, therefore EAs are known to be able to deal with multi-modal or multi-objective optimization problems.

The initial population of EA's search is created randomly in limits specified for each variable, but in the case of any knowledge about the problem optima, it is wise to utilize this information in the initial population. Then, the iterative search process starts: a new (and hopefully better) population (called *offspring* population) is created from the current population (*parent* population) by the use of evolutionary operators, until one or more prespecified termination criteria are met. As the termination criterion, a predetermined number of generations is mostly used.

The main evolutionary operators are: mating selection, crossover, mutation and environmental selection. Their particular implementation can differ in individual EAs, but their main purpose is described further [Deb, 2008].

Mating selection

The mating selection operator ensures the improvement of the solutions quality by choosing better individuals with a larger probability to be involved in the creation of a new population. For this purpose, several stochastic selection operators exist in the EA literature [Baker, 1987]. The simplest form is a *tournament selection*: two solutions are picked at random from the evaluated population and the better one is copied in an intermediate population (so called *mating pool*).

Crossover

Crossover, along with mutation, belongs to variation operators, which are used to create an offspring population. The purpose is to pick two (or more) solutions (parents) from the mating pool and create one (or more) solutions (offsprings) by exchanging their information. The crossover operator is applied with the user's defined probability, called crossover probability $p_c \in [0, 1]$. The probability determines the proportion of population members participating in the crossover operation. The remaining members are simply copied to the offspring population.

Mutation

Each solution, created by the crossover operator, can be then changed by a mutation operator with a mutation probability p_m. The purpose of mutation is to make small changes in the population to search locally around already found solutions. Therefore, p_m is usually set to be much smaller than p_c and the operator creates new solutions very close to the original ones. In real-coded EAs a simple Gaussian probability distribution can be used with its mean at the original variable value.

Environmental selection

After the offspring population is created, it must be evaluated, i.e. objective function(s) value(s) for each population member is/are computed. Then, the offspring population is merged with the parent population. Finally, the size of the merged population is reduced to the original size n. Moreover, in this step, so called *elitism* should be applied. Elitism ensures that the already found best solutions are not lost from the population, and therefore, the algorithm does not have degrading performance. Therefore, the simplest way to provide this step is to keep n best individuals.

Nowadays, there are many types of evolutionary algorithms. Although original genetic algorithms (**GA**s) proposed in [Holland, 1975] were based on binary coding, many engineering problems deal with real-valued representations, and therefore, real-coded algorithms were proposed and will be used in this thesis.

To conclude this section some terms used in the thesis should be stated:

- Gene - a particular variable value. EA's individuals are composed of genes.
- Fitness - a function derived from the values of objective function(s) and constraints. The value is used in selection steps to determine a better solution.
- Individual = Solution = Chromosome - all these terms are used mostly interchangeably. It is a population member, a set of particular variable values (genes), with or without assigned objectives and fitness values (depends on the algorithm's stage).
- Generation - one iteration of an EA.

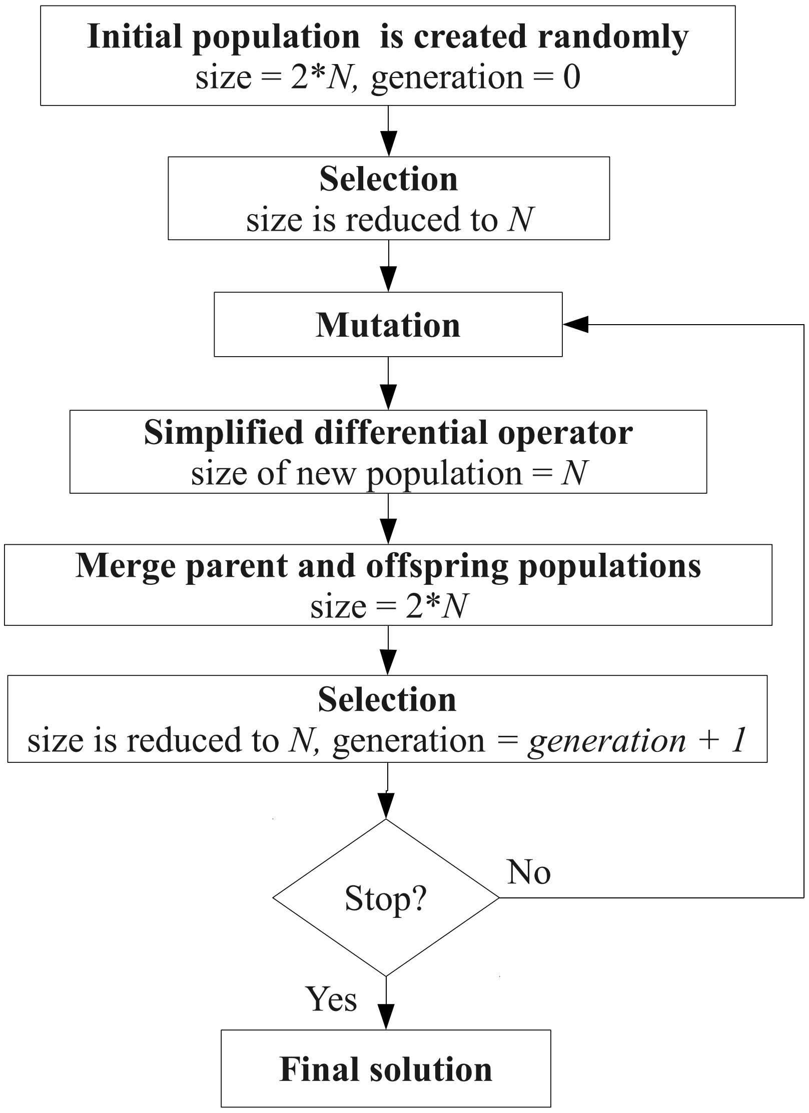
Initial population is created randomly
size = 2*N, generation = 0
Selection
size is reduced to N
Mutation
Simplified differential operator
size of new population = N
Merge parent and offspring populations
size = 2*N
Selection
size is reduced to N, generation = generation + 1
Stop?
No
Yes
Final solution

Figure 2.1: GRADE's flowchart.

2.2 GRADE

The GRADE genetic algorithm was introduced in [Hrstka and Kučerová, 2004]. The authors proposed the method for the optimization of high dimensional (up to 200 variables) real-valued problems. The algorithm combines the properties of differential evolution (the differential operator instead of the crossover operator) [Storn and Price, 1995] with the standard genetic algorithm, see the flowchart of GRADE in Figure 2. This algorithm differs from the general EA presented above in the selection steps. The difference and the implementation of variation operators are described further.

Selection

The authors use the modified tournament strategy: two chromosomes are randomly chosen, compared and the worse (regarding its fitness value) is expelled from the population. Therefore, the population size is decreased by one. This step is repeated until the population reaches its original size. Contrary to the traditional tournament strategy, this approach includes elitism implicitly, i.e. the best solution cannot be lost even if it is not chosen to any tournament.

Mutation

In the mutation, all individuals in the parent population can be used to create an offspring with a probability - $radioactivity$. Let $\mathbf{x}_i(g)$ be the i-th chromosome in a generation g,

$$\mathbf{x}_i(g) = (x_{i1}(g), x_{i2}(g), \dots, x_{iDim}(g)), \tag{2.1}$$

where Dim is the number of variables of the objective function. If the individual $\mathbf{x}_i(g)$ is chosen, then the offspring $\mathbf{x}_k(g+1)$ is computed as:

$$\mathbf{x}_k(g+1) = \mathbf{x}_i(g) + MR(\mathbf{x}_{RP} - \mathbf{x}_i(g)), \tag{2.2}$$

where $\mathbf{x}_{RP}$ is a random individual from the feasible space and MR is a parameter called *mutation rate*, which determines the size of mutation and is chosen randomly from the interval [0,1] for each chromosome.

Simplified differential operator

After M new individuals are created by mutation, this operator is applied to create $(N - M)$ individuals. Here, two parents ($\mathbf{x}_q(g)$ and $\mathbf{x}_r(g)$) are randomly chosen from the current population and one offspring $\mathbf{x}_k(g+1)$ is created:

$$\mathbf{x}_k(g+1) = \mathbf{x}_{better}(g) + CR(\mathbf{x}_q(g) - \mathbf{x}_r(g)), \tag{2.3}$$

where CR is a parameter reducing the vector of the parents' difference and is randomly chosen from the interval [0,CL], where CL is a configurable parameter. The reduced difference is then added to the better one of the parent chromosome $x_{better}(g)$. The better chromosome is the individual, which gives a better value of the fitness function (i.e. lower for minimization problems). Figure 2.2 presents possible geometrical meanings of this operator.

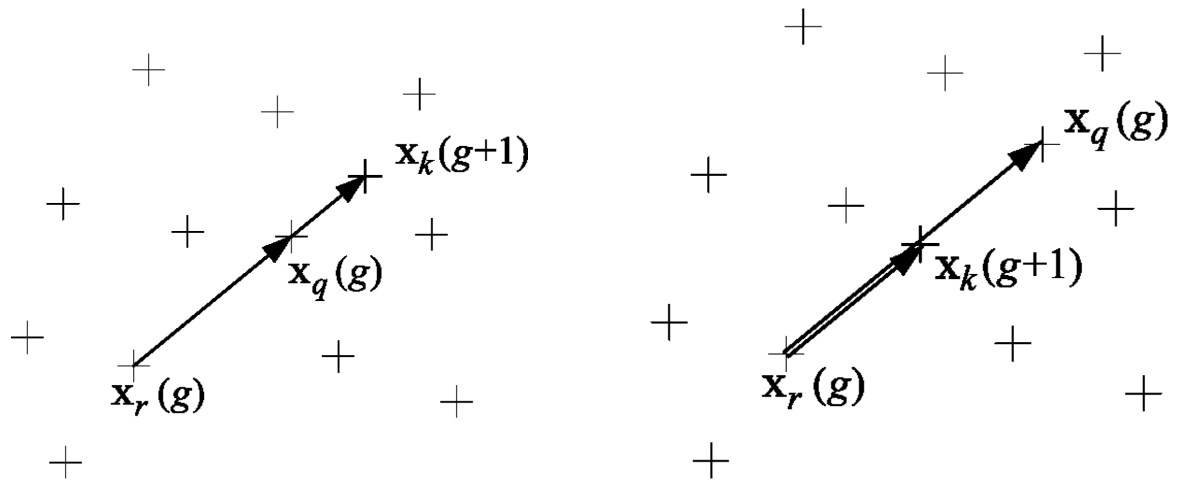

Figure 2.2: Geometrical meaning of simplified differential operator in GRADE algorithm. Left: chromosome $x_q(g)$ is better than $x_r(g)$, right: the opposite situation. Figure reproduced from [Kučerová, 2007].

The GRADE algorithm has only three configurable parameters: Except $radioactivity$ and the CL parameter described above, there is a $pool\_rate$ parameter determining the size of population.

2.2.1 CERAF

The GRADE algorithm itself tends to create a cluster of individuals and move it through the available domain. If such a cluster is trapped in a local extreme, the only chance to escape is the individual with a better fitness value created by mutation outside this cluster. Unfortunately, the probability of this effect is very low. This behaviour is very common within GA, therefore there are many ways to deal with this. It is possible to “restart” the algorithm whenever it is trapped, or a solution created in the neighbourhood of already found local optima can be “punished” with some penalization [Mahfoud, 1995].

The authors of GRADE proposed an improvement called CERAF (from French *CEn-*

Parameter	Value
pop_rate	10
CL	1.0
$radioactivity$	0.2
ceraf_radioactivity	1.0
RAD	0.25
$deact\_rate$	0.995
$quiet$	100

Table 2.1: Parameter setting for GRADE+CERAF algorithm [Kučerová, 2007]

tre RAdioactiF) based on the niching strategy [Hrstka and Kučerová, 2004]. The CERAF method creates areas with a higher level of "radioactivity" in the neighbourhood of all previously found local extremes. The mutation probability is increased many times in these areas (the probability is determined by a tunable parameter *ceraf_radioactivity*), i.e. the chance to create new solutions outside this area and escape from a local extreme is much higher than in GRADE itself. The radioactivity area is an n-dimensional ellipsoid, whose size is defined as a percentage of the domain - RAD. The next parameter of the method - $quiet$ - determines the number of generations, during which the best fitness value has not changed, necessary to mark a new radioactive zone. The radioactivity area is reduced by a small value each time some individual is created there (the radius is multiplied by the parameter called $deact\_rate$). However, the radioactive area never disappears completely, so the chromosomes can never find the marked local extreme again. The steps of the CERAF method are performed between the differential operator and the selection in the original GRADE algorithmic scheme:

1. If any radioactive zone already exists, each chromosome caught there is mutated with high probability a *ceraf_radioactivity*.

2. If the best fitness value stagnates more than $quiet$ generations, it declares a new radioactivity area.

3. Depending on the number of chromosomes created by the cross-over operator and

simultaneously determined in the previous step, the ranges of radioactive zones are decreased.

The results presented by the authors show that the CERAF method can be considered as a universal technique capable of solving any multi-modal optimization problem if the method that is running underneath (i.e. the algorithm that generates new chromosomes) has a sufficient ability to find new possible solutions. Also, based on these tests, the recommended values for parameters for both GRADE and CERAF are presented in Table 2.1. These values are used in all computations in this thesis. The interested reader can obtain the source code for GRADE and CERAF at `http://klobouk.fsv.cvut.cz/~anicka` (C++ and MATLAB® version). Note that whenever GRADE is mentioned in the following text, in fact, GRADE+CERAF is meant.

Chapter 3

MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

3.1 Basic Concepts of Multi-Objective Optimization

Problems where more than one goal should be satisfied simultaneously are as old as mankind itself. The decision between going out of the cave for food or staying safe inside, or decide whether to hunt animals which is more nutritious but also more dangerous or just pick roots and berries. From nowadays problems, the choice of a new car or a computer can be mentioned. There are more conflicting objectives in all these cases, i.e. to appease hunger on the one hand and stay safe on the other hand in caveman's case and to buy the best thing and to save money in the modern man's case. Of course, to solve these problems we do not need any special algorithm, but they illustrate well enough the basic obstacle of multi-objective (also called multi-criteria) problems: there is no solution which satisfies all our demands, all possible solutions are somehow compromised.

Multi-objective problems have been studied since the 19th century. The problems can be divided into two groups. In the first case, there is a list of single solutions and the task is to choose the "best" ones; such problems are called *multi-criteria decision making*. All problems mentioned above belong to this group.

In this thesis, the second group of multi-objective problems are solved, so called *multi-objective optimization problems* (**MOP**s), where the objectives are given by functions and the set of solutions is not listed in advance. As an example, the design of a cantilever beam can be mentioned: the minimum end deflection and the minimum weight of the beam is required. The formal development of mathematical programming techniques capable of dealing with MOPs dates back to the late 1950s. Assuming, without a loss of generality, the minimization

For any two decision vectors $\mathbf{a}$ and $\mathbf{b}$,

$\mathbf{a} \prec\prec \mathbf{b}$	($\mathbf{a}$ strictly dominates $\mathbf{b}$)	*iff*	$\forall i : f_i(\mathbf{a}) < f_i(\mathbf{b})$,
$\mathbf{a} \prec \mathbf{b}$	($\mathbf{a}$ dominates $\mathbf{b}$)	*iff*	$\forall i : f_i(\mathbf{a}) \leq f_i(\mathbf{b}) \wedge \exists i : f_i(\mathbf{a}) < f_i(\mathbf{b})$,
$\mathbf{a} \preceq \mathbf{b}$	($\mathbf{a}$ weakly dominates $\mathbf{b}$)	*iff*	$\forall i : f_i(\mathbf{a}) \leq f_i(\mathbf{b})$,
$\mathbf{a} \parallel \mathbf{b}$	($\mathbf{a}$ is incomparable with $\mathbf{b}$)	*iff*	$\exists i : f_i(\mathbf{a}) < f_i(\mathbf{b}) \wedge \exists j : f_j(\mathbf{a}) > f_j(\mathbf{b})$
$\mathbf{a} \sim \mathbf{b}$	($\mathbf{a}$ is indifferent to $\mathbf{b}$)	*iff*	$\forall i : f_i(\mathbf{a}) = f_i(\mathbf{b})$.

Table 3.1: Pareto optimality for minimization problem [Knowles et al., 2006]

problem, the (unconstrained) MOP can be defined as:

$$\begin{aligned} \text{minimize} \quad \mathbf{f}(\mathbf{x}) \quad &= [(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x}))] \\ \text{subject to} \quad \mathbf{x} \in \mathbf{X} & \end{aligned} \tag{3.1}$$

where $\mathbf{x}$ is the decision vector (i.e. the set of input parameters in our case), $\mathbf{X}$ is a finite set of feasible solutions and the objective function vector $\mathbf{f}(\mathbf{x})$ maps $\mathbf{X}$ into R^k, where $k \geq 2$ is the number of objectives. Note that the minimization problem is assumed in all further presented figures, unless stated otherwise.

The main difference between single and multi-objective optimization is in the concept of optimality. Whereas in the single-objective problem the solution comes with the minimum of one objective function, the objectives in a multi-objective problem are conflicting; therefore, a single point with minimal values for all objective function does not usually exist. The notion of optimality used in MOPs was proposed by Francis Ysisdro Edgeworth in 1881 and was generalized later by Vilfredo Pareto [Pareto, 1896]. The principle is called Pareto optimality after its second author (see Table 3.1).

According to the Pareto optimality definition, instead of a single solution, we seek for a set of solutions $\mathbf{X}_p \subseteq \mathbf{X}$ which are incomparable with each other and no solution from $\mathbf{X}$

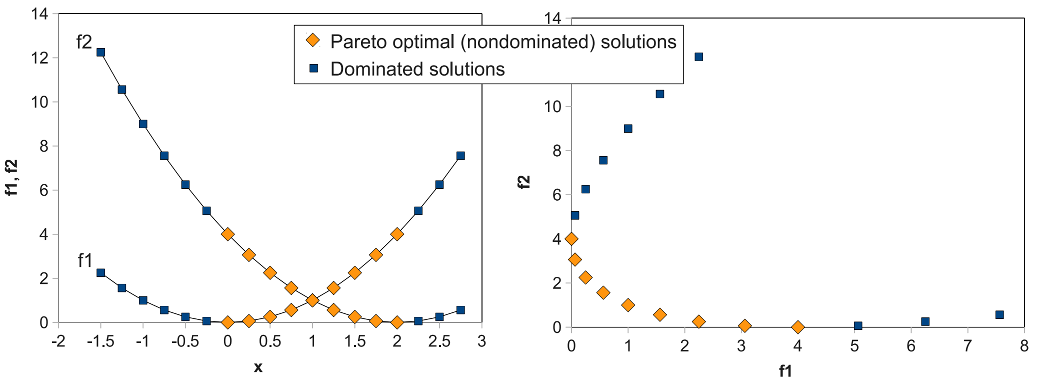

Figure 3.1: Two objectives problem ($min\ f_1 = x^2, f_2 = (x-2)^2$) with only one variable.

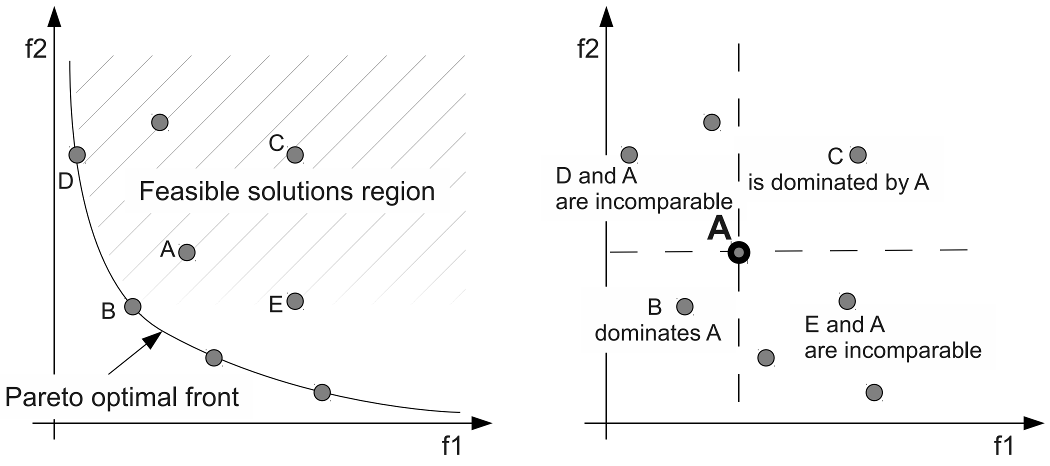

Figure 3.2: Principle of Pareto Optimality (general minimization problem).

dominates them. Such a set is called the *Pareto optimal set*:

$$\forall \mathbf{x}_p \in \mathbf{X}_p : \nexists \mathbf{x} \in \mathbf{X} : \mathbf{x} \prec \mathbf{x}_p \ . \tag{3.2}$$

In other words, solutions from $\mathbf{X}_p$ cannot be improved in any of k objectives unless other objective(s) deteriorate. Also note that the image of $\mathbf{X}_p$ in the objective space $\mathbf{Y}_p = \mathbf{f}(\mathbf{X}_p)$ is called the *Pareto optimal front*.

The principle of multi-objective optimization and Pareto optimality is presented in Figures 3.1 and 3.2.

3.2 Multi-objective algorithms review

Multi-objective optimization methods can be divided into three main groups according to the information available before the optimization process [Collette and Siarry, 2004]:

- *A priori* methods require knowledge of the problem, e.g. the preferences or the magnitude of objectives, the result is only one Pareto optimal solution usually.
- *A posteriori* methods do not need any special information about the problem, their goal is to approximate the Pareto optimal front with the finite number of solutions.
- In *progressive* methods (also called *interactive*) the preferences are specified during the optimization process thanks to the interaction with the user. These methods are beyond the scope of this thesis.

A priori methods

Many mathematical techniques have been developed in order to deal with multi-objective optimization problems [Miettinen, 1999], mostly in the field of Operational Research, during the last five decades. However, these ”classical” algorithms have a number of limitations. For example, some of them require differentiability and continuity of objective functions. Some of them are sensitive to the shape of the Pareto optimal front. Moreover, these techniques find only one solution; therefore, the algorithm needs to be executed several times from different starting points (or with different settings) in order to obtain more Pareto optimal solutions.

These methods are usually based on some transformation of MOP into SOP and this can be solved by any method available for SOP. The transformation can be done in two basic ways:

- Transformation using constraints: All objectives but (the most important) one are transferred into constraints and the constrained SOP is solved. Of course, the wrong choice of the optimized objective or an unrealistic setting of constraints can lead to bad results.

- Transformation without constraint: All objectives are aggregated into one function. From these methods the most popular one is probably the *weighted sum method* (**WSM**), see 3.3.1. Another well known method is the *utility function method*, where the user must provide a function relating all objectives and this function is to be maximized. Also, *goal programming* should be mentioned; in this method the weighted sum of differences of objectives from desired values is minimized.

  The method called *lexicographic ordering* does not really transform objectives, but solves them consecutively according to their importance. At first, the most important objective is optimized without considering any of the others; then, the second objective is optimized, but without decreasing the quality obtained for the first objective. This process is repeated for all remaining objectives. This method is easy to implement but has many drawbacks: to determine the importance of objectives can be a very difficult task and, again, with one setting only one Pareto optimal point is achieved.

A posteriori methods

The goal of these methods is to find a good approximation of the true Pareto optimal front. The good approximation involves two (conflicting) properties: the solutions must be as close to the Pareto optimal front as possible and well spread in the objective (or parameter) space to represent the entire range of the Pareto optimal front.

To reach this goal, a priori methods can be employed repeatedly, i.e. more runs with different settings provide more Pareto optimal solutions.

Also, the evolutionary algorithms mentioned above are capable to solve this problem. EAs work with populations, therefore, with a single run more Pareto optimal solutions can be obtained. Moreover, they do not need differentiable or continuous objective functions. The possible use of EAs for MOP was proposed at first by Rosenberg in 1967 [Rosenberg, 1967], but actually no MOP optimizer was developed. As the first multi-objective evolutionary algorithm (**MOEA**), the *Vector Evaluated Genetic Algorithm* (**VEGA**) designed in mid-1980s by David Schaffer [Schaffer, 1984] is considered.

With VEGA the so called “first generation of MOEAs” [Coello, 2006] started. In this period, the above mentioned (weighted sum method and lexicographic ordering) and other “simple” methods were adopted to work with EAs. Also, more sophisticated algorithms, based on a direct implementation of the Pareto optimality concept were developed. The most known algorithms from this period are the *Multi-Objective Genetic Algorithm* (**MOGA**) [Fonseca and Fleming, 1993], the *Nondominated Sorting Genetic Algorithm* (**NSGA**) [Srinivas and Deb, 1994] and the *Niched-Pareto Genetic Algorithm* (**NPGA**) [Horn et al., 1994].

