FACULTY OF **BIOSCIENCE ENGINEERING**

Department of Mathematical Modelling, Statistics and Bioinformatics

Metaheuristic versus tailor-made approaches to optimization problems in the biosciences

ir. Karolien Scheerlinck

Thesis submitted in fulfillment of the requirements for the degree of Doctor (Ph.D.) in Applied Biological Sciences

Academic year 2011-2012

Supervisor:	Prof. dr. Bernard De Baets Department of Mathematical Modelling, Statistics and Bioinformatics Ghent University, Belgium
Examination committee:	Prof. dr. ir. Christian Stevens (Chairman)
	Prof. dr. Jorge Casillas Prof. dr. Hans De Meyer Prof. dr. ir. Veerle Fievez Prof. dr. ir. Sidharta Gautama Prof. dr. ir. Niko Verhoest
Dean:	Prof. dr. ir. Guido Van Huylenbroeck
Rector:	Prof. dr. Paul Van Cauwenberge

ir. Karolien Scheerlinck

Metaheuristic versus tailor-made approaches to optimization problems in the biosciences

Thesis submitted in fulfillment of the requirements for the degree of

Doctor (Ph.D) in Applied Biological Sciences

Academic year 2011-2012

Dutch translation of the title:
Metaheuristische versus probleemspecifieke aanpakken voor optimalisatieproblemen in de biologische wetenschappen

Please refer to this work as follows:
Karolien Scheerlinck (**2011**). *Metaheuristic versus tailor-made approaches to optimization problems in the biosciences*, PhD Thesis, Department of Mathematical Modelling, Statistics and Bioinformatics, Ghent University, Ghent, Belgium.

ISBN 978-90-5989-478-5

Acknowledgements — Dankwoord

Het begin van dit dankwoord geeft aan dat ik het einde heb bereikt van een belangrijk hoofdstuk uit mijn leven. Negen jaar heb ik vertoefd aan de universiteit: vijf jaar studeren en nadien vier jaar doctoreren. Tijdens mijn masterthesis ontdekte ik dat er zoveel interessante onderzoeksdomeinen bestaan waarin ik me graag meer wou verdiepen. Met andere woorden, ik was nog niet klaar om de universiteit te verlaten. Een eerste poging in die richting was de aanvraag van een IWT-bursaal met als onderwerp het bestuderen van droogteproblematiek in Kenia. Hoewel het bursaal niet werd toegekend, heb ik geen moment getwijfeld dat ik de komende jaren als onderzoeker wou doorbrengen. Dankzij professor Bernard De Baets, die mij aanbood een doctoraat aan zijn vakgroep te starten, kreeg ik een nieuwe kans. Op KERMIT verdiepen we ons in 'state-of-the-art' computationele methoden, die gebruikt kunnen worden om een meer gedetailleerde en accurate modellering van allerhande biologische processen te bekomen. Hoewel ik hiermee een totaal andere richting insloeg, ligt het onderzoek van deze groep dus meteen aan de basis van alle onderzoeksdomeinen die me kunnen boeien. Vier jaar lang heb ik de kans gekregen mij te verdiepen en mijn bijdrage te leveren aan de zoektocht naar oplossingen voor allerhande optimalisatieproblemen. Ik hoop dan ook dat dit werk als bewijs mag dienen van de boeiende zoektocht die ik hier heb beëindigd. Dit alles werd in de eerste plaats mogelijk gemaakt door mijn promotor Bernard. Bernard, wat ik me nog herinner alsof het gisteren was, is de manier waarop je me trachtte te overtuigen om bij KERMIT te komen werken, het gevoel dat je in mij geloofde. Ik wil je dan ook graag bedanken dat je mij deze kans geboden hebt, dat je mij ingewijd hebt in al deze boeiende onderzoeksdomeinen en voor de continue ondersteuning gedurende de hele looptijd. Kortom, ik kan me geen betere promotor voorstellen.

Hoewel ik de voorbije vier jaar heel veel heb bijgeleerd, zal niets me beter bijblijven dan de fantastische mensen die ik heb mogen leren kennen en de hulp en vriendschap die ik zo uitgebreid heb gekregen. Een doctoraat maak je niet solo, maar met de hulp van een heel orkest. En in mijn orkest zaten enkele fantastische muzikanten. Ik ben dan ook blij dat eindelijk het moment gekomen is waarop ik hen kan bedanken.