The second generation of MOEAs started in the late 1990s and is characterized by the implementation of elitism. Elitism is a method, known from single-objective EAs, which guarantees the best individuals from the current generation to pass into the next generation without being affected by a crossover or a mutation. Thanks to elitism, the convergence is ensured. In SOP, elitism is easy to implement, because there is always just one best solution. In MOP, it is not so straightforward, because all Pareto optimal solutions in a population are equally good.

The most known algorithms from this generation are NSGA-II and SPEA2, which are often used as benchmarks which new algorithms are compared to. Plenty of EAs were developed and new ones are still arising, e.g. the Pareto archived evolutionary strategy PAES [Knowles and Corne, 2000] or multi-objective micro-GA [Coello and Pulido, 2001]. The interested reader is referred to [Coello et al., 2006] for a very extensive review of MOEAs. Recently, also other bio-inspired algorithms such as Simulated Annealing [Ulungu et al., 1999], Particle Swarm [Coello and Lechuga, 2002] or Ant Colony [Mariano and Morales, 1999] optimizations are employed for multi-objective optimization problems.

To conclude this section, it must be emphasized that the difference between a priori and a posteriori methods is not only in the information available before the optimization procedure, but also in the output from the process. While in the case of a priori methods (which are in fact SOP), the user receives one solution found by any algorithm, in the second case there is a set of different, equally good solutions and the user must decide and choose the one most suitable for his/her purpose (i.e. after MOP a multi-criteria decision making problem must be solved). For a small review on decision making concerning evolutionary multi-objective optimization, see e.g. [Coello, 2000]. Recently, research on incorporating user preferences in the search process of MOEA has started, see [Rachmawati and Srinivasan, 2006] or [Jaszkiewicz and Branke, 2008].

For testing the proposed approach to parameter estimation, two already mentioned evolutionary algorithms, NSGA-II and SPEA2, were chosen. To these algorithms, the traditional weighted sum method and the genetic algorithm based on objectives' ranking are added. Their main features and particular implementation are discussed in the next sections.

3.3 Implementation

Before the multi-objective algorithms used for computations are presented in detail, the framework for their implementation will be described. The source code in C++ is available at `http://mech.fsv.cvut.cz/~zuzanka/COMA.html`.

The main loop of optimizers is based on a standard real coded genetic algorithm (see Figure 3.3). The scheme is common for all optimizers, only the environmental selection and the fitness assignment necessary for the mating selection differs for individual algorithms. Note that elitism is ensured in the environmental selection step for all algorithms, because the best solutions are always copied to the next generation. As the stopping criterion, the maximum number of generations is used. The steps are described in detail in the following paragraphs.

Initial population

At first the (*Pop*+*Off*) individuals are created randomly with a uniform distribution in desired limits and their objective function values are calculated. *Pop* and *Off* are configurable parameters of the algorithm. *Pop* is the number of solutions which are chosen in the environmental selection step, and then *Off* individuals from the survivors are chosen for the variation. As the last step of the whole optimization process is the environmental selection, *Pop* is also the number of solutions presented as a result.

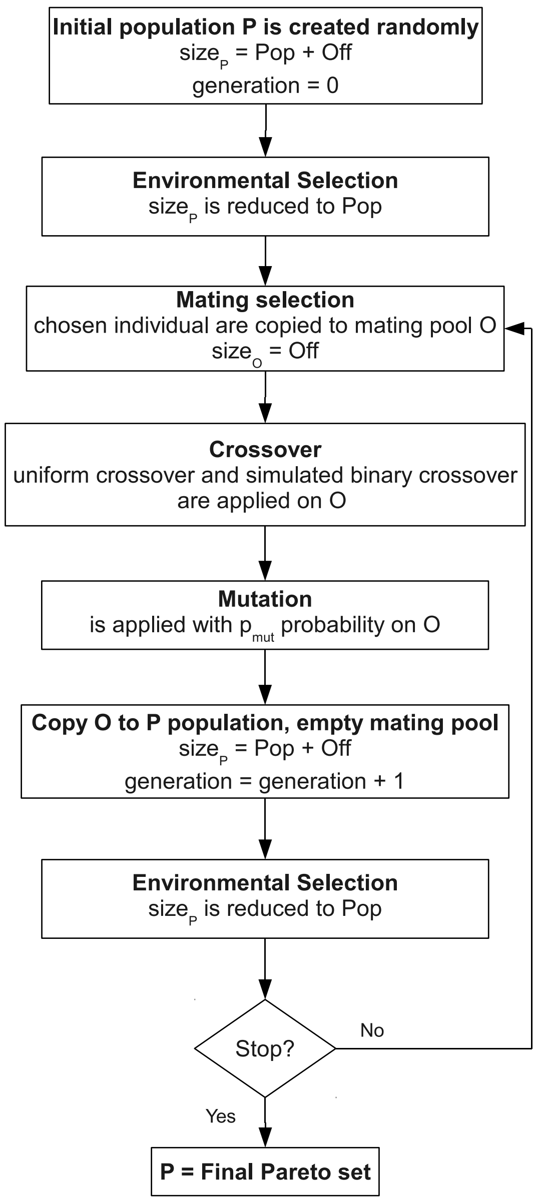
Initial population P is created randomly
size_P = Pop + Off
generation = 0
Environmental Selection
size_P is reduced to Pop
Mating selection
chosen individual are copied to mating pool O
size_O = Off
Crossover
uniform crossover and simulated binary crossover are applied on O
Mutation
is applied with p_mut probability on O
Copy O to P population, empty mating pool
size_P = Pop + Off
generation = generation + 1
Environmental Selection
size_P is reduced to Pop
Stop?
No
Yes
P = Final Pareto set

Figure 3.3: Main loop of used MOEA.

In most studies of genetic algorithms, both parameters have the same value. This thesis is focused on a small population size (i.e. *Pop* = 20 - 40 individuals) to simplify the final decision making step and to shorten the runtime of the optimization process. However, for these cases, i.e. *Pop* = *Off*, the population tends to converge very fast into a single point. It is preferable to have fewer offsprings than parents, and therefore, *Off* was set at $0.5 * Pop$ for all computations. The setting of the *Pop* parameter should also take into account the number of design variables, to create the initial population spread in the whole search space.

Mating selection

For the mating selection, the bi-tournament selection with replacement is used, i.e. two solutions are chosen randomly and the winner of the tournament is copied to the mating pool. The solution can be chosen for a tournament repeatedly, therefore, at the end of the mating selection, there can be more than one copy of good solutions in the mating pool. The winner of a tournament is determined by the so called fitness function, which differs for individual algorithms and is presented later. Note that in SOP the fitness function and the objective function are usually the same, whereas in MOP, the fitness function has to be calculated separately because of the existence of more objective functions.

Crossover

After the mating pool is generated, the crossover is performed in two steps. Genetic algorithms were originally binary coded, which brings many problems with precision and coding. Therefore, real coded GAs were proposed. They overcome the problem with coding (i.e. the variable value of a real problem can be directly a gene of GA), but, on the other hand, the implementation of variation operators is not so straightforward. However, many types of crossover and mutation were developed for real coded GAs. Two types of crossover are used in this thesis: a naive uniform crossover [Deb, 2001] and a simulated binary crossover presented in [Deb and Agrawal, 1995].

The first type of crossover creates the offspring by the change of the parents' genes between each other. The probability of a swap for each gene is p_{swap}.

Then, the second type of crossover is used with the probability p_{SBX}. The operator (called the simulated binary crossover or **SBX**) was designed to have the same properties and search power as the one-point crossover used in binary-coded GAs:

- The average of the parents' genes values is the same as the children's ones.
- The difference between the offspring's genes (x_i^{o1}, x_i^{o2}) is in proportion (so called

spread factor) to the parent solution (x_i^{p1}, x_i^{p2}):

$$\beta = \left| \frac{x_i^{p1} - x_i^{p2}}{x_i^{o1} - x_i^{o2}} \right| . \tag{3.3}$$

- Solutions near parents are more likely to be chosen as offspring's than distant solutions. The probability distribution used to create children solutions is given by:

$$P(\beta) = \begin{cases} 0.5(n_{SBX}+1)\beta^{n}_{SBX}, & \text{if } \beta \le 1 \\ 0.5(n_{SBX}+1)\frac{1}{\beta^{n_{SBX}+2}}, & \text{otherwise} \end{cases} , \tag{3.4}$$

  where n_{SBX} is any non-negative real number. The higher value of n_{SBX} gives a higher probability of creating children near parents. Note that whereas in single objective evolutionary algorithms n_{SBX} is usually set at 2, in MOEAs this parameter is usually set higher $n_{SBX} \in [5, 20]$.

The SBX procedure step by step:

1. Random number $u \in [0, 1)$ is chosen.

2. $\bar{\beta}$ is calculated by equating the area under the probability function 3.4 equal to u:

$$\bar{\beta} = \begin{cases} (2u)^{\frac{1}{n_{SBX}+1}}, & \text{if } u \le 0.5 \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{n_{SBX}+1}}, & \text{otherwise} \end{cases} \tag{3.5}$$

3. Children solutions are calculated:

$$x_i^{o1} = 0.5\left[(1+\bar{\beta})x_i^{p1} + (1-\bar{\beta}x_i^{p2})\right],$$
$$x_i^{o2} = 0.5\left[(1-\bar{\beta})x_i^{p1} + (1+\bar{\beta}x_i^{p2})\right]. \tag{3.6}$$

Mutation

After the crossover step, a mutation is applied with the probability p_{mut}. The mutation with a polynomial distribution proposed in [Deb and Goyal, 1996] is implemented. The current value of the variable is changed to a nearby value using a polynomial probability distribution with the mean at the current value:

$$P(\delta) = 0.5(n_m + 1)(1 - |\delta|)^{n_m}, \tag{3.7}$$

where n_m is a parameter of the mutation. A higher value of n_m creates a steeper probability distribution, i.e. the probability of a big mutation is lower.

The steps of the mutation operator are similar to SBX:

1. Random number $u \in [0, 1)$ is chosen.
2. $\bar{\delta}$ is calculated by equating the area under the probability function 3.3 equal to u:
$$\bar{\delta} = \begin{cases} (2u)^{\frac{1}{n_m+1}} - 1, & \text{if } u < 0.5 \\ 1 - [2(1-u_m)]^{1/(n_m+1)}, & \text{if } u \geq 0.5 \end{cases} \tag{3.8}$$
3. The new value of the gene i is calculated:
$$x_i = x_i + (x_i^{max} - x_i^{min})\bar{\delta}. \tag{3.9}$$

After the mutation step is finished, the values of objective functions calculated for new individuals.

Optimizer's parameters

To conclude this section, the summary of parameters necessary to be set, their meaning and feasible values are resumed in Table 3.2. Also, the parameter values used for all computations presented further are stated. Note that for selection steps of individual algorithms no special parameters are needed.

3.3.1 Weighted Sum Method

As was already mentioned, this is the most popular classical method. Each objective is multiplied by a user defined weight and their sum is optimized, instead of solving equa-

Parameter	Meaning	Possible values	Used
Pop	parent population size	> 0	-
Off	offspring population size	> 0	0.5**Pop*
p_{swap}	probability of uniform crossover	[0,1]	0.2
p_{SBX}	probability of SBX	[0,1]	0.9
n_{SBX}	SBX distribution parameter	[1,∞)	15
p_{mut}	probability of mutation	[0,1]	$1/Dim$
n_{mut}	mutation distribution parameter	[0,∞)	20
$generation_limit$	stopping criterion	> 0	-

Table 3.2: Algorithm parameters summary. Dim is the number of decision variables. Note that contrary to GRADE algorithm the probability of variation operators is applied for each gene.

tion 3.2. The problem is converted into:

$$\text{minimize } F(\mathbf{x}) = \sum_{i=1}^{k} w_i f_i(\mathbf{x}), \tag{3.10}$$

where w_i is the weight of the i-th objective function. Usually, the weights are chosen such that their sum is equal to one.

Although this method is really easy and intuitive, the biggest obstacle is obvious: setting the weights. The weights express the relative importance of individual objectives, which, in real world applications, is difficult to determine. The success of WSM also depends on the scaling of objectives; all of them should have more or less the same order of magnitude to affect the value of $F(\mathbf{x})$ similarly. Therefore, not only the weight vector must be set, but also the normalization of objectives must be performed. Similarly to the weight vector, it is difficult to determine in advance which objectives' values can be reached and, accordingly, to properly set the normalization vector.

It is shown in [Miettinen, 1999] that for a convex problem WSM is able to find solutions on the entire Pareto optimal front. Unfortunately, nonconvex problems are unbeatable for WSM [Deb, 2001], see Figure 3.4.

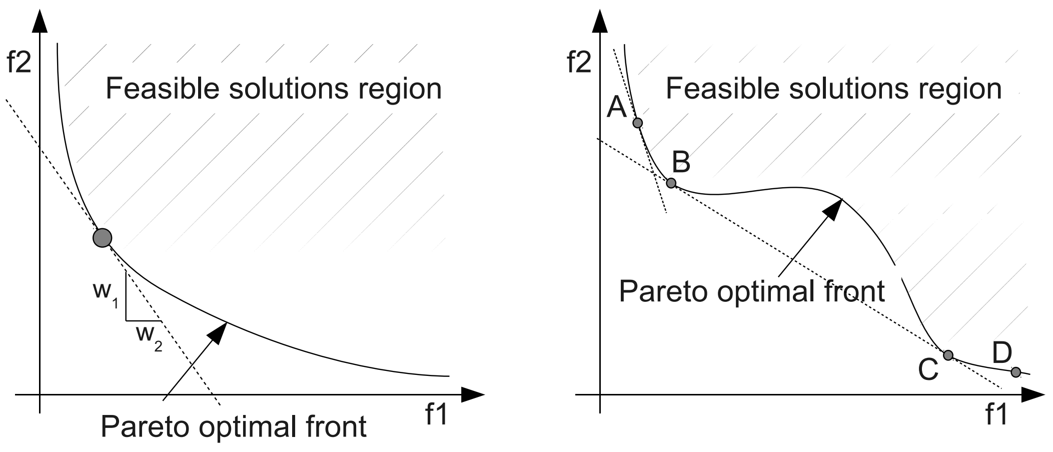

Figure 3.4: The principle of weighted sum method. The weight vector determines the slope $(-w_1/w_2)$ of lines with the same value of $F(\mathbf{x})$. With moving the line from right to left the $F(\mathbf{x})$ value is decreasing; the minimum comes with a line which is tangential to the feasible space, see the left picture. With different weight vectors, all Pareto optimal solutions in left picture can be found. On the contrary, in the right picture, with a nonconvex shape of the Pareto optimal front, no weight vector can produce a tangent point within the region BC. [Deb, 2001]

Implementation

The implementation of this method is very straightforward: in the environmental selection step, *Pop* individuals with the lowest value of $\mathbf{F}(\mathbf{x})$ survive. And it is again the $F(\mathbf{x})$ value, which determines the winner in the tournaments during mating selection. The result of one optimization run is only one solution- the one with the lowest $\mathbf{F}(\mathbf{x})$ value.

3.3.2 Average Ranking Method

The *Average ranking method* (**AR**) [Bentley and Wakefield, 1998] is another algorithm which merges all objectives into one function. But, contrary to WSM, this method is range-independent, does not require any a priori information and can find more Pareto optimal solutions in one optimization run. Another difference is the ability of AR to find the Pareto optimal solution even for nonconvex functions. On the other hand, this method does not ensure the identification of nondominated solutions; moreover, such solutions can be lost in

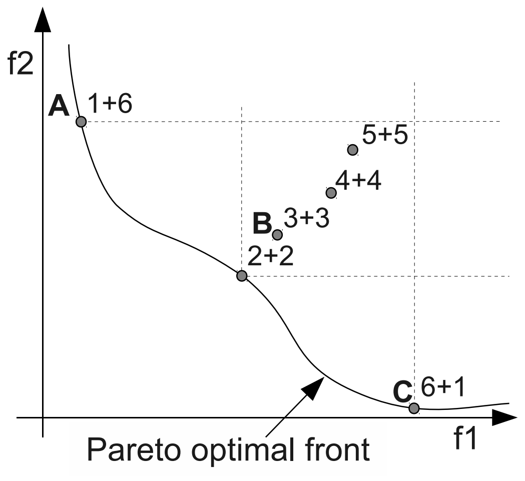

Figure 3.5: Principle of average ranking method. One can see that the method can fail in determining nondominated solutions, i.e. the dominated solution B has a lower average rank than nondominated solutions A and C.

the next generations.

Solutions are sorted according to individual objectives and the value corresponding to its order is assigned to the particular solution, see Figure 3.5. Then, the fitness value for each solution is calculated as an average of all its ranks. The method prefers solutions which dominate more solutions than others do.

Implementation

The implementation of AR is not complicated either: The value of the average rank determines which solution survives the environmental selection and wins the tournament. The result of the optimization process are *Pop* solutions with the lowest average rank.

3.3.3 Nondominated Sorting Genetic Algorithm II (NSGA-II)

This method was proposed in [Deb et al., 2000] as an improved version of the earlier NSGA algorithm. The main problems of NSGA were: a computationally demanding algorithm for the sorting of solutions, no elitism and the necessity of determining the sharing parameter. The improved version eliminates all these obstacles. The quality of a single individual p is determined by two attributes: the nondominated rank p_{rank} and the crowding

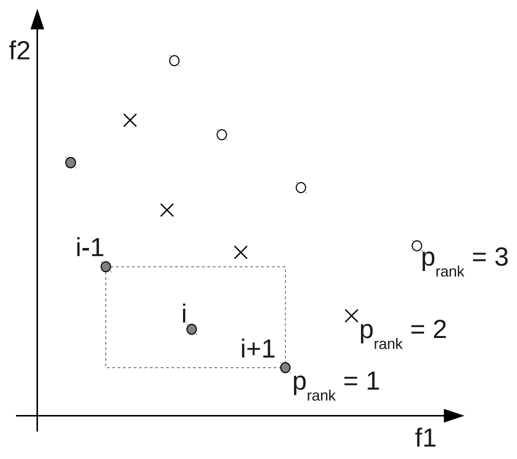

Figure 3.6: Nondominated rank and crowding distance calculation for NSGA II.

distance $p_{distance}$.

The *nondominated rank* determines the order of the Pareto optimal front to which the individual belongs; it expresses the position of the individual in the objective space regarding the Pareto optimality and guides the convergence to the true Pareto optimal front, see Figure 3.6. As the sorting procedure the authors proposed so called "fast nondominated sorting": At first, for each solution p the domination count n_p is determined, i.e. the number of solutions which dominate the solution p and a set S_p of solutions which are dominated by the solution p. Obviously, nondominated solutions have a domination count equal to zero. Thereafter, $p_{rank} = 1$ is assigned for all solutions with $n_p = 0$, their S_p is visited and the domination count of all its members q is reduced by one. If any $n_q = 0$ after the reduction, it is placed in a separate list Q. This step is repeated with the members of Q and the next front is identified. Although this sorting procedure is fast, i.e. it has a lower time complexity than the naive approach t sorting, it has higher memory demands.

The *crowding distance* attribute determines the density of solutions in the neighbourhood of an individual, see Figure 3.6, and it maintains the diversity within the Pareto optimal front. Before the computation, the population is sorted in an ascending order according to each objective. Then, for each objective, the infinite distance is assigned to the boundary solutions (i.e. solutions with the lowest and the highest values). Thanks to this assignment, the boundary points are always selected. For all other solutions, the distance value is computed as

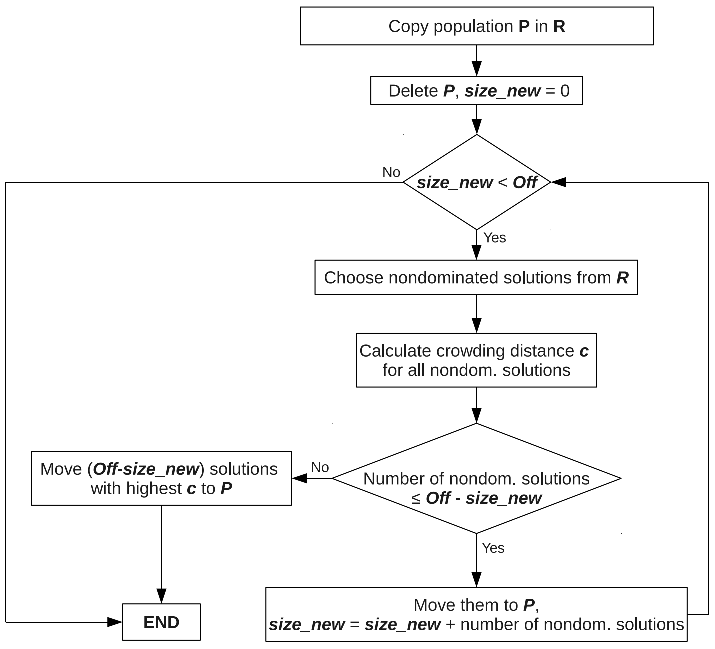
Copy population **P** in **R**
Delete ***P***, ***size_new*** = 0
size_new < ***Off***
No
Yes
Choose nondominated solutions from ***R***
Calculate crowding distance ***c*** for all nondom. solutions
Number of nondom. solutions ≤ ***Off*** - ***size_new***
No
Yes
Move (***Off-size_new***) solutions with highest ***c*** to ***P***
Move them to ***P***, ***size_new*** = ***size_new*** + number of nondom. solutions
END

Figure 3.7: Environmental selection of NSGA II.

an absolute normalized difference between two adjacent solutions. At the end, the crowding distance is a sum of distance values for all objectives.

Implementation

The environmental selection is shown in Figure 3.7. For the mating selection, there is not one fitness function value, but both the above mentioned attributes are used. At first, the *nondomination rank* of individuals chosen for the tournament is compared. If the rank is different, the one with the lower rank is copied to the mating pool. In the case that both individuals belong to the same Pareto optimal front, the one with a higher *crowding distance* (i.e. the one from a less crowded region) will be part of the recombination.

3.3.4 Strength Pareto Evolutionary Algorithm 2 (SPEA2)

Similarly to NSGA-II, SPEA2 was proposed in [Zitzler et al., 2001] as an improved version of an earlier SPEA algorithm. In SPEA2, each individual i is assigned a fitness value $F(i)$, composed of two parts. The first one, so called *raw fitness* $R(i)$, is determined by the strength of individuals with regard to the Pareto dominance, while the second part, *density*

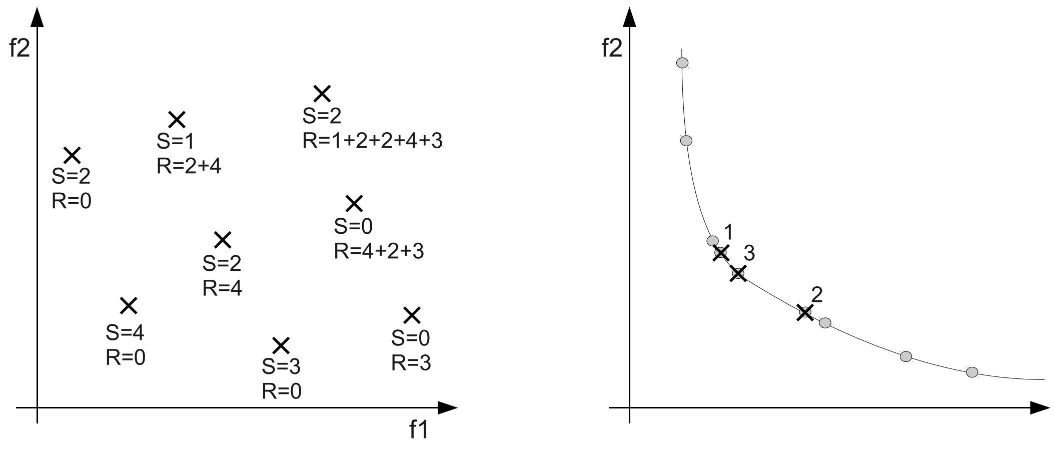

Figure 3.8: Left: Strength (S) and raw fitness (R) calculation. Right: The truncation procedure, numbers determine the order in which solutions will be rejected.

$D(i)$, expresses the quality of the individual regarding the proximity of the other solutions.

To calculate $R(i)$, the strength $S(j)$ is determined for all individuals in the population P as a number of solutions which dominates j. Then, raw fitness is computed as:

$$R(i) = \sum_{j \in P, j \succ i} S(j), \tag{3.11}$$

i.e. the strength of the individual is given by the sum of the strength of its dominators. Therefore, nondominated solutions have $R(i) = 0$, see Figure 3.8.

For density, the second part of $F(i)$, the k-th nearest neighbour method is adopted in SPEA2. The distances to all individuals j (in the objective space) are quantified for each individual i. Then, these distances are sorted in increasing order. The k-th element of this sorted list (σ_i^k) is used for the density calculation:

$$D(i) = \frac{1}{\sigma_i^k + 2} \tag{3.12}$$

The constant 2 is added to the denominator to ensure that $D(i) \in (0, 1)$. The value of k is determined by the square root of the population size: $k = \sqrt{Pop}$.

Finally, the fitness $F(i)$ can be calculated:

$$F(i) = R(i) + D(i). \tag{3.13}$$

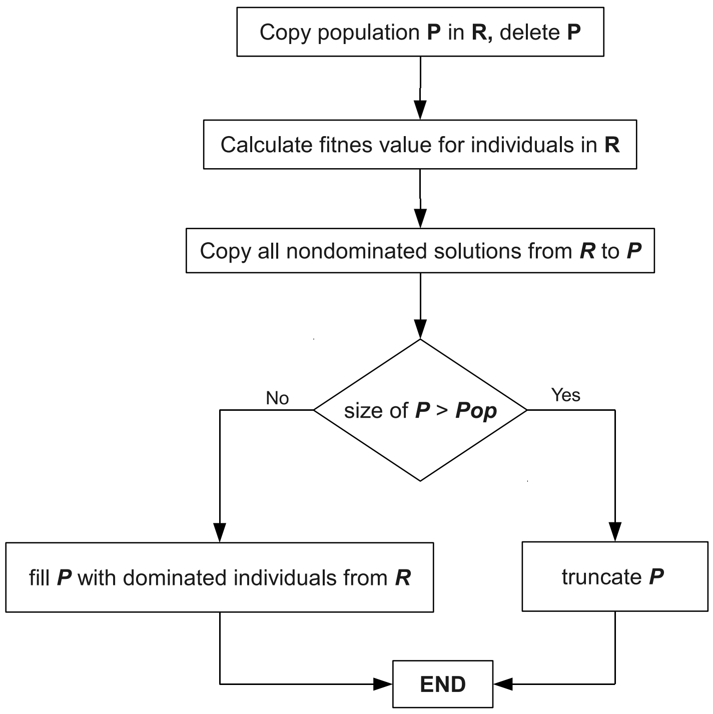
Copy population P in R, delete P
Calculate fitnes value for individuals in R
Copy all nondominated solutions from R to P
size of P > Pop
No
Yes
fill P with dominated individuals from R
truncate P
END

Figure 3.9: Environmental selection of SPEA2.

In the environmental selection step, nondominated solutions (individuals with $F(i)$ lower than one) are copied to the new population. If their number n_{nondom} is equal to the population size Pop, the step is completed. But this is very rare; usually the new population is too small ($Pop > n_{nondom}$) or too big ($Pop < n_{nondom}$). In the first case, the population is filled up to Pop individuals with dominated solutions. The original population is sorted in increasing order according to theindividual's fitness value and the first $(Pop - n_{nondom})$ individuals with $F(i) > 1$ are added to the new population. In the second case, the non-dominated set must be truncated: Solutions are removed iteratively from the nondominated set until $(n_{nondom} = Pop)$. In each iteration, the solution with the minimum distance to the other solutions is removed. If there are two solutions with the same minimum distance, the solution with the second smallest distance is rejected and so forth, see Figure 3.8 for a better understanding of the truncation procedure.

Implementation

The environmental selection is shown in Figure 3.9 and all steps are described above. In the mating selection, the winner of a tournament is the solution with a lower fitness value.

3.4 Multi-objectivization and Helper-objectives

The popularity of multi-objective optimization has been increasing during the last decades. Thanks to it, publications are appearing in the last years pointing out that reformulating a single-objective optimization problem (**SOP**) in terms of more objective functions can reduce the runtime or increase the efficiency of the optimization process.

This idea has firstly appeared for solving constrained SOP, which is transformed into an unconstrained multi-objective optimization problem. Two main approaches to the transformation can be distinguished [Mezura-Montes and Coello, 2006]: Another function determined by the sum of constraint violations is added to the original objective function, or individual constraints are transformed into additional objectives.

The approach to the parameter estimation proposed in the thesis is inspired by an idea presented in [Knowles et al., 2001]. Here, the authors introduced a way how to deal with the multi-modality of a single-objective problem. The authors suppose that expanding the SOP problem to the multi-objective space gives an algorithm more freedom to explore and decreases the probability to be trapped in local minima. They call the method *multi-objectivization*: the original objective is replaced with a set of new objectives or new objectives are added to the original one. In either case, it is necessary to ensure the global optimum of the original problem is one of the Pareto optimal points of the new problem. As a solution to the original problem, the one with the lowest original objective from the Pareto optimal front is chosen.

Two examples are solved in the referenced article: the hierarchical-if-and-only-if function and the travelling salesman problem. As optimizers a simple hill-climbing algorithm and simulated annealing for the original single-objective problem and the Pareto archived evolutionary strategy (PAES) [Knowles and Corne, 2000] for the new, reformulated, problem are used. The results show improvements with respect to the single-objective strategy. The theoretical runtime analysis of the multi-objectivization method on a plateau-function is presented in [Brockhoff et al., 2007]. The authors conclude that adding objectives can both speed up or slow down the optimization process. In extreme cases, the effect can make a difference between a polynomial and exponential runtime.

A similar method was proposed by Jensen [Jensen, 2004]. The author uses additional *helper-objectives* to guide the search of evolutionary algorithms in high-dimensional spaces. In the cited paper, it is shown how the solving of the problem as a multi-objective one can lead to the decrease or even the disappearance of difficulties known from the single-objective optimization, such as falling into local minima or decreasing diversity. The main difference from the multi-objectivization theory is the emphasis on the conflict between the original objective and helper-objectives, which is, according to the author, necessary to maintain the diversity. In the referenced paper, the proposed concept is tested on the job shop scheduling problem and the travelling salesman problem. Presented results show an improvement compared to the single objective optimization. The author recommends to use one dynamic helper-objective as the most promising approach, since using too many helper-objectives at the same time decreases the selection pressure of the algorithm.

The helper-objective approach was later used by Greiner et al. [Greiner et al., 2007] in a real design optimization problem belonging to the field of computational mechanics. Particularly, the bar frame optimum design problem of constrained mass minimization was solved. The authors compare single objective optimization with three multi-objective algorithms. The best overall average results were achieved with the multi-objective approach.

Chapter 4

PERFORMANCE ASSESSMENT

4.1 Test problems

To identify the algorithm suitable for the multi-objective parameter estimation, the behaviour of algorithms was studied on a set of known multi-objective test problems; *ZDT1 - ZDT6* functions are from [Zitzler et al., 2000] and *dtlz2a - dtlz4a* from [Knowles, 2006]. They are designed to test the MOEAs' ability to deal with particular difficulties, which can occur in real world problems. All presented problems are to be minimized. Note that constrained problems are not tested, because parameter estimation problems are not constrained either.

The ZDT1 test function has a convex Pareto optimal front:

$$\begin{aligned} f_1(x_1) &= x_1 \\ f_2(\mathbf{x}) &= g\left(1 - \sqrt{f_1/g}\right) \\ g(x_2, ..., x_n) &= 1 + 9\sum_{i=2}^{n} \frac{x_i}{n-1}, \end{aligned} \tag{4.1}$$

where $n = 30$ and $x_i \in [0, 1]$. The Pareto optimal front is formed with $g(\mathbf{x}) = \mathbf{1}$.

The ZDT2 test function is the nonconvex counterpart to ZDT1:

$$\begin{aligned} f_1(x_1) &= x_1 \\ f_2(\mathbf{x}) &= g\left(1 - (f_1/g)^2\right) \\ g(x_2, ..., x_n) &= 1 + 9\sum_{i=2}^{n} \frac{x_i}{n-1}, \end{aligned} \tag{4.2}$$

where $n = 30$ and $x_i \in [0, 1]$. The Pareto optimal front is formed with $g(\mathbf{x}) = \mathbf{1}$.

The ZDT3 test function examines the ability of MOEAs to tackle discontinuity in the Pareto front; it consists of several non-contiguous convex parts:

$$\begin{aligned} f_1(x_1) &= x_1 \\ f_2(\mathbf{x}) &= g\left(1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1)\right) \\ g(x_2, ..., x_n) &= 1 + 9\sum_{i=2}^{n} \frac{x_i}{n-1}, \end{aligned} \tag{4.3}$$

where $n = 30$ and $x_i \in [0, 1]$. The Pareto optimal front is formed again with $g(\mathbf{x}) = \mathbf{1}$.

The ZDT4 test function contains 21^9 local Pareto optimal fronts and, therefore, tests the MOEAs' ability to deal with multi-modality:

$$\begin{aligned} f_1(x_1) &= x_1 \\ f_2(\mathbf{x}) &= g\left(1 - \sqrt{f_1/g}\right) \\ g(x_2, ..., x_n) &= 1 + 10(n-1) + \sum_{i=2}^{n}\left(x_i^2 - 10\cos(4\pi x_i)\right), \end{aligned} \tag{4.4}$$

where $n = 10$ and $x_1 \in [0, 1]$, and $x_2, ..., x_m n \in [-5, 5]]$. The global Pareto optimal front is formed with $g(\mathbf{x}) = \mathbf{1}$; the best local Pareto optimal front with $g(\mathbf{x}) = \mathbf{1.25}$.

The ZDT6 test function includes two difficulties caused by the nonuniformity of the search space: first, the Pareto optimal solutions are nonuniformly distributed along the global Pareto front (the front is biased for solutions for which $f_1(\mathbf{x})$ is near one); second, the density of solutions is the lowest near the Pareto optimal front and the highest away from the front:

$$\begin{aligned} f_1(x_1) &= 1 - exp(-4x_1)\sin^6(6\pi x_1) \\ f_2(\mathbf{x}) &= g\left(1 - (f_1/g)^2\right) \\ g(x_2, ..., x_n) &= 1 + 9\left(\frac{\sum_{i=2}^{n} x_i}{n-1}\right)^{0.25}, \end{aligned} \tag{4.5}$$

where $n = 10$ and $x_i \in [0, 1]$. The Pareto optimal front is formed with $g(\mathbf{x}) = \mathbf{1}$ and is nonconvex.

DTLZ2a and DTLZ4a:

$$
\begin{aligned}
f_1 &= (1+g)\cos(x_1^{\alpha}\pi/2)\cos(x_2^{\alpha}\pi/2) \\
f_2 &= (1+g)\cos(x_1^{\alpha}\pi/2)\sin(x_2^{\alpha}\pi/2) \\
f_3 &= (1+g)\sin(x_1^{\alpha}\pi/2) \\
g &= \sum_{i=3}^{n}(x_i-0.5)^2 \\
\alpha &= \begin{cases} 1, & in \quad DTLZ2a \\ 10, & in \quad DTLZ4a \end{cases}
\end{aligned}
\tag{4.6}
$$

where $x_i \in [0,1], n = 8$. The Pareto front is one eighth of a sphere of radius 1, centred on 0,0,0. The Pareto optimal set consist of all solutions where all but the first decision variables are equal to 0.5, and the first decision variable may take any values in [0,1]. The effect of setting $\alpha = 100$ is to severely bias the density distribution of solutions toward the $f_3 - f_1$ and $f_2 - f_1$ planes.

DTLZ7a

$$
\begin{aligned}
f_1 &= x_1 \\
f_2 &= x_2 \\
f_3 &= (1+g)h \\
g &= 1 + 9/6\sum_{i=3}^{n} x_i \\
h &= 3 - \sum_{i=1}^{2}\left[\frac{f_i}{1+g}\left(1+\sin(3\pi f_i)\right)\right],
\end{aligned}
\tag{4.7}
$$

where $x_i \in [0,1], n = 8$. This problem has four disconnected regions of the Pareto front.

4.2 Performance assessment

The goal of performance assessment is to determine whether an algorithm A gives overall better results than an algorithm B. Unlike in single-objective optimization problems, where the solution is only one individual, in MOP the solution is a set of individuals, a so called approximation set. Therefore, the decision whether one approximation set is better than another is not so easy. In the MOEA literature, many approaches are proposed to measure and

Problem	Dimension	Objectives	Properties
ZDT1	30	2	convex
ZDT2	30	2	nonconvex
ZDT3	30	2	convex, disconnected
ZDT4	10	2	nonconvex, multimodal
ZDT6	10	2	nonconvex, nonuniformly spaced
DTLZ2a	8	3	nonuniformly spaced
DTLZ4a	8	3	nonuniformly spaced
DTLZ7a	8	3	disconnected

Table 4.1: Test problems summary.

compare the MOEA's performance. They usually measure both the above mentioned goals of MOP, i.e. the proximity to the true Pareto front and the area covered by the approximation set. Anyway, different assessment methods can give different results for the same approximation set. Therefore, it is necessary to mention which method was used for the assessment; or more quality indicators can be used for the same approximation set and only if they give the same result, the result is assumed to be reliable.

The methods for performance assessment proposed in [Knowles et al., 2006] are used in the thesis. From presented indicators, two unary indicators[1] - the hypervolume and the epsilon indicator - and an empirical attainment function were chosen for comparison. All methods require the normalized approximation set which contains only nondominated solutions. For this preprocessing as well as for the computation of indicators PISA software tool, available at `http://www.tik.ee.ethz.ch/pisa/`, was employed.

Epsilon indicator

A binary indicator $I_\epsilon(A, B)$ determines the minimum factor ϵ by which each point of the approximation set B needs to be modified such that the transformed approximation set is weakly dominated by A. There are two versions of the epsilon indicator - additive and

[1] Unary indicators are defined as the mapping of the approximation set to the real number.

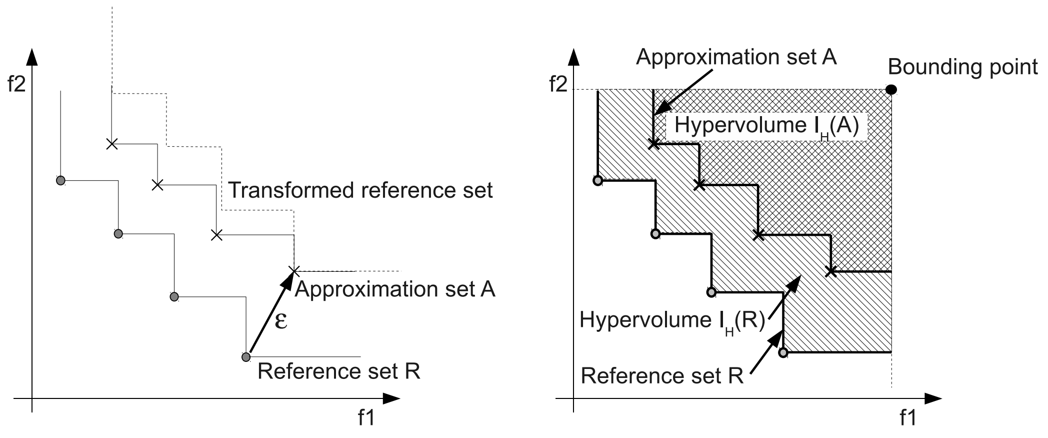

Figure 4.1: Unary indicators. Left: epsilon indicator. Right: hypervolume indicator.

multiplicative. The modification for an additive version of this indicator is adding the factor ϵ to the set B, for a multiplicative version the set B is multiplied by $I_\epsilon(A, B)$. To compare the optimizers among each other it is better to use a modified, unary, version of this operator:

$$I_\epsilon(A) = I_\epsilon(A, R), \tag{4.8}$$

where R is a reference set, which is the same for all optimizers. The lower value of the additive indicator $I_\epsilon(A)$ means a better performance. $I_{\epsilon+}(A)$ is used in this thesis, see Figure 4.1.

Hypervolume indicator

The hypervolume indicator I_H measures the hyperspace enclosed by the Pareto set approximation and is to be maximized. Note that the objective space must be bounded or a bounding point, which is at least weakly dominated by all points from the approximation set, must be chosen. Again, an unary version of the indicator exists to compare more optimizers:

$$I_H^-(A) = I_H(R) - I_H(A), \tag{4.9}$$

i.e. the modified version is defined as a difference between the hypervolume indicator for a reference set R and an approximation set A, see Figure 4.1. On the contrary to the original indicator, a lower value identifies a better approximation set. A value lower than zero means that the set A dominates the reference set R.

Empirical attainment function

Unlike unary indicators, the output of this performance measure is a function, so called attainment surface function. This indicator was chosen because its results can be plotted in the objective space and it provides better understanding of the obtained nondominated front and optimizers can be compared visually (of course only up to 3 objectives).

Because MOEAs are stochastic, the results of the optimizer from r runs can be described by a distribution. The empirical attainment function calculates a relative frequency that some objective vector was attained. Using this frequency distribution, a $k\%$ attainment surface can be derived and plotted. It divides the objective space into two parts: the goals that have been attained and the goals that have not been attained with a frequency of at least $k\%$ from r runs.

Statistics

As mentioned above, MOEAs are stochastic algorithms; therefore, the output can differ in each run. To evaluate the performance of individual optimizers, it is necessary to make more runs, calculate the above proposed indicators and then perform statistical analysis to make a conclusion. In [Knowles et al., 2006], a statistical analysis approach is proposed and the mentioned PISA software tool includes particular statistical tests. As the normal distribution of the resulting indicators cannot be assumed and more than two algorithms are compared, the Kruskal-Wallis rank test is used. The values of indicators are transformed into ranks and the hypothesis that the samples are not identical is tested. If this test is passed, the samples are pair-wise compared and one tailed p-value is computed.

4.3 Results on test problems

Population size

Studies which can be found in multi-objective literature usually deal with big populations (i.e about 100 individuals), but the task of multi-objective parameter estimation is not the same as a classical MOP. We are not interested in the whole Pareto front; in fact, we expect only one optimal solution, but, on the other hand, if this optimal solution is not found, the Pareto optimal solutions can give interesting information about the model and the user can choose the least poor solution. Therefore, the tests presented here are focused on smaller populations to simplify the final decision making phase and to decrease the computational time. With the number of solutions up to 40, the user can choose the best solution visually. Note that the term population size used here refers to the parameter Pop. All presented statistics are for 100 optimizer's runs.

As the WSM algorithm provides only one Pareto optimal solution in one run, for the performance and behaviour study, the algorithm was started Pop-times with a random weight vector for each run, to get the same size of the approximation set as with other algorithms. Of course, this approach is very time consuming and therefore, not applicable in real engineering applications.

At first, to decide whether and how much a small population deteriorates the MOEAs behaviour, the comparison of hypervolume and epsilon indicators for all test problems was provided for population sizes: 12, 20, 40 and 100. Figure 4.2 shows boxplots for indicators after 100 generations of NSGA2. As the number of objective function evaluations (and consumed time) is different for individual methods in this case, comparison is also shown for the same number of evaluations, see Figure 4.3. The results for other algorithms are presented in Appendix A, Figures A.1 - A.6. Note that a lower value of an indicator indicates a better performance regarding to this indicator. To mark any population size performance significantly better both indicators should be lower.

The population size 100 outperforms smaller population sizes for ZDT4, dtlz2a and dtlz4a problems for all algorithms. For the remaining test problems, a smaller population gives better or comparable results. Only the WSM algorithm shows higher sensitivity to the population size.

Another important parameter for the estimation of the population size effect is the time needed for optimization. Values for three different population sizes are stated in Table 4.2. Only three test problems representing all test problems are listed: ZDT1 has 30 variables

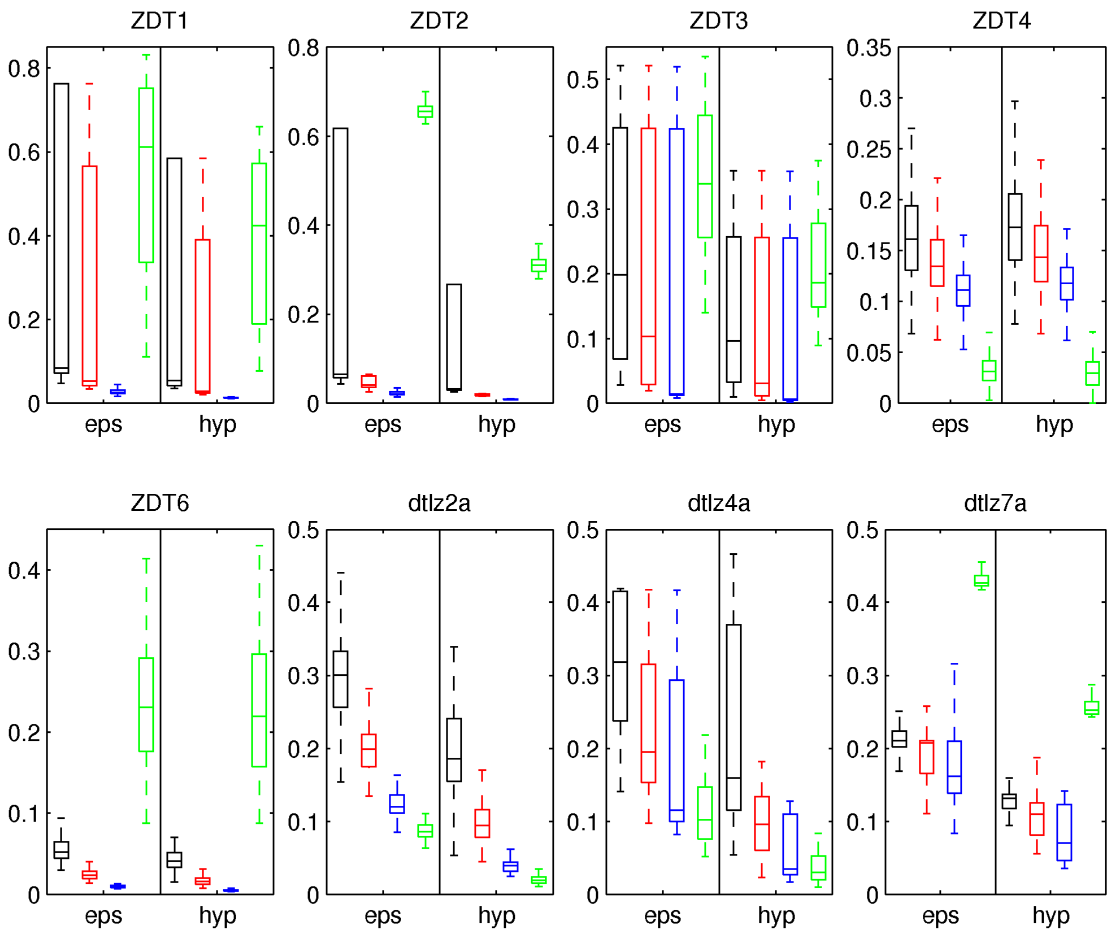

Figure 4.2: Influence of population size for NSGA-II algorithm - after 100th generation. Population size: black = 12, red = 20, blue = 40, green = 100

and 2 objectives, ZDT6 10 variables and 2 objectives and dtlz7a 8 variables and 3 objectives. Therefore, not only the effect of a population size but also the influence of variable or objective numbers can be studied. Figure 4.4 shows the time needed for dtlz7a function. Obviously, except the WSM algorithm times are not linear (in the case of SPEA2 algorithm even far from being linear).

Based on indicators and time analysis, it was decided that a higher precision which can be obtained with a bigger population is not so advantageous due to the necessary time required for the optimization and the decision making process. Therefore, all the following computations work with population sizes up to 40.

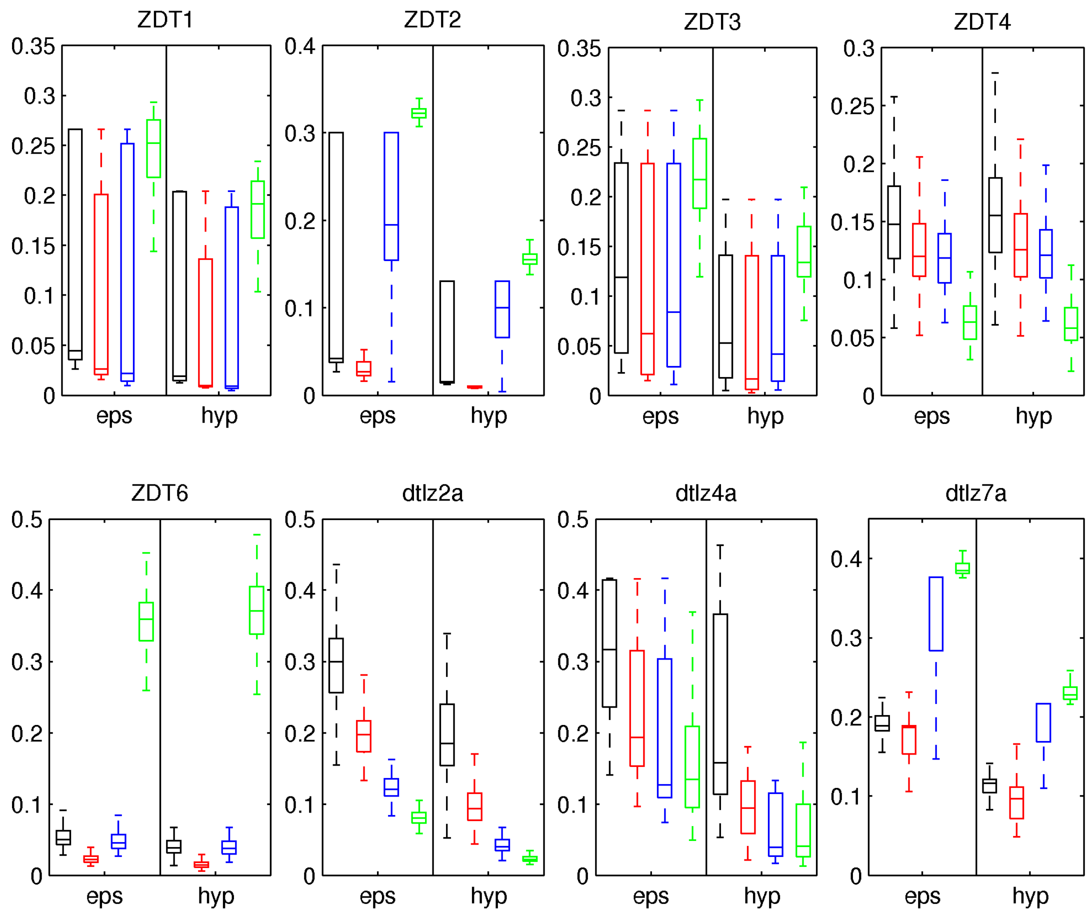

Figure 4.3: Influence of population size for NSGA-II algorithm - after 1000 function evaluations. Population size: black = 12, red = 20, blue = 40, green = 100

Performance of algorithms in time

The behaviour of individual algorithms during the optimization process and their comparison was studied, to get an idea which optimizer(s) can be useful for the parameter estimation. From above presented test problems, only four were chosen as representatives of the difficulties one can meet in a real engineering parameter estimation problem. One possible obstacle can be a relatively high number of decision variables; ZDT1 was chosen as a representative of this problem. Another probable situation in parameter estimation is a nonuniform density in the objective space - therefore, ZDT6 is the next function for detailed study. The next two functions represent three objectives problems - dtlz2a and dtlz7a - they again test the ability of MOEAs to deal with nonuniformly distributed solutions in the decision space. For

Algorithm	Problem								
	ZDT1			ZDT6			dtlz7a		
Population size	20	50	100	20	50	100	20	50	100
AR	0.039	0.162	0.512	0.020	0.111	0.354	0.020	0.121	0.548
NSGA-II	0.044	0.163	0.447	0.020	0.097	0.308	0.022	0.116	0.362
SPEA2	0.107	1.096	6.569	0.081	0.923	6.431	0.098	1.007	6.767
WSM	0.045	0.115	0.247	0.016	0.043	0.103	0.012	0.034	0.083

Table 4.2: Time consumption for test problems, average time for 100 generations.

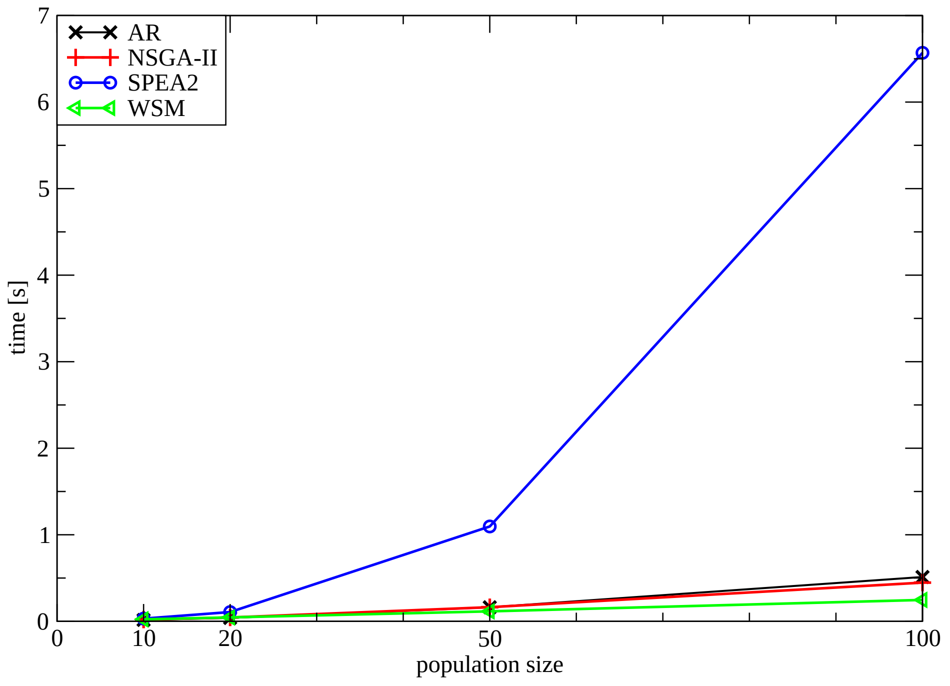

Figure 4.4: Time needed by each algorithm for 100 generations of dtlz7a optimization.

this testing, the population size was set at 40; all the other algorithm's parameters have values presented in Table 3.2. 100 runs were performed, for WSM 40 * 100 runs for reasons mentioned above.

As the first step of this comparison, epsilon and hypervolume indicators for the chosen generation were computed and the Kruskal-Wallis test was performed. In Table 4.3 a sample output for this test is shown. In the table are p-values for each pair of algorithms with respect to the alternative hypothesis that the indicator values for algorithms in rows are significantly better than those for algorithms in columns (significance level $\alpha = 0.05$). From the first part of the table one can see that all algorithms perform better than AR, moreover, NSGA-II and WSM perform better than SPEA2. The same conclusion can be made from the second part of the table for the hypervolume indicator.

	dtlz7a/$I_{\epsilon+}$				dtlz7a/I_H^-			
	AR	NSGAII	SPEA2	WSM	AR	NSGAII	SPEA2	WSM
AR	-	>0.05	>0.05	>0.05	-	>0.05	>0.05	>0.05
NSGAII	2.6e^{-84}	-	4.2e^{-30}	>0.05	2.2e^{-99}	-	3.0e^{-54}	>0.05
SPEA2	6.2e^{-32}	>0.05	-	>0.05	1.1e^{-23}	>0.05	-	>0.05
WSM	1.02e^{-84}	>0.05	2.7e^{-30}	-	3.1e^{-104}	>0.05	2.3e^{-59}	-

Table 4.3: Kruskal-Wallis test for dtlz7a function in 100th generation.

The analysis from these tables is not very convenient, therefore, a graphical representation of this test is presented. It is counted how many times each algorithm is better than the other algorithm with regard to both indicators, i.e. for results in Table 4.3 it is 2 for NSGA-II and WSM and 1 for SPEA2. These numbers are then plotted in the stacked bar graph. This is done for each watched generation and algorithm, see Figure 4.5 for the resulting graphs.

Another possibility for displaying and comparing the evolution of indicator values in time are boxplots, see Figure 4.6 for dtlz2a function, for other functions see Appendix A Figures A.7 - A.9. For better readability, the boxes are without outliers.

The results of NSGA-II, SPEA2 and WSM are comparable. WSM converge faster at the beginning, but, keep in mind, multiple runs are needed for these results. The performance of AR is very poor. The variance in the obtained results is high for the SPEA2 algorithm. Moreover, it seems SPEA2 has problems with the three objectives problem dtlz7a.

Next, the empirical attainment function for ZDT1 and for the ZDT6 function were computed, to observe the evolution of the Pareto front, see Figure 4.7 and 4.8 for 50% and 75% attainment surfaces. Note that for both problems the original function values are transformed

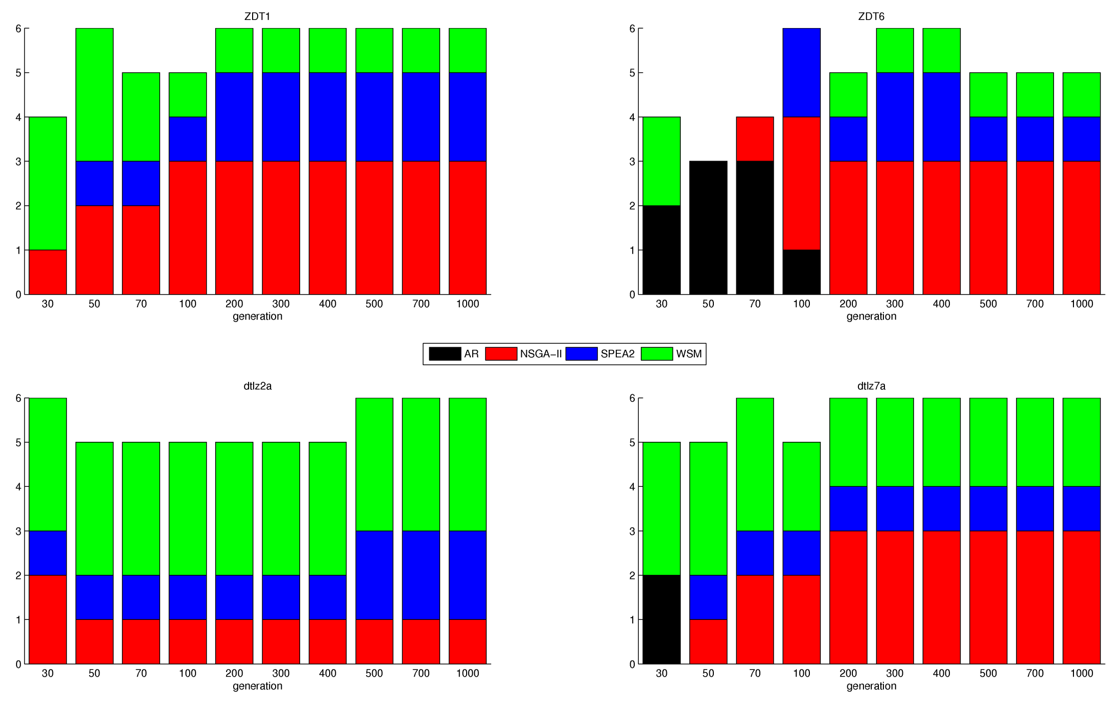

Figure 4.5: Graphical comparison of MOEAs performance in time.

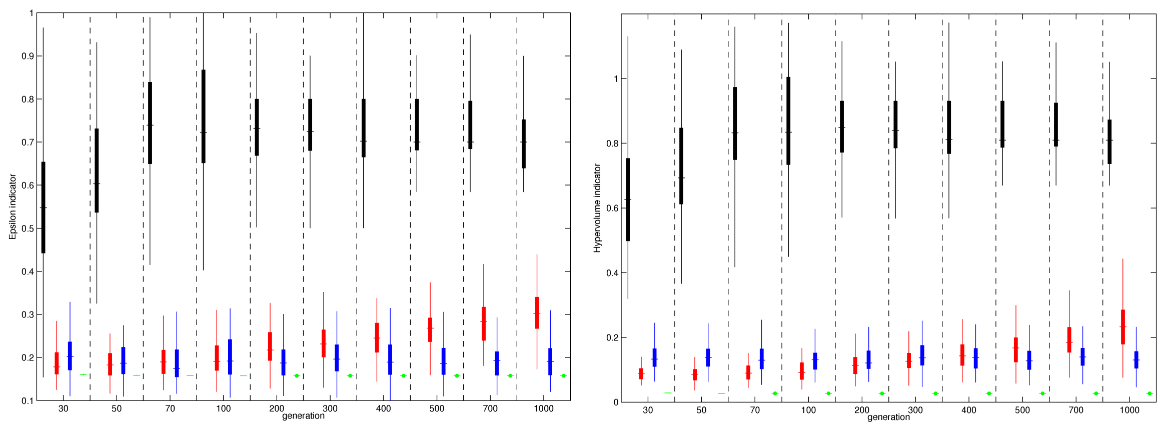

Figure 4.6: Evolution of epsilon (left) and hypervolume (right) indicator for dtlz2a function. Black: AR, red: NSGA-II, blue: SPEA2, green: WSM.

into the interval $[1, 2]$ required by the PISA algorithm. Therefore, the values do not correspond with the Pareto optimal front described in equations 4.1 and 4.5. These plots confirm the conclusions obtained from the indicator comparison, i.e. WSM and NSGA2 seem to be the most promising methods. For the ZDT1 problem, all algorithms except AR perform well; WSM converges faster than other algorithms but in the 100th generation their performance

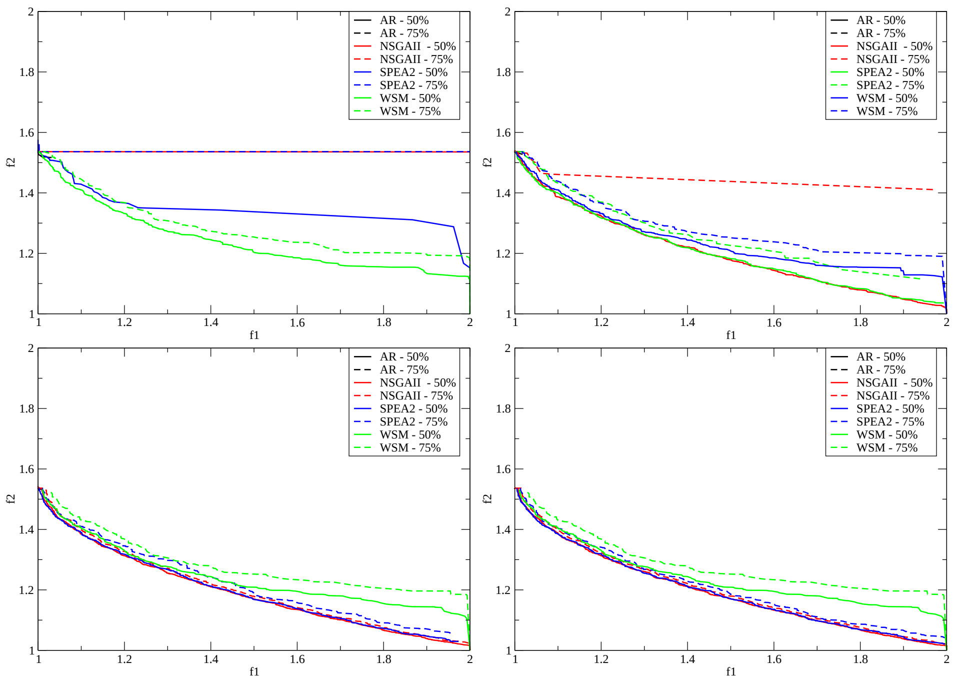

Figure 4.7: Attainment surface for ZDT1 function in 50 (top left), 100 (top right), 200 (bottom left) and 500 (bottom right) generations.

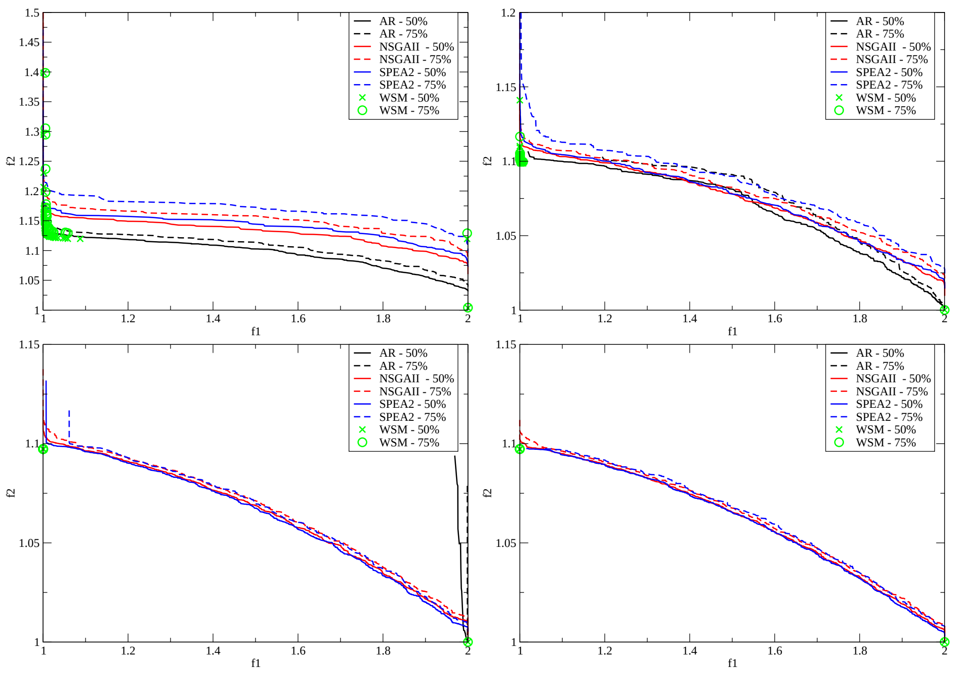

Figure 4.8: Attainment surface for ZDT6 function in 50 (top left), 100 (top right), 200 (bottom left) and 500 (bottom right) generations.

is almost the same. The AR method is able to find only solutions where f_1 has its optima, which is quite an easy task because f_1 is a function of only one variable.

In the ZDT6 problem, there are two important phenomena. At first, the inability of WSM to find a solution of nonconvex problems is clearly demonstrated. Only extreme solutions for single objectives were found (to show this, the WSM attainment surface is plotted only as marks). Second, the behaviour of AR must be mentioned; the algorithm performs really well at the beginning but then converges to extreme solutions leaving intermediate parts.

Part II

Real world applications

Chapter 5

MULTI-OBJECTIVE PARAMETER ESTIMATION

5.1 Methodology

As was stated in Chapter 1, the forward mode of the parameter estimation problem can be expressed by this equation:

$$\text{minimize } F(\mathbf{x}) = \|\mathbf{y}^E - \mathbf{y}^M)\| = \|\mathbf{y}^E - M(\mathbf{x})\|. \tag{5.1}$$

where $\mathbf{y}^E$ are experimental data and $\mathbf{y}^M$ the output of the model. The goal is to find **x** with which the minimum comes. As was noted earlier, this error function is often multi-modal.

Based on the multi-objectivization theory [Knowles et al., 2001], we propose to add additional objectives to the original error function. As the output of the model or experiment is usually some curve, the original function is usually represented by the Root Mean Square error (**RMS**). The easiest way for new objectives is adding error functions based on problematic or more important parts of the curve. If this knowledge is not available, a few runs of a traditional single-objective parameter estimation can be processed to determine these parts. Additional objectives can be the differences between the desired and simulated curve not only in the terms of the functions' values but also the difference in the x-axis values. The difference between the slopes of curves (i.e. numerical derivatives) can be used as well.

In the next two chapters, the proposed methodology will be tested. Therefore, all four algorithms are still taken into account. The testing algorithms phase will consist of:

1. *Verification*: The process of determination whether the estimation method is able to re-find the model parameters $\mathbf{x}^M$ from the outputs $\mathbf{y}^{ref}$ of a reference simulation done for any choice of $\mathbf{x}^{ref}$ from the domain. Here, two steps can be distinguished:

(a) comparing reference model inputs $\mathbf{x}^{ref}$ with the identified ones $\mathbf{x}^{M}$;

(b) comparing reference model outputs $\mathbf{y}^{ref}$ with the identified ones $\mathbf{y}^{M}$;

2. *Validation*: The process of determining whether the identification method is able to find the model parameters $\mathbf{x}^{M}$ corresponding to the experimental outputs $\mathbf{y}^{E}$. Here, only the experimental outputs $\mathbf{y}^{E}$ with the identified ones $\mathbf{y}^{M}$ can be compared.[1]

Note that in engineering practice, the parameter estimation process cannot be completely separated from the model verification and validation. Therefore, it is really difficult to judge if the obtained errors are caused by inaccuracy in the mathematical model or by inaccuracy of the estimation method. As the models used in this work are proposed by other authors, all of them will be a priori supposed to be already verified and validated.

Applications are focused on small populations to simplify the decision making process and on a small number of generations (or functions evaluations) to speed up the optimization process.

A graphical comparison for stochastic methods proposed in [Nosek and Lepš, 2009] is used for presenting the results. This method proposes a metric called Relative Winning Score (**RWS**) that counts only winners in some discrete steps of the optimization run. The winner is a method outperforming all the others in the particular run, i.e. for the method i:

$$RWS_i = \frac{\text{number of runs where } i \text{ is the winner}}{\text{number of runs}}. \tag{5.2}$$

As the sum of RWS for all compared methods is equal to one, the results can be plotted in the stacked bar graph, where the winner is clearly visible. On the other hand, this method does not give any information about the average or the worst values of objectives.

5.2 Existing Applications of Multi-Objective Parameter Estimation

Although multi-objective parameter estimation is not very common, there are some publications dealing with this approach. The presented problems are multi-objective by their

[1] Recall that physical experimental inputs $\mathbf{x}^{E}$ are practically always unknown.

nature, i.e. there are more experiments or measurements. Most of these papers do not solve the problem as multi-objective but use the weighted sum method or the min-max solution (i.e. the maximum of all objectives is to be minimized).

Mertens et al. [Mertens et al., 2006] present the multi-objective parameter estimation for a soil model. The authors use two different measurements to define objective functions based on the difference between the output from the model and the experiment. They use a weighted sum approach to merge both objectives and a specialized hydrology optimization algorithm is used for optimization.

Gendy and Saheb [Gendy and Saleeb, 2000] developed constitutive material estimation software COMPARE which enables determining material parameters for a hyper elastic large strain model. The objectives are based on a differential form of constitutive models and sensitivity analysis. The optimization objectives are merged using the weighted objective methodology and then optimized by a sequential quadratic nonlinear programming technique.

Reardon [Reardon, 1998] used a fuzzy logic based multi-objective algorithm to optimize micromechanical model parameters of copper powder. This algorithm incorporates all objectives into one using a fuzzy rule, but works with a population and selection based on the Pareto optimality concept.

The multi-objective parameter estimation of microbiological and biological models is presented in [Wang and Sheu, 2000] and in [Liu and Wang, 2008], respectively. In both papers, the MOP problem is solved as a min-max problem by the differential evolution.

A new schema, so called multi-objective evolutionary annealing-simplex, is presented in [Efstratiadis and Koutsoyiannis, 2008], where the method is able to deal with the calibration of complex hydrological models. Their model has 18 parameters to estimate and 7 objectives to be optimized. Wohling et al. [Wöhling et al., 2008] compare three multi-objective optimization algorithms for inverse modelling of a vadose zone. Last but not least, Shi et al. [Shi et al., 2006] used NSGA II for dynamic differential equations parameters estimation.

Chapter 6

AFFINITY MODEL OF CEMENT PASTE HYDRATION

The first presented problem is the parameter estimation for the affinity model of cement paste hydration. Modelling cement paste is very important in civil engineering, because concrete inherits the majority of its properties from the cement paste. Also, the final reliability and durability of construction strongly depends on the heat released during the hydration process of cement paste. The modelling of concrete hydration represents a challenging task especially due to its multi-scale nature and the missing mathematical formulation of several underlying phenomena. The missing description can be replaced by a cellular automata model [Šmilauer, 2006]. Recently, a combination of the CEMHYD3D model [Bentz, 2005] with homogenization processes was employed as a basis for the optimization of the cement paste composition [Šmilauer et al., 2008].

The CEMHYD3D model requires inputs which are not easily accessible in engineering practice, such as the proportion of clinker minerals or the gypsum volume (other parameters are the water to cement ratio and fineness) and the computation is time demanding. This excludes the CEMHYD3D model from its common use for the prediction of hydration heat. On the other hand, a phenomenological model derived from isothermal calorimetry ([Šmilauer, 2010]) has only four parameters and its time demands are very low. This model is called *affinity model* and is based on the product of the potential heat Q_{pot} given by the chemical composition and the curve of the degree of hydration (DoH).

The affinity model provides a simple framework describing all stages of cement hydration. The rate of hydration can be expressed by the temperature-independent *normalized chemical* *affinity* $\tilde{A}(\alpha)$ ([Gawin et al., 2006])

$$\frac{\mathrm{d}\alpha}{\mathrm{d}t} = \tilde{A}(\alpha) \exp\left(-\frac{E_a}{RT}\right), \tag{6.1}$$

where α stands for DoH, T is an arbitrary constant temperature of hydration, R is the universal gas constant (8.314 Jmol^{-1}K^{-1}) and E_a is the apparent activation energy.

For the hydration heat prediction, an analytical form presented in [Šmilauer, 2010] is used:

$$\tilde{A}(\alpha) = B_1 \left(\frac{B_2}{\alpha_\infty} + \alpha\right) (\alpha_\infty - \alpha) \exp\left(-\bar{\eta}\frac{\alpha}{\alpha_\infty}\right), \tag{6.2}$$

where $B1, B2$ are coefficients related to chemical composition, α_∞ is the ultimate hydration degree and $\bar{\eta}$ represents microdiffusion of free water through formed hydrates. Then, a curve of the DoH development can be obtained by t numerical integration of Equation 6.2.

The task of the parameter estimation was to estimate parameters $B1, B2$ and $\bar{\eta}$ for measured hydration curves. Instead of experimental curves, the outputs from the CEMHYD3D model were used since this model had already been verified to be capable of the hydration heat prediction with a good agreement with experiments. A potential hydration heat Q_{pot} can be obtained from the Portland cement mineral composition, however, for presented calculations Q_{pot} was set at 500 J/g. The value of α_∞ is derived from the water to cement ratio:

$$\alpha_\infty = \begin{cases} 0.9, & \text{if } \frac{w/c}{0.42} > 0.9, \\ \frac{w/c}{0.42} & \text{otherwise.} \end{cases} \tag{6.3}$$

The probable values of parameters were estimated in limits shown in Table 6.1, but only the initial population was created inside these limits. Then, the searching process was not bounded. Only if a parameter lower than zero was created, its value was set at zero.

The main objective function was defined as a sum of differences between the desired curve and the affinity model simulation in discrete time steps (ERR). The differences between the curves in 2 hours (BEGIN) and 100 days (END) were added as additional objectives, see Figure 6.1 left. For WSM, all objectives were set to have the same importance. The influence of adding more objectives was studied. Therefore, the results for only two objectives (ERR

	min	max
$B1$	$1.0 \cdot 10^6$	$1.0 \cdot 10^8$
$B2$	$1.0 \cdot 10^{-5}$	$1.0 \cdot 10^{-3}$
$\bar{\eta}$	6	9

Table 6.1: Limits for affinity model parameters.

and BEGIN) optimization and for the case of all three objectives are presented further. All errors were introduced in a square form. In order to obtain some idea about the importance of the proposed objectives for individual parameters, the sensitivity analysis was performed. 100 samples were generated in limits presented above using Latin Hypercube Sampling, hydration curves were created by the affinity model and errors computed. Then, the Pearson correlation coefficient was calculated for all errors and parameters:

$$cor = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}, \tag{6.4}$$

where $\bar{x}$ and $\bar{y}$ are the means of $\mathbf{x}$ (parameters) and $\mathbf{y}$ (errors). A higher absolute value of the correlation coefficient indicates a stronger relation between a particular parameter and an error, which means a higher probability of success in the estimation of that parameter, see Figure 6.1 (right) for resulting coefficients. From this, $B1$ seems to be the most important parameter.

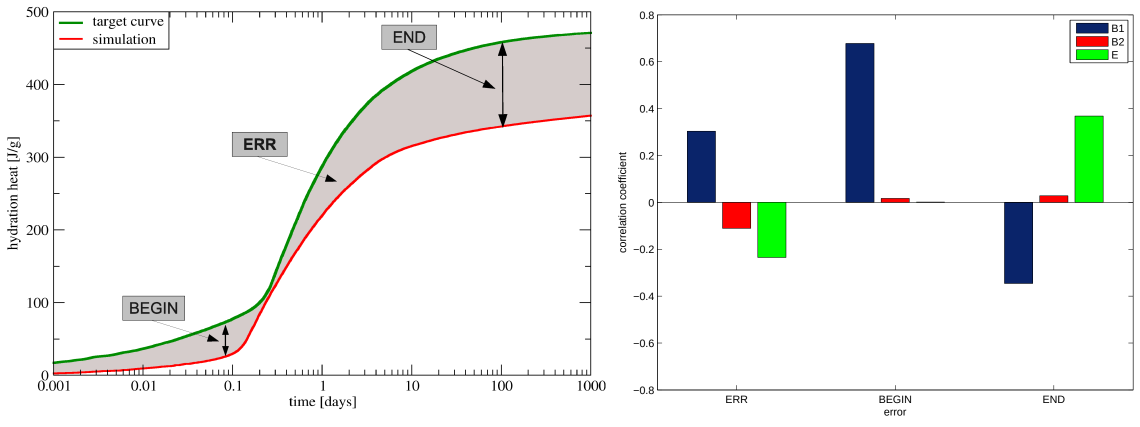

Figure 6.1: Proposed objectives (left) and their correlation with individual parameters (right).

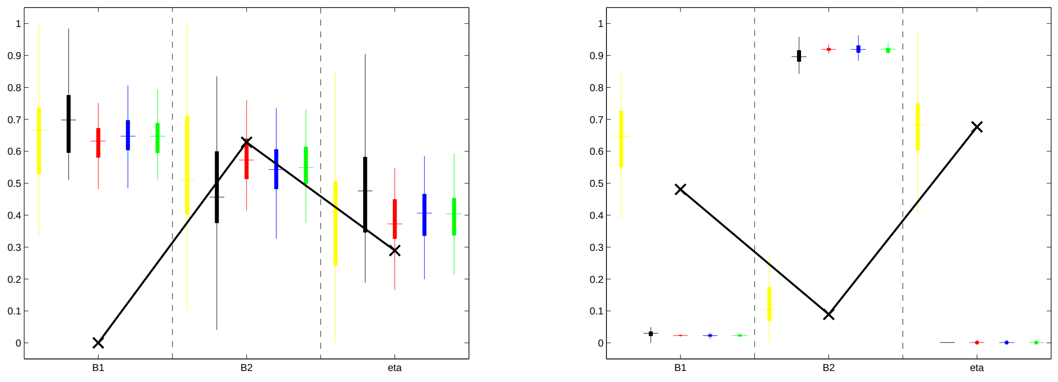

Figure 6.2: Normalized parameters values of two randomly selected curves. Black line: desired values. Yellow: GRADE, black: AR, red: NSGA-II, blue: SPEA2, green: WSM.

6.1 Verification

Following the methodology from the previous chapter, the ability of the proposed algorithms to re-find the known parameters was tested first. Five curves created by the affinity model were chosen and their parameters searched. The population size was 20; each algorithm was started 50 times.

Figure 6.2 presents the estimated parameters' values in the boxplot form and the desired values as a line normalized on the interval (0, 1) for two curves. In the first case, all algorithms found almost the same (wrong) solution. In the second case, GRADE was more successful than all multi-objective algorithms, see also Figure 6.3 for curves generated with randomly chosen final parameters. Here, the difference between the two and three objectives version is almost unnoticeable, therefore, only one result for each algorithm is presented.

As the time demands of GRADE have not been compared with MOEAs yet, this comparison is presented here. Average times needed for the presented task are stated in Table 6.2. The time needed for GRADE is the second highest after SPEA, however, all algorithms are very fast, therefore, taking time demands into account while choosing the algorithm makes sense only if the model simulation is very fast as well, let us say up to 30 seconds.

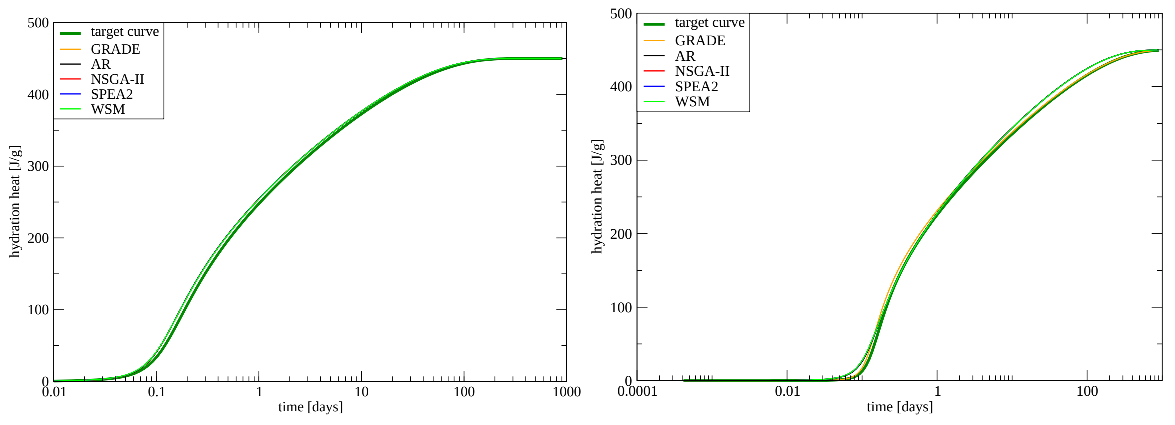

Figure 6.3: Hydration heat curves for verification. Note that in the left figure all resulting curves are almost identical and in the right figure all MOEAs' resulting curves are identical.

	GRADE	AR	NSGA-II	SPEA2	WSM
time [s]	00.94	00.75	00.72	01.00	00.63

Table 6.2: Comparison of execution time for affinity model estimation.

6.2 Validation

In the next step, five curves created by the CEMHYD3D model were chosen as "experimental" data which are to be approximated by the affinity model, see Figure 6.4. The setting was the same as in the previous section, i.e. $Pop = 20$, $generation\_limit = 1000$. Performance according to RWS is shown in Figure 6.5. Comparison was based on the lowest ERR values taken from the obtained Pareto fronts. Three extreme cases are studied in detail further: C1, where WSM outperforms the other methods in all generations; C3, where GRADE is the winner and C5, where GRADE, NSGA-II and WSM seem to have similar success ratio. The influence of an increasing number of objectives is a subject of a study as well. However, the number of objectives does not significantly change the winning score for individual algorithms.

To understand the differences among the mentioned cases, the evolution of average values of the main objective ERR was calculated (see Figure B.1 in Appendix B) and all final

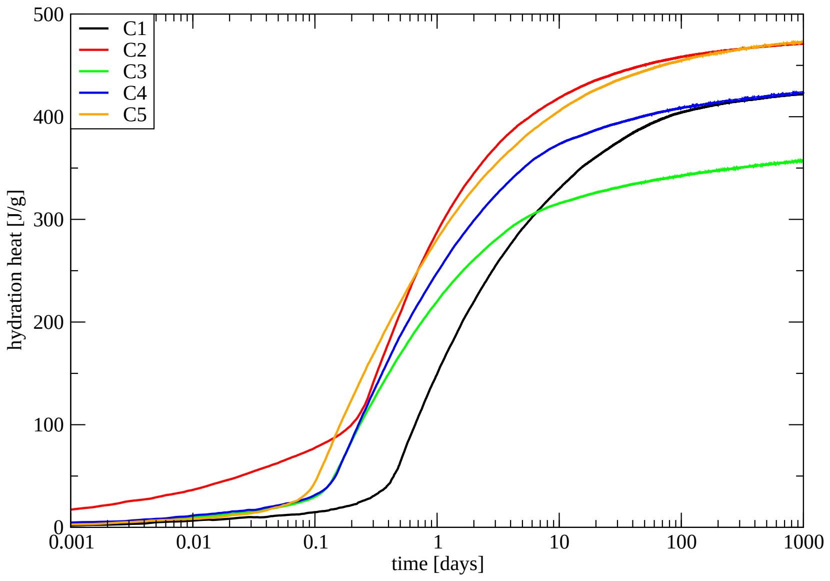

Figure 6.4: Target curves for affinity model estimation.

solutions for the two objectives case are plotted in the objective space, see Figures 6.6 - 6.7. The second objective for GRADE results was also calculated. From these pictures, the main difference between multi-objective, single-objective and multi-objective by WSM parameter estimation is clear. WSM finds only one solution (or very similar solutions for the curve C5), because of constant weight vector. Multi-objective algorithms (namely NSGA-II and SPEA2) are ”distracted” by searching in more dimensions, which can lead to finding the same solutions as WSM and GRADE, see Figure 6.6, or even better than GRADE (Figure 6.8). For the curve C3 (Figure 6.7), all multi-objective algorithms are probably trapped in the local minimum (the average value of ERR does not change from the 20th generation), from which GRADE was able to escape thanks to CERAF methodology. (Note that crossover and mutation operators are different for GRADE and MOEAs which can be the cause as well.)

Figure 6.9 shows the best and the worst obtained results for the curve C1, the other curves are presented in Appendix B, Figures B.2 - B.3. Here, the difference between the 100th and the 1000th generation is visible as well as the influence of more objectives. Except the C3 case, the best found curves are almost identical for all algorithms.

The main advantage of multi-objective parameter estimation is presented in Figure 6.10,

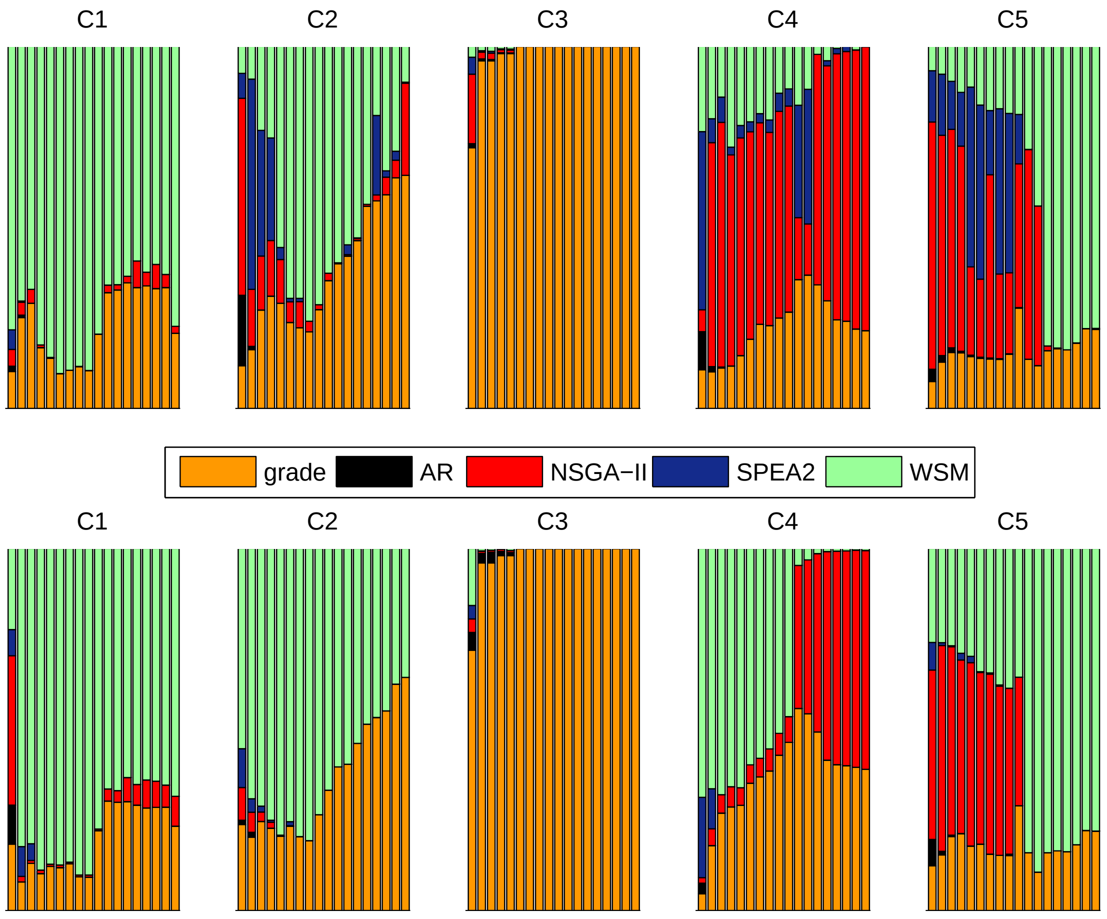

Figure 6.5: Relative winning score for affinity model, upper row: 2 objectives, lower row: 3 objectives.

where all final Pareto solutions from a randomly chosen run of NSGA-II for the three objectives case are plotted and compared with the best curve found by GRADE. Neither GRADE nor NSGA-II found the desired curve, but the NSGA-II results give the user a possibility of choosing a solution suitable for his/her purposes (with the same computational effort as GRADE).

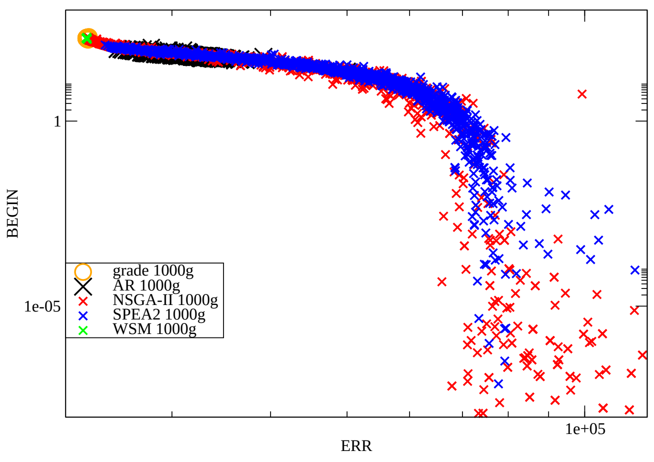

Figure 6.6: Pareto fronts for C1 - original objective vs. error at the beginning.

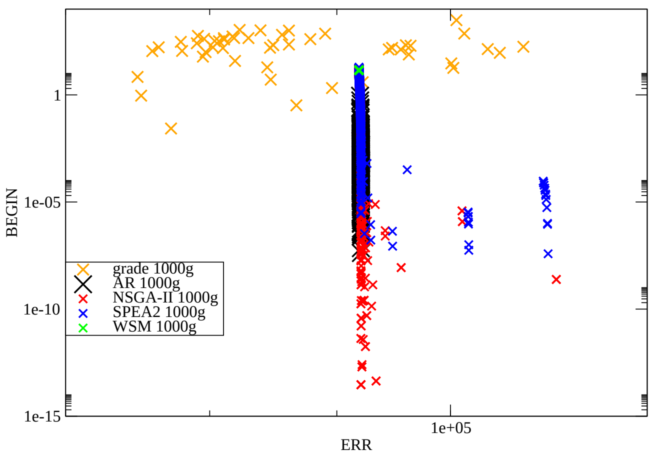

Figure 6.7: Pareto fronts for C3 - original objective vs. error at the beginning.

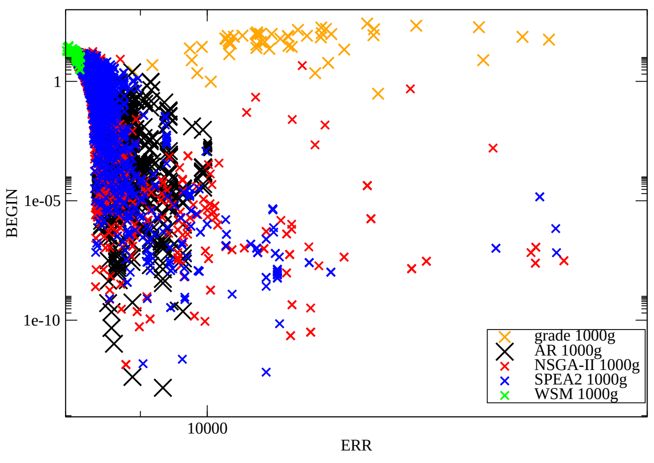

Figure 6.8: Pareto fronts for C5 - original objective vs. error at the beginning.

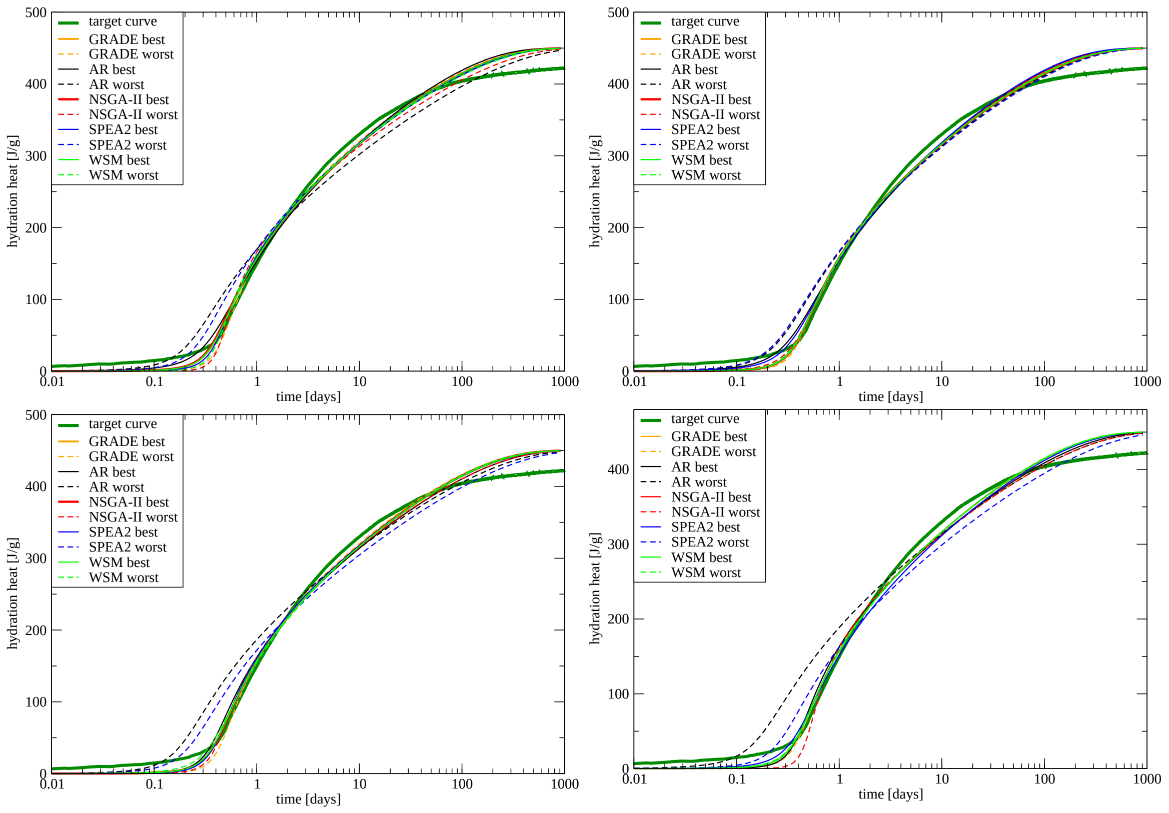

Figure 6.9: Resulting best and worst curves for C1. Upper figures: 2 objectives, lower: 3 objectives. Left: 100th generation, right: 1000th generation.

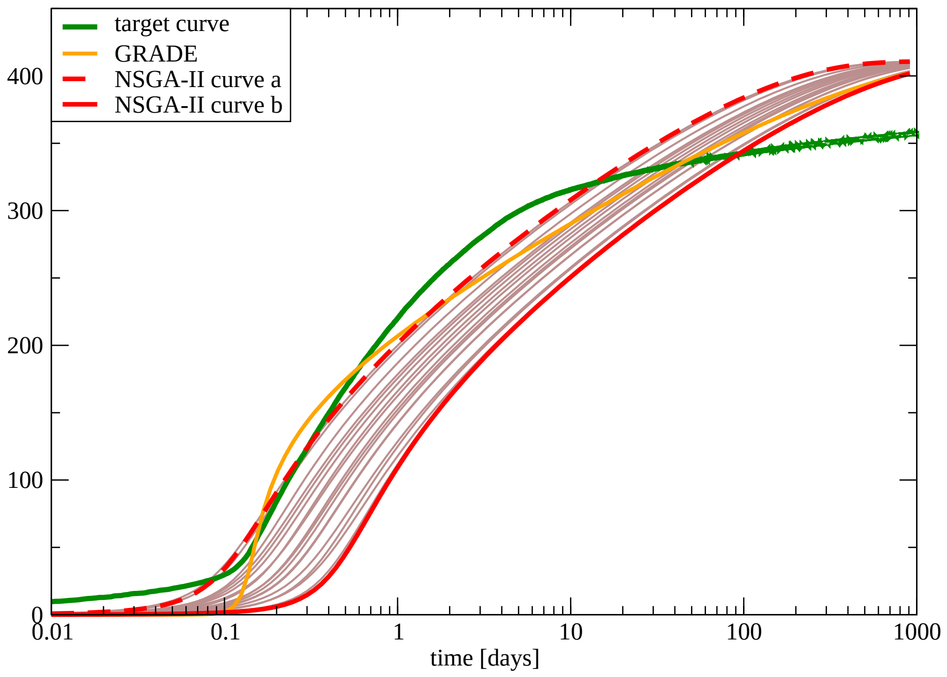

Figure 6.10: Pareto curves for C3.

Chapter 7

DUAL POROSITY MODEL OF RICHARDS' EQUATION

This chapter is inspired by a paper presenting a numerical solution for Richards' equation with a dual porosity conceptual model [Kuraz et al., 2010]. The mathematical and hydrology details are not discussed here, the interested reader is referred to the original paper. The classical Richards' equation describes fluid movement in an unsaturated/saturated zone. The techniques for an accurate solution are available for isotropic material, however, many practical engineering problems deal with a flow in an environment with fractures or fissures, which cannot be accurately homogenized. Therefore, thedual porosity model has been established. The medium is separated into two parts, each of them is homogenized and assigned hydraulic properties. Then, the solution is given by the superposition of these parts over the same volume.

The numerical solver, called *DRUtES*, presented in [Kuraz et al., 2010] is calibrated on the approximation of a one-dimensional vertical infiltration experiment on a rock sample with discrete fractures[1]. The parameters to be calibrated are the volume ratio of the part with fractures and unsaturated hydraulic parameters for this part - such as porosity or saturated hydraulic conductivity; a total of seven parameters, hereafter denoted as $X1 - X7$[2].

The authors presented a multi-objective parameter estimation procedure based on the SADE genetic algorithm (previous version of GRADE [Hrstka and Kučerová, 2004]) with the Average Ranking selection. Therefore, their result (see Table 7.3 or Figure 7.5) will be compared with the tested algorithms.

[1] The model should be used in future to evaluate the contaminant transport from the nuclear waste repository.
[2] In Table 7.1 the ranges for individual parameters are stated.

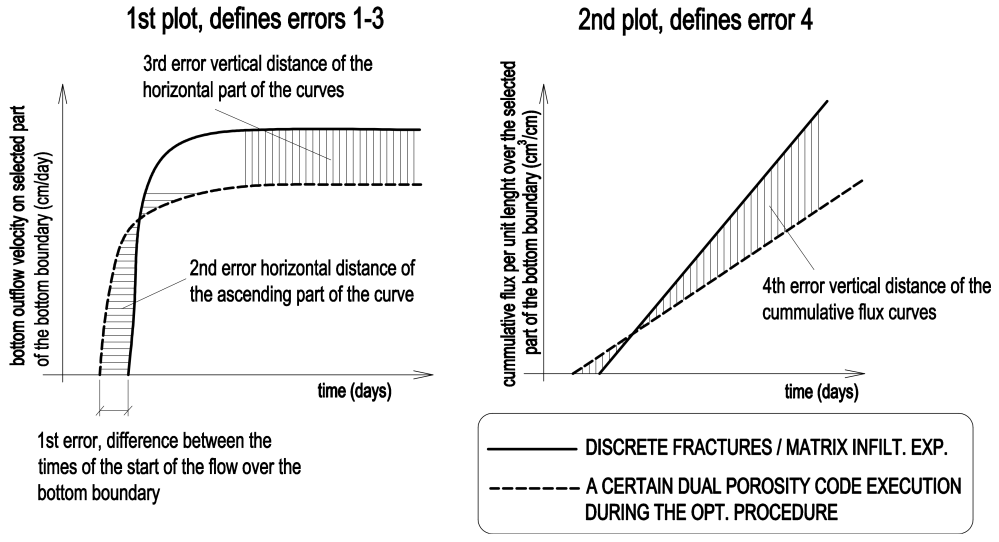
1st plot, defines errors 1-3
3rd error vertical distance of the horizontal part of the curves
bottom outflow velocity on selected part of the bottom boundary (cm/day)
2nd error horizontal distance of the ascending part of the curve
time (days)
1st error, difference between the times of the start of the flow over the bottom boundary
2nd plot, defines error 4
cummulative flux per unit lenght over the selected part of the bottom boundary (cm3/cm)
4th error vertical distance of the cummulative flux curves
time (days)
DISCRETE FRACTURES / MATRIX INFILT. EXP.
A CERTAIN DUAL POROSITY CODE EXECUTION DURING THE OPT. PROCEDURE

Figure 7.1: Objectives for DRUtES model proposed in [Kuraz et al., 2010].

Four objectives were proposed by the authors (see Figure 7.1 for description and Figure 7.2 for their sensitivity analysis): The first objective determines the beginning of the flow through the bottom boundary. It is a squared difference between the start of flows from the experiment and the simulation. The second function is the horizontal RMS error in terms of the ascending part of the graph. The third function is the RMS error of the whole curve. This is important for obtaining macroscopic velocities, which is crucial for contaminant transport modelling. The last objective is the vertical RMS error on the curve of cumulative fluxes (Figure 7.1 (right)), in order to place the emphasis of the optimization algorithm on the mass. For the GRADE algorithm and the comparison presented further, the third error function was chosen as the main one.

7.1 Verification

Again, the ability of an algorithm to re-find known parameters was tested at first. Three previously generated curves were to be found again. The optimization process was stopped after 1000 generations; the population contained 40 individuals and the computation was performed 20 times. Figure 7.3 shows the final boxplots of the found parameters and the desired

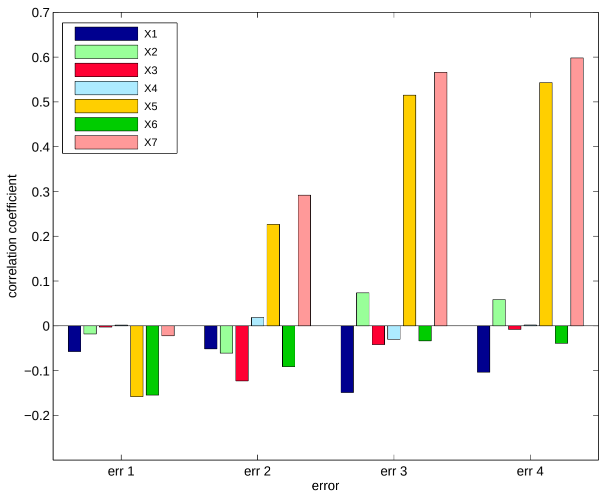

Figure 7.2: Correlation of proposed objectives with individual parameters.

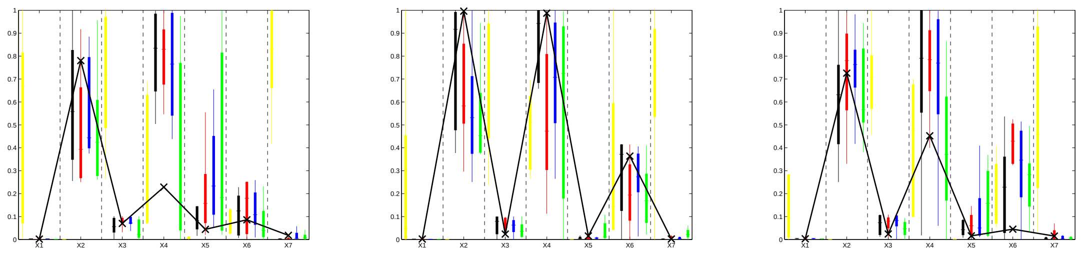

Figure 7.3: Normalized parameters values for three randomly chosen curves. Black line: desired values. Yellow: GRADE, black: AR, red: NSGA-II, blue: SPEA2, green: WSM.

values. Here, the searching process was more successful than for the affinity model from the previous chapter. It can be said that the parameters with high correlation coefficients, i.e. X5 and X7, are identified accurately enough. Moreover, parameters X1 and X3 are identified by multi-objective algorithms and not by GRADE. Although it is not noticeable from sensitivity analysis, these parameters probably affect other parts of the curve than the steady state flux.

7.2 Validation

As this problem has a relatively high number of objectives and variables as well, the difference between population sizes 20 and 40 was studied. Figure 7.4 shows the perfor-

parameter	min	max
$X1$	10^{-2}	10^{-3}
$X2$	6.0	1.5
$X3$	0.7	0.1
$X4$	1.0	10^{-4}
$X5$	$2.785 \cdot 10^{-4}$	1.0
$X6$	10.0	0.0
$X7$	0.75	10^{-4}

Table 7.1: Limits for DRUtES model parameters.

mance of algorithms regarding RWS. This is first case of already presented results, where AR has some remarkable success, and, it is also the first case where WSM lost completely. Note that the RWS method takes into account only winners, therefore the average, best and worst values of the main objective after 1000 generations are stated in Table 7.2. From these measurements, the performance of NSGA-II seems to be comparable with GRADE.

algorithm	mean	min	max
GRADE	0.0792	0.0111	0.1576
AR	0.1919	0.0289	0.3017
NSGA-II	0.0669	0.0326	0.1496
SPEA2	0.2763	0.1580	0.3440
WSM	0.3717	0.2899	0.6180

Table 7.2: Comparison of the main objective values after 1000 generations.

Next, the curves for the best and worst found parameters are plotted in Figure 7.5. The comparison with the curve identified in [Kuraz et al., 2010] is also provided. From the two types of curves presented above as a target, only the flux over the bottom boundary is shown, because it is more interesting. The algorithms for the population size of 40 succeeded in identifying the most important part of the curve, i.e. the maximal flux. Besides that, the

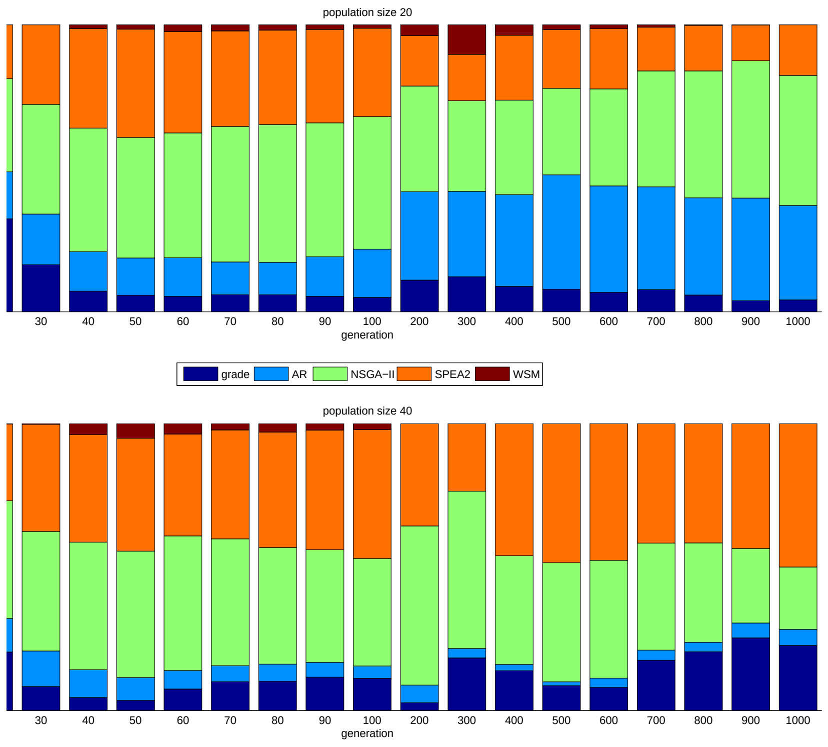

Figure 7.4: RWS for DRUtES model and two population sizes.

graphs offer some interesting ideas:

- The problem is probably multi-modal (see e.g. the worst curves in Figure 7.5 A or B) and numerically unstable (see the worst curves in Figure 7.5 C or D).
- The population size of 40 performs significantly better, the results for this population size are better in the 100th generation than those for the population size of 20 in the 1000th generation. This confirms the general recommendation to set the population size according to the dimension of the problem.
- Only GRADE was able to find a solution almost identical with the desired one not only in the steady state part but also in the steeply ascending part (see Figure 7.5 D), although it did not have special "information" about this part like the multi-objective

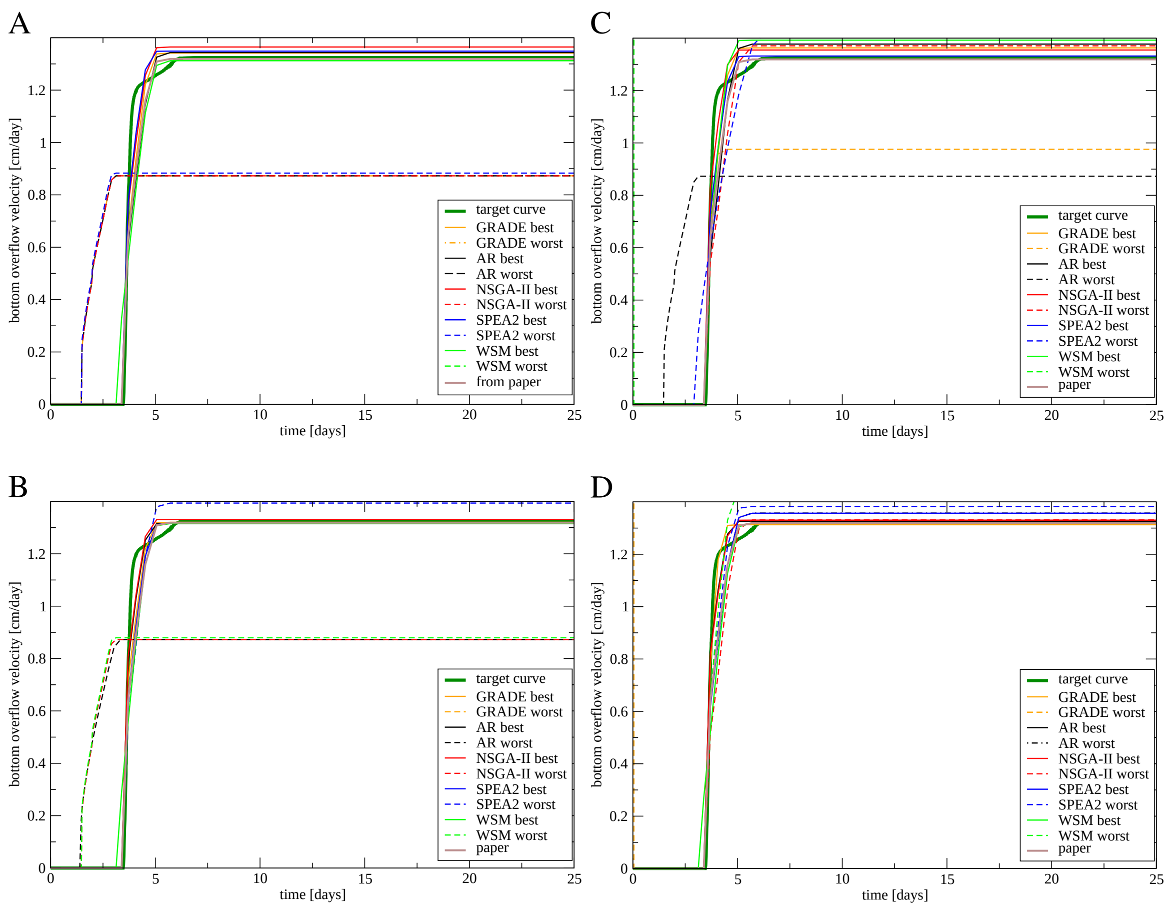

Figure 7.5: Resulting best and worst curves for DRUtES model compared with the experimental curve and the curve found in the paper. Upper figures: population size 20, lower: population size 40. Left: 100th generation, right: 1000th generation.

algorithms. On the other hand, reaching an equivalently good solution is not reliable, see the worst curve in the same graph.

- The best fit found in the cited paper is comparable with the presented results in the maximal flux, but lost in the ascending part of the curve.

As the last comparison, the parameters identified in [Kuraz et al., 2010] and found parameters are stated in Table 7.3.

	$X1$	$X2$	$X3$	$X4$	$X5$	$X6$	$X7$
Kuraz	**0.0068**	**5.4829**	**0.4337**	**0.9109**	**15.9464**	**8.8991**	**0.0293**
population size 20							
GRADE	0.0100	5.9993	0.51553	0.2410	9.2703	0.8186	0.0578
AR	0.0100	5.4735	0.69735	0.6887	26.650	9.9913	0.0194
NSGA-II	0.0067	5.8514	0.6674	0.9595	24.2304	2.7508	0.02043
SPEA2	0.0068	5.4613	0.5931	0.6799	15.632	9.9418	0.0308
WSM	0.0080	5.3217	0.3424	0.1350	3.4712	1.2399	0.1974
population size 40							
GRADE	0.0076	4.5484	0.4642	0.0462	1.6549	0.0080	0.5510
AR	0.0063	5.8838	0.6994	0.4144	9.5833	2.9022	0.05143
NSGA-II	0.0061	5.9741	0.6980	0.9832	20.5257	5.5109	0.02305
SPEA2	0.0083	5.4116	0.2874	0.7026	13.6379	9.2199	0.0375
WSM	0.0091	5.7455	0.6420	0.9999	33.6089	8.5364	0.0135

Table 7.3: Comparison among parameters identified in [Kuraz et al., 2010] and resulting parameters in 1000th generation.

Chapter 8

CEMENT PASTE NANOINDENTATION

An experimental method called nanoindentation allows testing physical properties of heterogeneous materials in the scale of individual components [Němeček et al., 2006]. It is based on loading the testing material by a very sharp and rigid point, see Figure 8.1. As the loading imposed by the indenter introduces highly heterogeneous stress and strain fields, the extraction of the material parameters from the experiment is far from being straightforward. In particular, closed-form relations are available only for the simplest material models (linear elasticity); a more realistic constitutive description leads to a large-scale computational simulation based on, e.g., the Finite Element Method.

In this case, the physical properties of cement paste are tested. Specimens are characterized by a 30 mm diameter and a 4 mm height. The water to cement ratio (w/c) is equal to 0.5; CEM I 52.5 N Portland cement is used. For indentation, the Berkovich indenter with a pyramidal shape is applied. The loading is cyclic and is exerted by a force acting in a short period of time (only several minutes). The whole experiment consists of five loading and unloading periods with a small constant force period aimed at creep development, see Figure 8.4.

The numerical analysis was implemented using the ADINA software [ADINA, 2005]. The spatial problem could be solved as a planar problem thanks to axisymmetry [Jůn, 2005]. A finite element mesh is decomposed into 1800 isoparametric four-node elements and is refined around the tip. The indenter is ideally rigid and the contact between the indenter and the paste is updated in every iteration. To properly describe the cement paste non-linear behaviour, a combined visco-plastic model was chosen.

The tensor of the total strain is composed of three parts:

$$\epsilon_{ij} = \epsilon_{ij}^{E} + \epsilon_{ij}^{C} + \epsilon_{ij}^{P}, \tag{8.1}$$

where ϵ_{ij}^{E} is the time independent elastic part,
ϵ_{ij}^{C} is the time dependent creep strain and
ϵ_{ij}^{P} is the time independent plastic part.

The effective creep strain is described by the power creep law:

$$\bar{\epsilon}^{C} = a_0\, \sigma^{a_1}\, a_2, \tag{8.2}$$

where a_0, a_1 and a_2 and Young's modulus E and the yield stress σ_y are parameters to be estimated. The limits of all parameters are stated in Table 8.1.

Because of the computationally demanding run of the model (approximately 1.5 hour per simulation of the whole cyclic loading), only the first cycle of the loading was simulated

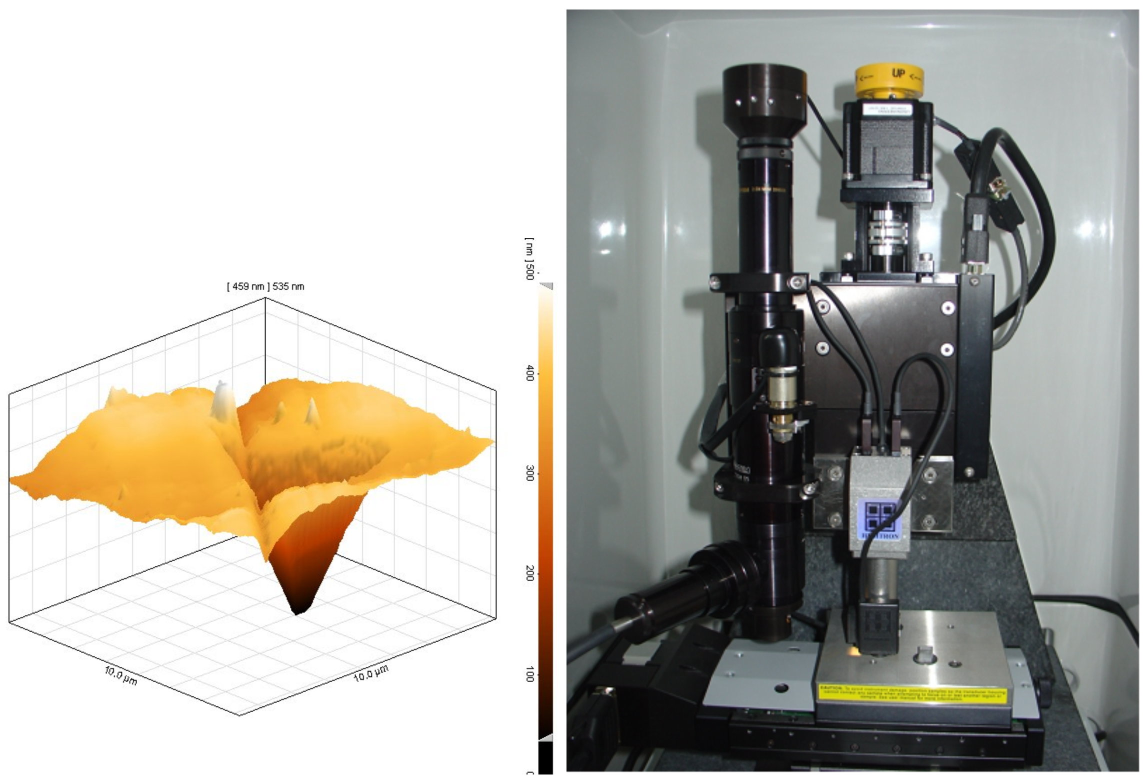

Figure 8.1: Left: Indent from atomic force microscope. Right: Nanoindenter.

Parameters	Units	Minimum	Maximum
E	GPa	15	45
σ_y	MPa	20	600
a_0	-	$1.32 \cdot 10^{-19}$	$1.32 \cdot 10^{-14}$
a_1	-	0.49	2.50
a_2	-	0.05	0.55

Table 8.1: Bounds for nanoindentation model parameters.

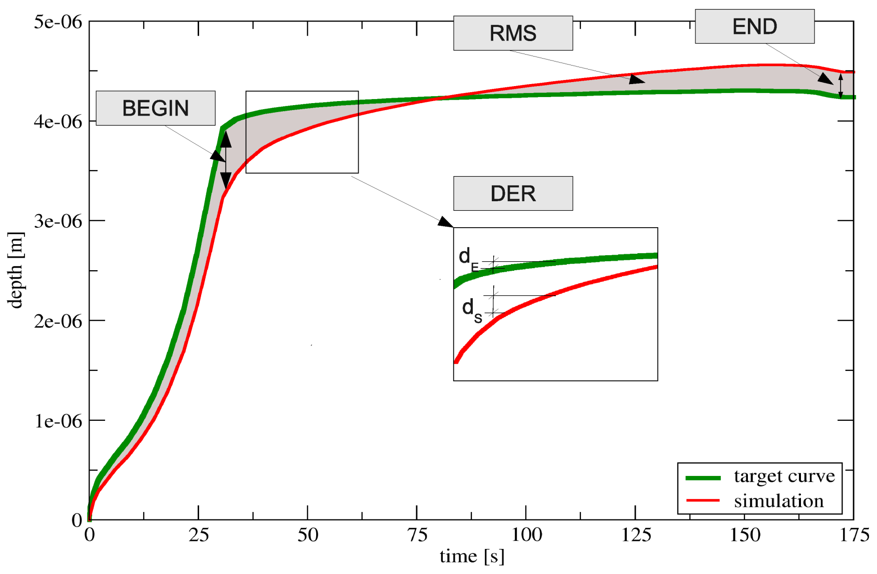

Figure 8.2: Objectives proposed for nanoindentation parameter estimation.

(about 20 minutes for one simulation). It can be assumed that a good fit of the first cycle leads to sufficient accuracy in the other cycles. Moreover, after the best and the fastest algorithm will be found, the accuracy can be increased by adding more cycles.

There are four objective functions, see Figure 8.2. The first and the main one is the RMS error between the desired curve and the simulation. Additional objective functions are: the difference between the "shapes" of two curves by minimizing the errors among the slopes of the given curves, the difference between the curves at the beginning of a constant force part and the difference between the curves at the end of the unloading part.

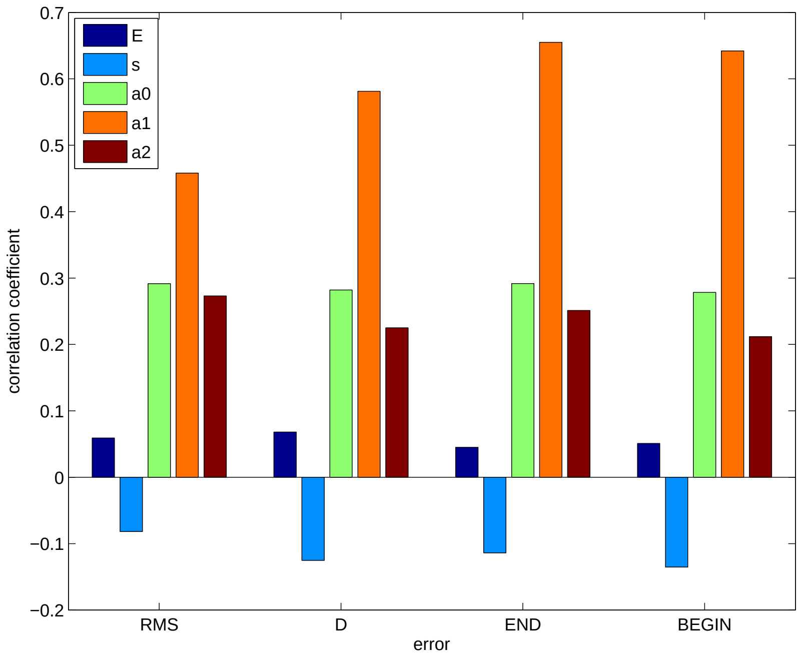

Figure 8.3: Sensitivity of objectives for nanoindentation parameter estimation.

8.1 Verification

The SPEA2 algorithm was not considered in this case anymore, its performance is comparable with NSGA-II in general, but for a low number of generations SPEA2 is outperformed.

As was mentioned above, this model is very time consuming. Therefore, only one optimization run was performed. The population size of 20 was chosen (the number of variables is relatively low) and the optimization process was stopped after 50 generations. Although the number of function evaluations was set so low ($Off * generation\_limit = 500$), the optimization process takes about 5 days for one curve. The weight vector for WSM was set to assign the same importance to all objectives, but, based on previous experience, additional objectives were multiplied by 10^5 to have the order of magnitude about twice smaller than the RMS objective.

For the verification of the nanoindentation model two curves were chosen, curves C1 and

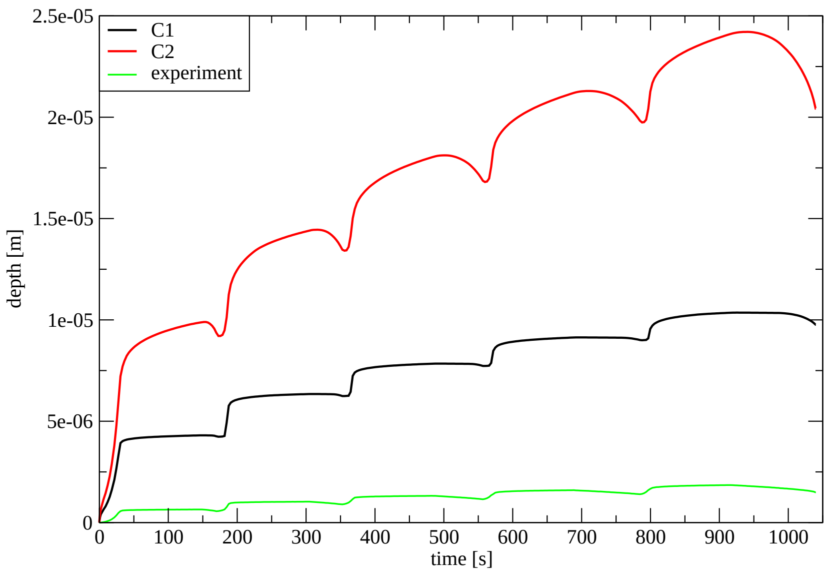

Figure 8.4: Loading curves for C1 and C2 verification and experimental curve.

C2 in Figure 8.4. The found parameters normalized to the (0,1) interval and the resulting curves are shown in Figures 8.6 and 8.5, respectively. In the C1 case, WSM found a significantly better solution than the other algorithms. Moreover, the parameters were found very accurately. For the C2 case, all algorithms found solutions almost identical to the desired one, and in the parameter space, NSGA-II is more successful than the others. However, no conclusion can be made based on these results, because there is only one optimization run.

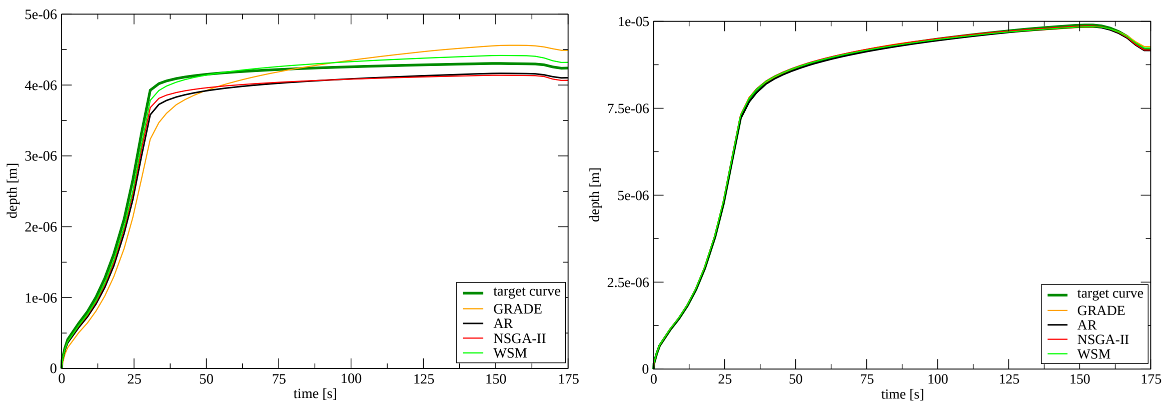

Figure 8.5: Verification on nanoindentation model: time vs. depth curves. Left: C1, right: C2.

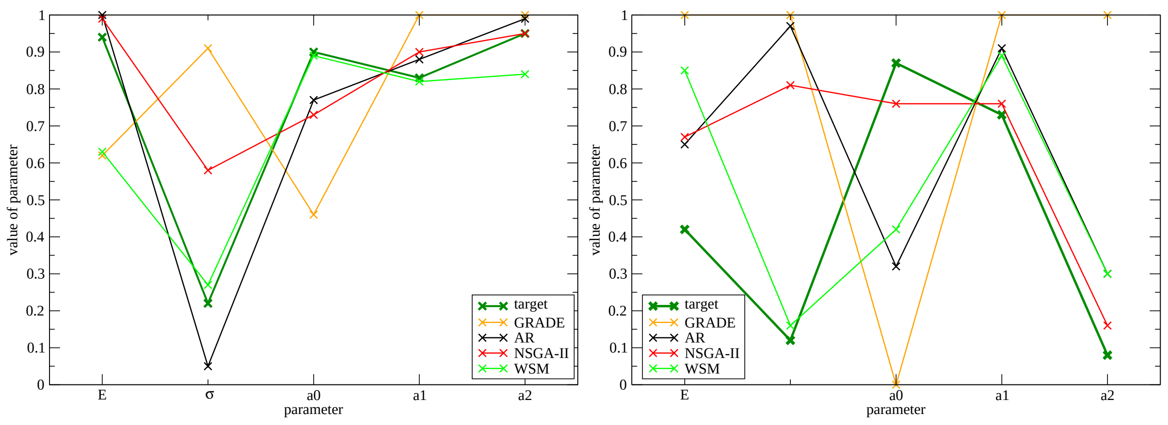

Figure 8.6: Verification on nanoindentation model: parameters in parallel coordinates. Left: C1, right: C2.

8.2 Validation

The experimental curve plotted in Figure 8.4 was to be approximated by the model. The optimizers settings are the same as for the verification, and again, only one optimization run was performed. The curve determined by the resulting parameters is plotted for the first cycle, which was the subject of the estimation procedure, and for the whole cyclic loading as well (Figure 8.7). It is clear from the latter that the idea of fitting only one cycle whose results can be later extended to the other cycles is not correct for the experimental curve. This can be caused by the estimation results as well as by the model itself. However, for the first cycle, the WSM algorithm again gives the best results.

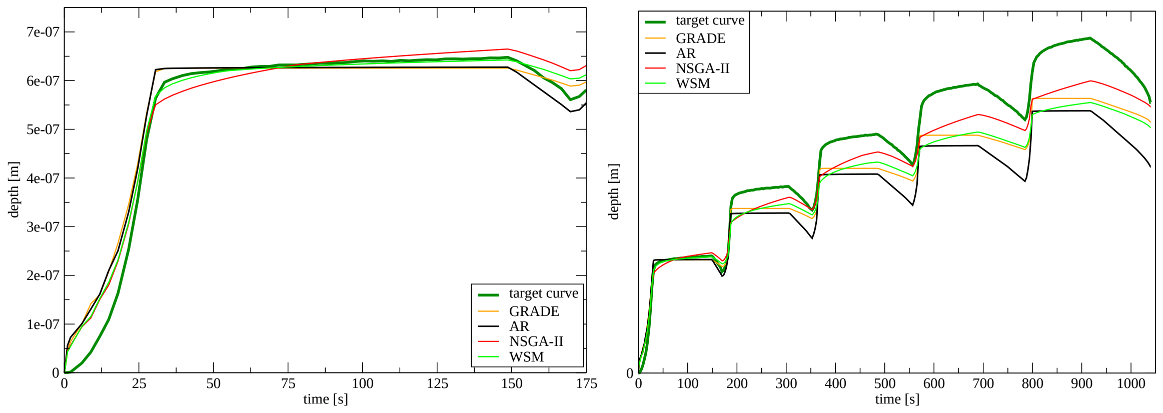

Figure 8.7: Parameter estimation for nanoindentation experiment. Left: first loading cycle, right: all cycles.

Chapter 9

CONCLUSIONS

The presented thesis introduces a relatively new approach to parameter estimation - a multi-objective method. The parameter estimation problem is a process of finding the input model parameters using the response of the model. As the estimation process is supposed to be used repeatedly, not only the accuracy but also the efficiency of the estimation method is essential for its choice. In some aspects, multi-objective optimization seems to fulfil these criteria.

The thesis is divided into two parts. The first one contains the basic notation for parameter estimation together with the introduction of methods. Then, the tested algorithms are described: Chapter 2 presents in detail the GRADE algorithm, as a standard approach to which the proposed methodology is compared. Chapter 3 deals with the multi-objective approach to parameter estimation. At first, basic concepts of multi-objective optimization and a brief review of existing algorithms, with an emphasis on multi-objective evolutionary algorithms are presented. The algorithms tested in this thesis, i.e. Weighted Sum Method, Average Ranking method, Nondominated Sorting Genetic Algorithm II and Strength Pareto Evolutionary Algorithm 2, are introduced. Then, the idea of *multi-objectivization* and *objective helpers*, i.e. the idea of adding more objective functions to the single-objective optimization problem, is presented. Chapter 4 provides a brief introduction to performance assessment for multi-objective problems and results of chosen algorithms in standard multi-objective test problems are presented to show the advantages and disadvantages of individual methods. From this comparison, NSGA-II and WSM algorithms seem to be suitable for multi-objectivized parameter estimation.

In the second part of this thesis, the proposed methodology is tested on three real engineering tasks. Parameter estimation for the affinity model of cement paste hydration is discussed at first. The next problem is the identification of parameters for a numerical solver of the dual porosity model of Richard's equation. As the third example, parameters for the finite element model of cement paste nanoindentation are estimated.

The main goal of this thesis was to decide whether the multi-objectivization of parameter estimation could help. From presented results no unambiguous answer can be given. It may help in some cases and worsen the problem in others. However, determining these cases is a difficult task, part of future work.

From the presented algorithms, NSGA-II and WSM reach results comparable with the classical approach represented here by the GRADE algorithm.

The advantage of NSGA-II is the result in the form of a set of equally good solutions, from which a user can choose the best based on his/her purposes. The time demands are comparable with GRADE, therefore, the only disadvantage is not such an easy implementation. This method can be recommended to be used, when the time available for the estimation process is not long enough to ensure the a good convergence of the standard approach.

On the contrary, WSM provides only one solution with one weight vector setting. The time demands are also comparable with the standard method, and the implementation is very easy in this case. Moreover, the known disadvantage of WSM is the inability to find a solution for nonconvex problems. Another obstacle with WSM is the necessity of setting the weight and scale vector. In the presented examples, all weight vectors had the same values for all objectives. The reason, why WSM failed in some cases, can probably be associated with the scale vector. In my opinion, the WSM method is advantageous when the main objective is dominant and the additional ones are just supportive, i.e. the searching process is guided mostly by the first objective and the others help only in some small limits. This works when the supportive objectives are by about two orders of magnitude smaller than the main one. This was fulfilled in the cases where WSM succeeded, i.e. the parameter estimation for the nanoindentation model and curves C1 and C5 for the affinity model parameter estimation. In other cases, one or more of the additional objectives were higher than this limit.

Of course, without a good knowledge aof the model, this setting cannot be made. If this information is not known, a few iterations with the traditional single-objective method can be performed in advance. These iterations can be used not only to determine the order of magnitude of additional objectives, but also the problematic parts for identifying these objectives.

BIBLIOGRAPHY

ADINA (2005). Adina - theory and modeling guide, Volume 1: Adina solids and structures.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm. In Grefenstette, J., editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 101–111. Lawrence Erlbaum Associates.

Bentley, P. J. and Wakefield, J. P. (1998). Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. In Chawdhry, P., Roy, R., and Pant, R., editors, *Soft Computing in Engineering Design and Manufacturing*, pages 231–240. Springer-Verlag.

Bentz, D. P. (2005). Cemhyd3d: A three-dimensional cement hydration and microstructure development modeling package. Technical report, Building and Fire Research Laboratory Gaithersburg.

Brockhoff, D., Friedrich, T., Hebbinghaus, N., Klein, C., Neumann, F., and Zitzler, E. (2007). Do additional objectives make a problem harder? In *GECCO '07: Proceedings of the 9th annual conference on genetic and evolutionary computation*, pages 765–772, New York, NY, USA. ACM Press.

Coello, C. A. C. (2000). Handling Preferences in Evolutionary Multiobjective Optimization: A Survey. In *2000 Congress on Evolutionary Computation*, pages 30–37, Piscataway, New Jersey. IEEE Service Center.

Coello, C. A. C. (2006). Evolutionary multiobjective optimization: A historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36.

Coello, C. A. C., Lamont, G. B., and Veldhuizen, D. A. V. (2006). *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Coello, C. A. C. and Lechuga, M. S. (2002). MOPSO: a proposal for multiple objective particle swarm optimization. In *CEC ’02: Proceedings of the Evolutionary Computation on 2002. CEC ’02. Proceedings of the 2002 Congress*, pages 1051–1056, Washington, DC, USA. IEEE Computer Society.

Coello, C. A. C. and Pulido, G. T. (2001). A micro-genetic algorithm for multiobjective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO’2001*, pages 126–140. Springer-Verlag.

Collette, Y. and Siarry, P. (2004). *Multiobjective Optimization: Principles and Case Studies*. Decision Engineering. Springer.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Wiley.

Deb, K. (2008). *Introduction to Evolutionary Multiobjective Optimization*, pages 59–96. Lecture Notes in Computer Science. Springer Berlin / Heidelberg.

Deb, K. and Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. Technical report, Department of Mechanical Enginering, Indian Institute of Technology, Kanpur, India.

Deb, K., Agrawal, S., Pratab, A., and Meyarivan, T. (2000). A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In et al., M. S., editor, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France. Springer. Lecture Notes in Computer Science No. 1917.

Deb, K. and Goyal, M. (1996). A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, 26:30–45.

Efstratiadis, A. and Koutsoyiannis, D. (2008). *Fitting Hydrological Models on Multiple Responses Using the Multiobjective Evolutionary Annealing-Simplex Approach*, volume 68 of *Water Science and Technology Library*, pages 259–273. Springer Berlin Heidelberg.

Fairbairn, E. M. R., Ebecken, N. F. F., Paz, C. N. M., and Ulm, F.-J. (2000). Determination of probabilistic parameters of concrete: solving the inverse problem by usign artificial neural networks. *Computers & Structures*, 78:497–503.

Fonseca, C. M. and Fleming, P. J. (1993). Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California. University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers.

Gawin, D., Pesavento, F., and Schrefler, B. (2006). Hygro-thermo-chemo-mechanical modelling of concrete at early ages and beyond. Part I: Hydration and hygro-thermal phenomena. *International Journal for Numerical Methods in Engineering*, 67(3):299–331.

Gendy, A. S. and Saleeb, A. F. (2000). Nonlinear material parameter estimation for characterizing hyper elastic large strain models. *Computational Mechanics*, 25(1):66–77.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley.

Greiner, D., Emperador, J. M., Winter, G., and Galván, B. (2007). *Improving Computational Mechanics Optimum Design Using Helper Objectives: An Application in Frame Bar Structures*, pages 575–589. Lecture Notes in Computer Science. Springer Berlin / Heidelberg.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. MIT Press.

Horn, J., Nafpliotis, N., and Goldberg, D. E. (1994). A niched pareto genetic algorithm for multiobjective optimization. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 82–87 vol.1.

Hrstka, O. and Kučerová, A. (2004). Improvements of real coded genetic algorithms based on differential operators preventing the premature convergence. *Advances in Engineering Software*, 35:237–246.

Ingber, L. (1993). Simulated annealing: Practice versus theory. *Mathematical and Computer Modelling*, 18(11).

Jaszkiewicz, A. and Branke, J. (2008). *Interactive Multiobjective Evolutionary Algorithms*, pages 179–193. Lecture Notes in Computer Science. Springer Berlin / Heidelberg.

Jensen, M. T. (2004). Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation. *Journal of Mathematical Modelling and Algorithms*, pages 323–347.

Jůn, P. (2005). Numerická analýza mikromechanických experimentů na cementové pastě. Master's thesis, České vysoké učení technické v Praze, Fakulta stavební, Praha.

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

Knowles, J. D. and Corne, D. W. (2000). Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172.

Knowles, J. D., 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.

Knowles, J. D., Watson, R., and Corne, D. W. (2001). Reducing local optima in single-objective problems by multi-objectivization. In *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 269–283. Springer-Verlag. Lecture Notes in Computer Science No. 1993.

Kuraz, M., Mayer, P., Leps, M., and Trpkosova, D. (2010). An adaptive time discretization of the classical and the dual porosity model of Richards' equation. *Journal of computational and applied mathematics*, 233(12, Sp. Iss. SI):3167–3177. 2nd International Conference on Finite Element Methods in Engineering and Science, Tahoe City, CA, JAN 05-09, 2009.

Kučerová, A. (2007). *Identification of nonlinear mechanical model parameters based on softcomputing methods*. PhD thesis, Ecole Normale Supérieure de Cachan, Laboratoire de Mécanique et Technologie.

Lehký, D. and Novák, D. (2005). Probabilistic inverse analysis: Random material parameters of reinforced concrete frame. In *Ninth International Conference on Engineering Applications of Neural Networks, EAAN2005, Lille, France*, pages 147–154.

Lepš, M. (2007). Multi-Objective Identification of Material Parameters. In *Engineering Mechanics 2007*, pages 155–156, Prague. Institute of Thermomechanics, AS CR, v.v.i.

Lepš, M. (2008). *Soft-Computing Methods in Material Parameters Identification: State of the Art and Open Issues*, pages 317–322. CIMNE, Barcelona.

Liu, P.-K. and Wang, F.-S. (2008). Inverse problems of biological systems using multi-objective optimization. *Journal of the Chinese Institute of Chemical Engineers*, 39(5):399–406.

Mahfoud, S. W. (1995). *Niching methods for genetic algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, USA.

Mahnken, R. (2004). *Identification of Material Parameters for Constitutive Equations*, chapter 19, pages 637–656. Encyclopedia of Computational Mechanics Part 2. Solids and Structures. John Wiley & Sons, Ltd.

Mahnken, R. and Stein, E. (1996). Parameter identification for viscoplastic models based on analytical derivatives of a least-squares functional and stability investigations. *International Journal of Plasticity*, 12(4):451–479.

Maier, G., Bocciarelli, M., Bolzon, G., and Fedele, R. (2006). Inverse analyses in fracture mechanics. *International Journal of Fracture*, 138:47–73.

Mariano, C. E. and Morales, E. (1999). A multiple objective ant-q algorithm for the design of water distribution irrigation networks. Technical report, Instituto Mexicano de Tecnologia del Agua.

Mertens, J., Stenger, R., and Barkle, G. F. (2006). Multiobjective inverse modeling for soil parameter estimation and model verification. *Vadose Zone J.*, 5(3):917–933.

Mezura-Montes, E. and Coello, C. A. C. (2006). A survey of constraint-handling techniques based on evolutionary multiobjective optimization. Technical report, Evolutionary Computation Group at CINVESTAV, Departamento de Computacion, CINVESTAV-IPN, Mexico.

Michalewicz, Z. (1999). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 3rd edition.

Miettinen, K. (1999). *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Dordrecht.

Nosek, J. and Lepš, M. (2009). A Practical Methodology for Comparison of Evolutionary Optimization Algorithms. In *Proceedings of the First International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering*, pages 1–13, Stirling. Civil-Comp Press Ltd.

Němeček, J., Kabele, P., and Jůn, P. (2006). Effect of creep in evaluation of nanoindentation of cement pastes. In *Engineering Mechanics 2006*.

Pareto, V. (1896). *Cours D'Economie Politique*, volume I and II. F. Rouge, Lausanne.

Rachmawati, L. and Srinivasan, D. (2006). Preference incorporation in multi-objective evolutionary algorithms: A survey. In *IEEE Congress on Evolutionary Computation, 2006. CEC 2006.*, pages 962–968.

Reardon, B. J. (1998). Optimization of Micromechanical Densification Modeling Parameters For Copper Powder using a Fuzzy Logic Based Multiobjective Genetic Algorithm. Technical Report LA-UR-98-0419, Los Alamos National Laboratory, Los Alamos, New Mexico.

Rosenberg, R. S. (1967). *Simulation of genetic populations with biochemical properties*. PhD thesis, University of Michigan, Ann Harbor, Michigan.

Schaffer, J. D. (1984). *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University.

Shi, Y., Lu, J., and Zheng, Q. (2006). A new strategy for parameter estimation of dynamic differential equations based on NSGA II. *Simulated Evolution and Learning*, pages 345–352.

Srinivas, N. and Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248.

Storn, R. and Price, K. (1995). Differential Evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, University of Berkeley.

Ulungu, E., Teghem, J., Fortemps, P., and Tuyttens, D. (1999). MOSA method: A tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8:221–236.

Vidal, R. V. V. (1993). *Applied Simulated Annealing*, volume 396 of *Lecture Notes in Economics and Mathematical Systems*. Springer-Verlag.

Šmilauer, V. (2006). *Elastic properties of hydrating cement paste determined from hydration models*. PhD thesis, CTU in Prague.

Šmilauer, V. (To appear, 2010). *Multiscale Modeling of Hydrating Concrete*. Saxe-Coburg Publications, Stirling.

Šmilauer, V., Vitingerová, Z., and Lepš, M. (2008). Micromechanical modelling and optimisation on cement paste performance. In *Proceedings of the Ninth International Conference on Computational Structures Technology*. Stirling: Civil-Comp Press Ltd.

Wang, F.-S. and Sheu, J.-W. (2000). Multiobjective parameter estimation problems of fermentation processes using a high ethanol tolerance yeast. *Chemical Engineering Science*, 55(18):3685–3695.

Wöhling, T., Vrugt, J. A., and Barkle, G. F. (2008). Comparison of three multiobjective optimization algorithms for inverse modeling of vadose zone hydraulic properties. *Soil Sci Soc Am J*, 72(2):305–319.

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

Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland.

Appendix A

TEST PROBLEMS RESULTS

A.1 Effect of populations size

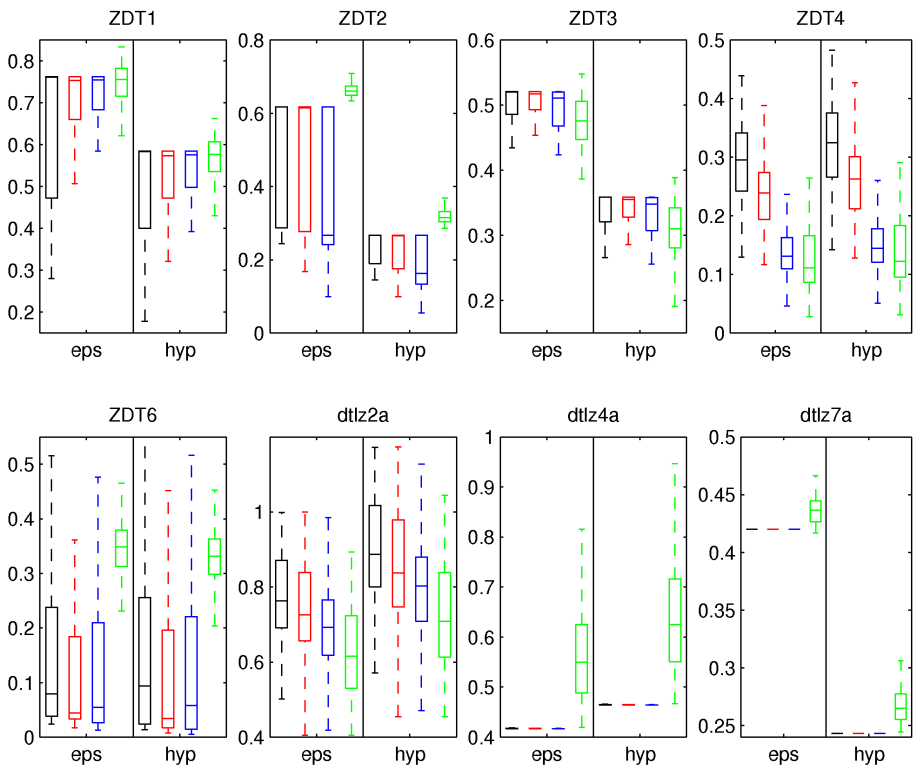

Figure A.1: Influence of population size for AR algorithm - after 100th generation. Population size: black = 12, red = 20, blue = 40, green = 100

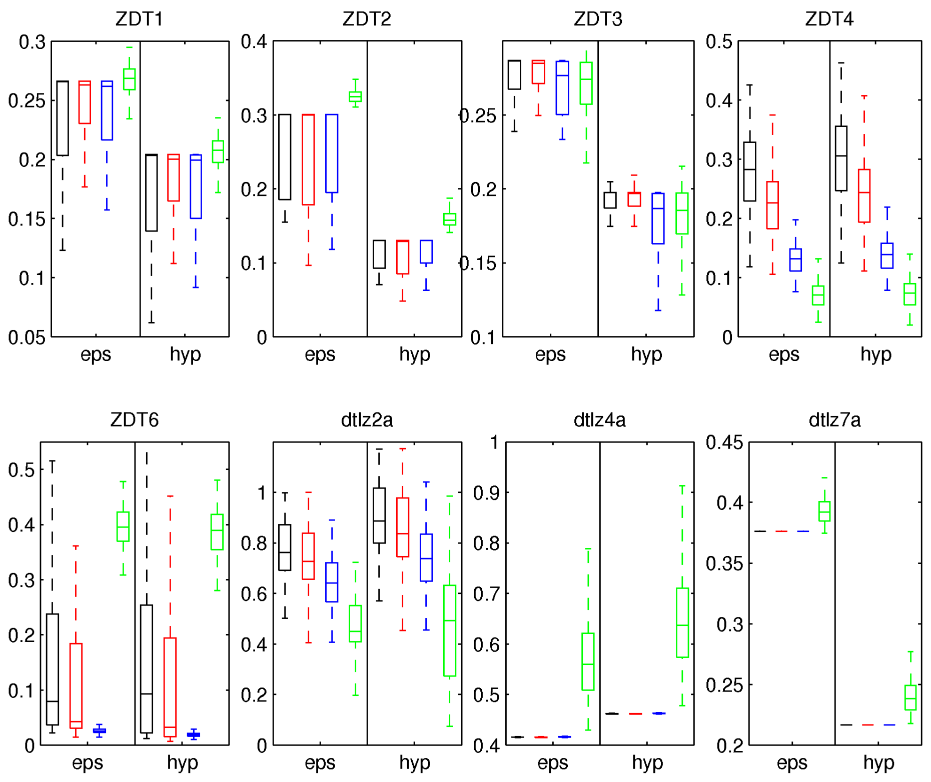

Figure A.2: Influence of population size for AR algorithm - after 1000 function evaluation.

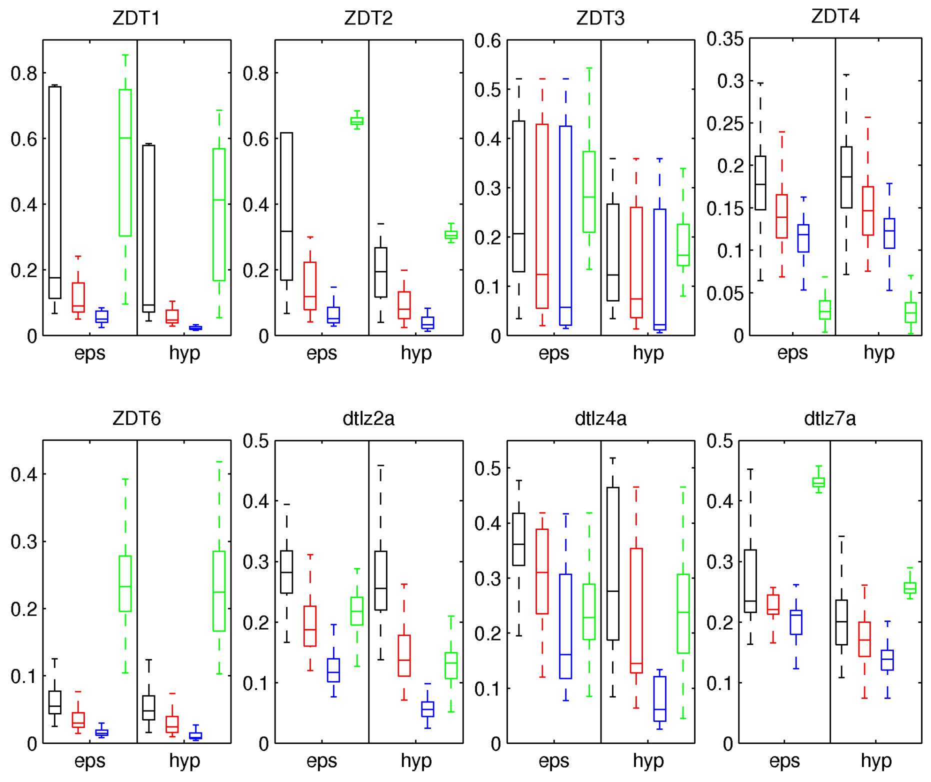

Figure A.3: Influence of population size for SPEA2 - after 100th generation. Population size: black = 12, red = 20, blue = 40, green = 100

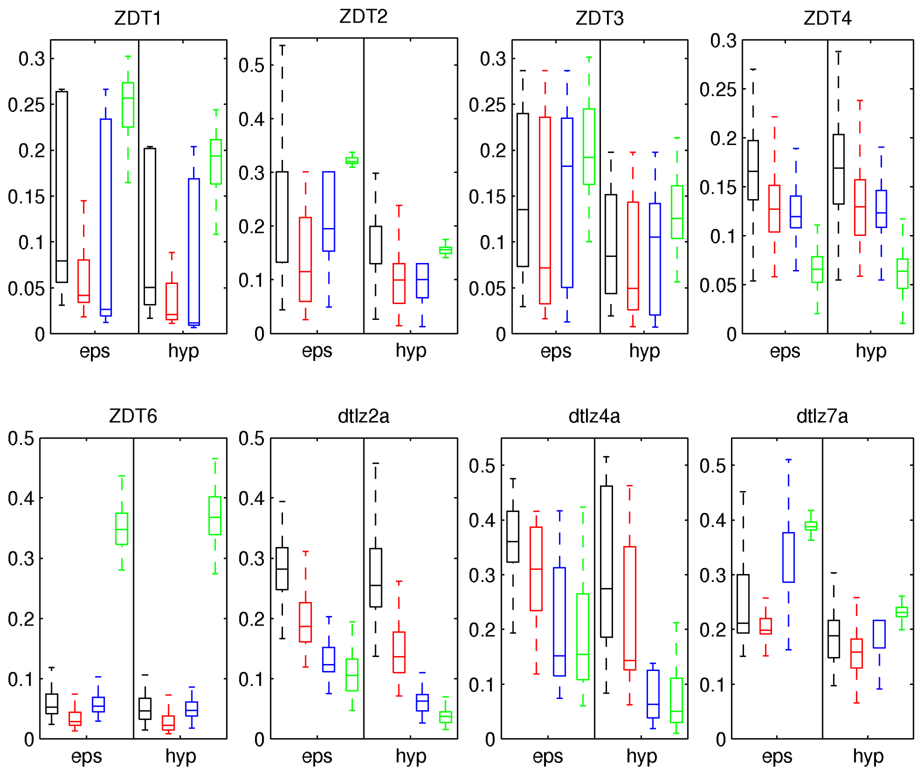

Figure A.4: Influence of population size for SPEA2 - after 1000 function evaluation.

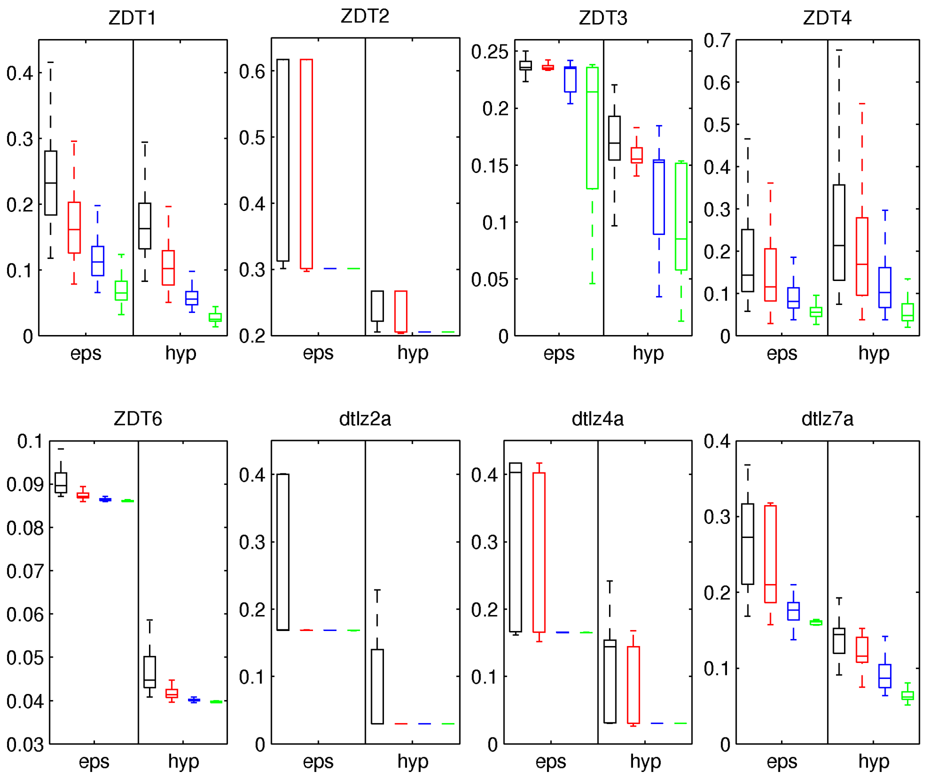

Figure A.5: Influence of population size for WSM algorithm - after 100th generation. Population size: black = 12, red = 20, blue = 40, green = 100

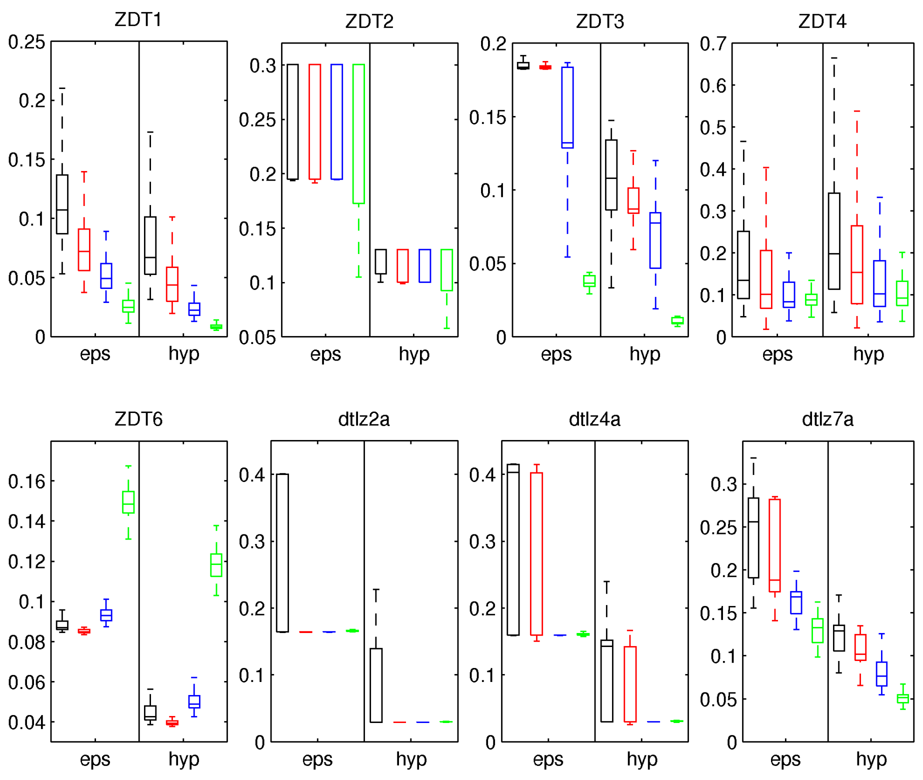

Figure A.6: Influence of population size for WSM algorithm - after 1000 function evaluation. Population size: black = 12, red = 20, blue = 40, green = 100

A.2 Performance of algorithms in time

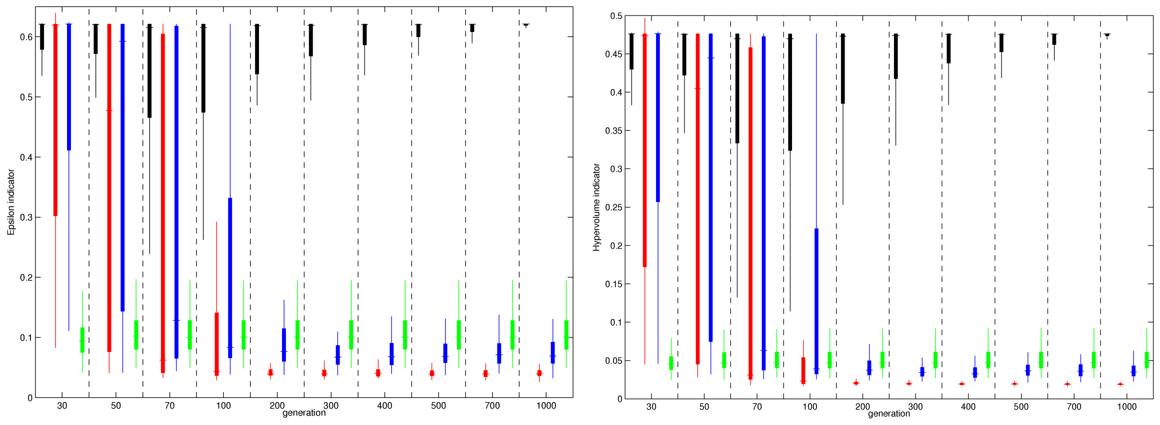

Figure A.7: Evolution of epsilon (left) and hypervolume (right) indicator for ZDT1 function.

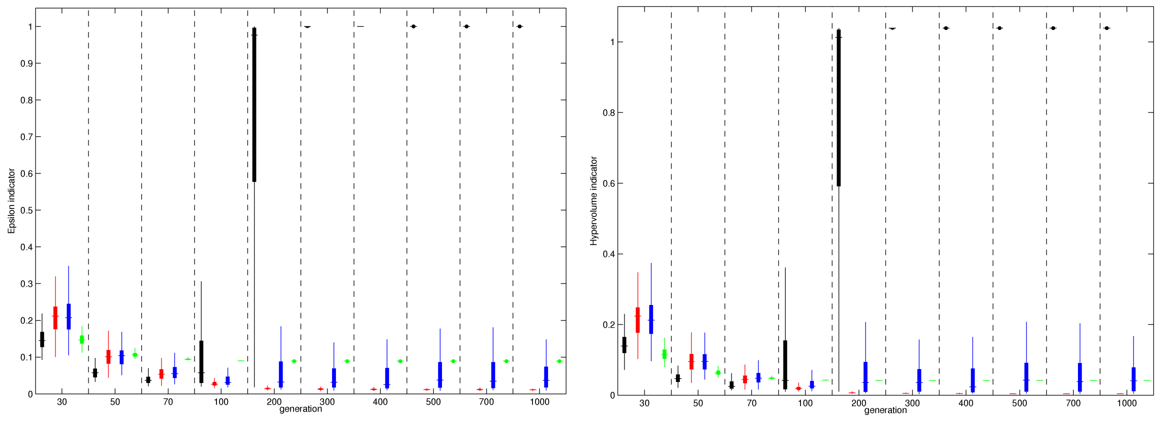

Figure A.8: Evolution of epsilon (left) and hypervolume (right) indicator for ZDT6 function.

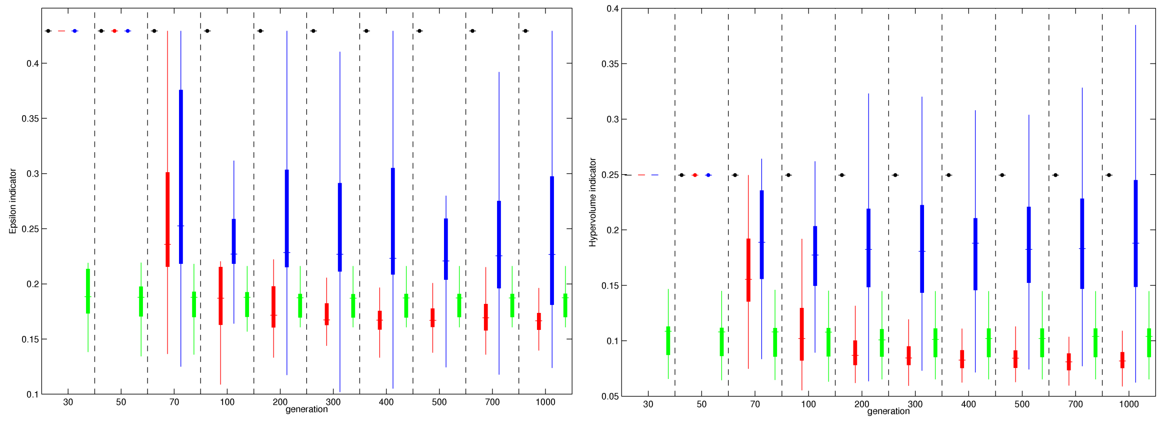

Figure A.9: Evolution of epsilon (left) and hypervolume (right) indicator for dtlz7a function. Black: AR, red: NSGA-II, blue: SPEA2, green: WSM.

Appendix B

AFFINITY MODEL PARAMETERS ESTIMATION

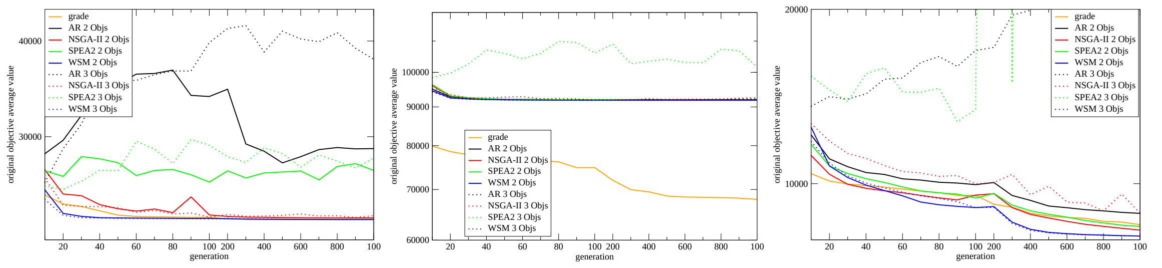

Figure B.1: Average values for the original square error function in time.

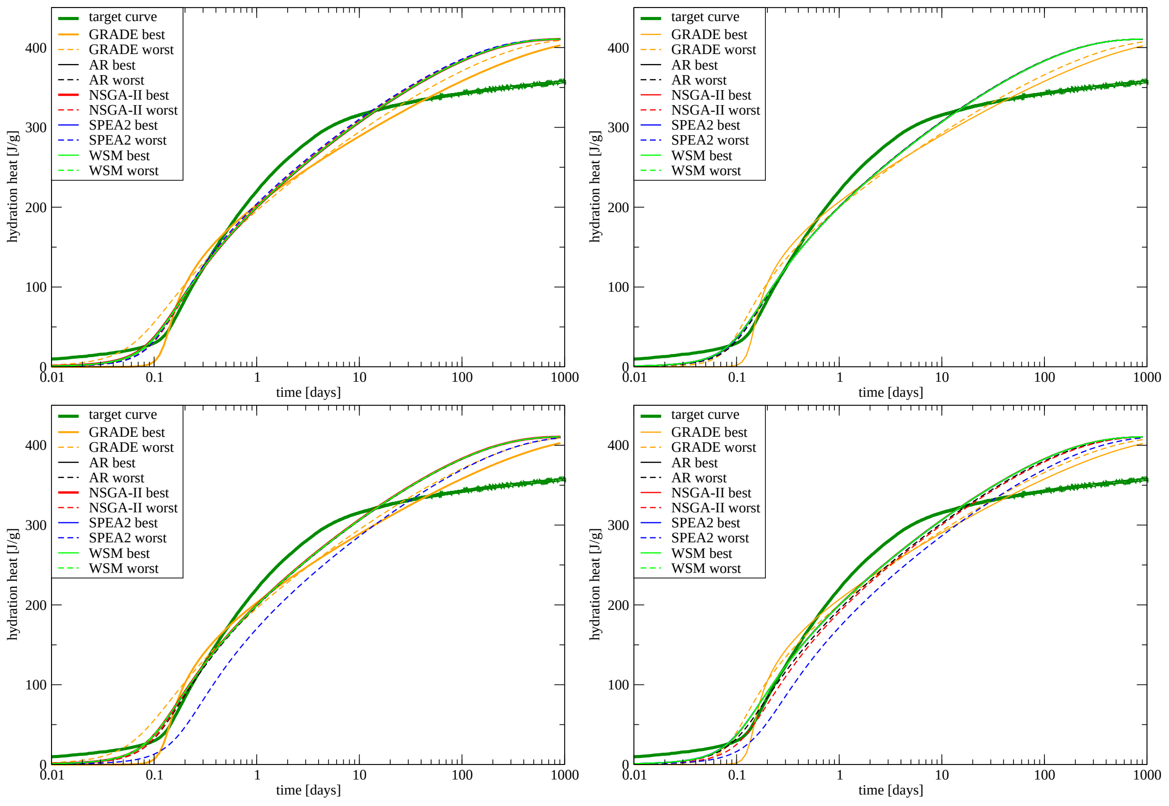

Figure B.2: Resulting best and worst curves for C3.

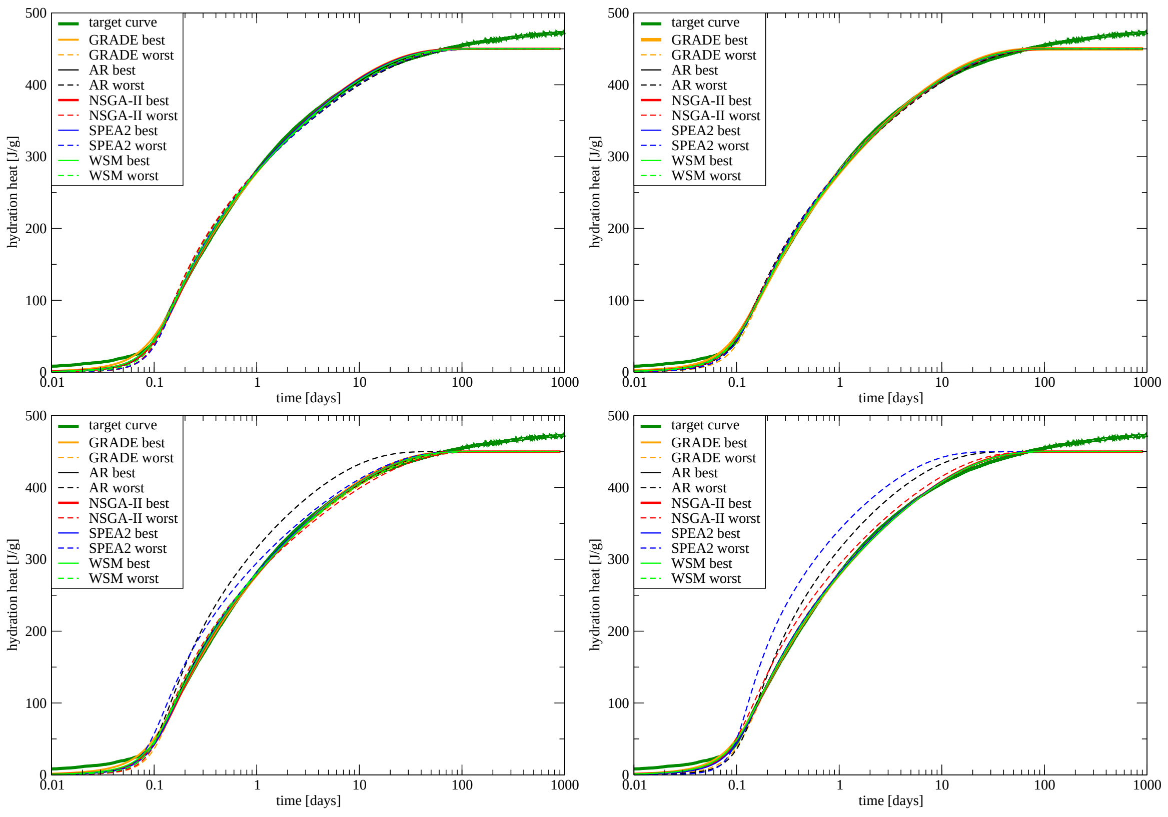

Figure B.3: Resulting best and worst curves for C5.