Dit doctoraatsproefschrift vat de resultaten van mijn onderzoek samen. Het is een compilatie geworden van onderzoeksproblemen die door verschillende mensen werden aangereikt. Al deze mensen verdienen een welgemeend woord van dank. Veerle, Ivan en Valentijn, bedankt voor het aanreiken van een interessant probleem, voor de leuke samenwerking en de boeiende discussies. Hilde, als collega's hebben we vier jaar samen een bureau gedeeld en gedurende die vier jaar ook vaak samengewerkt. Wat ik me vooral herinner uit deze samenwerking is je grote

behulpzaamheid. Je stond altijd voor me klaar en was steeds bereid te helpen, over welk probleem het ook ging. Bedankt!

Furthermore, I would like to thank the chairman, Christian Stevens, and the members of the jury, Hans de Meyer, Jorge Casillas, Niko Verhoest, Veerle Fievez and Sidharta Gautama, for reading my dissertation and suggesting valuable improvements to it. I am grateful for the time and effort put into this.

Enkele collega's waarbij ik steeds terecht kon met allerhande problemen zijn Peter, Joris, Jan V. en Karel. Peter en Joris, bedankt voor de leuke babbels en de statistische ondersteuning. Jan V., we hebben eerst samen gestudeerd en dan samen gedoctoreerd. We kennen elkaar dus al een tijdje. Gedurende deze periode heb ik je leren kennen als iemand die altijd bereid was zijn kennis te delen. Bedankt voor de leuke discussies en het grondig nalezen van mijn doctoraat. Karel, jij bent er als laatste bijgekomen, maar trouw aan het gezegde waren ook voor mij de laatste loodjes het zwaarst en gedurende deze zware maanden (vooral de laatste weken) heb jij voor heel veel lichtpuntjes gezorgd.

De boog kan niet altijd gespannen staan en tijdens de pauzes en na de werkuren waren er een heleboel collega's met wie ik (even) kon ontspannen. In de eerste plaats mijn bureaugenoten: Hilde, Karel, Michael en Willem. De voorbije vier jaar is er veel gewerkt, maar gelukkig ook voldoende gebabbeld. Ik zal met plezier aan deze momenten terugdenken. Tinne, bedankt om mij steeds te voorzien van de nodige calorieën en die er dan ook gezamenlijk weer te gaan aflopen. Jan B. en Lien, onze zwemmiddagen vormden een ideale uitlaat tijdens vermoeiende werkdagen. Lien, bedankt voor de hydrologische hulp. Isa, onze vriendschap gaat ver terug. Altijd kon ik bij jou terecht: het kopiëren van cursussen, de vele telefoontjes tijdens examenperiodes, de samenwerking bij het maken van verslagen, de troostende babbels tijdens moeilijkere perioden, en dies meer. Bedankt voor deze leuke periode. Tot slot kon ik tot voor kort steeds uitkijken naar de vele pauzes en middaguitstapjes met Soetkin. Hoewel je nog maar enkele maanden van de unief weg bent, merk ik het verschil enorm. Tot slot bedankt aan ons assisterend personeel, Timpe, Roos en Annie, voor alle administratieve hulp.

Ook buiten de werksfeer om zijn er nog een aantal mensen die ik graag wil bedanken voor hun vriendschap de voorbije jaren. Wouter, Karolien, Matthieu, Evi, Stephane, Ellen en Piet, de voor mij iets minder dan twee-wekelijks frequente café avondjes waren de ideale ontspanning na een zware werkdag. De loopavondjes met Sanne en Soetkin (als ze niet alweer geblesseerd was) vormden een leuke afwisseling. Sarah, Jan, Dieter, Inge, Pieter, Anneleen, Kevin, Soetkin en Boudewijn, bedankt voor de leuke weekends. Geertrui en Elke, bedankt voor de leuke stadsverkenningen. Virginie, bedankt voor de leuke etentjes. De echte vakanties werden steeds opgevuld met het verkennen van woeste natuurgebieden, in het gezelschap van Isa en Nelson. Hopelijk kunnen we nog veel van dergelijke vakanties meemaken. Een andere belangrijke ontspanningsfactor waren de leuke 'koken-eten-filmpje' avonden met

Soetkin. Sinds onze studies zijn we heel close geworden en sindsdien zijn we beste maatjes. Je behoort tot de weinige vrienden aan wie ik werkelijk alles kan zeggen. Bedankt voor de vele leuke momenten; ik zal naar 'Twee meisjes aan het werk' met veel heimwee terugkijken...

Naast de vrienden, is er ook de familie. Mama en papa, zonder jullie was dit niet mogelijk geweest. Bedankt voor de vele hulp en steun; jullie zijn de beste ouders die er bestaan! Daarnaast mag ik zeker mijn zus Katrien, Hans en Emiel niet vergeten. Hoewel we veel verschillende interesses en hobby's hebben, toch komen we steeds goed overeen en kan ik altijd bij jullie terecht. De laatste en tegelijkertijd de belangrijkste persoon die ik wil bedanken, is Jutho. Zonder jou had dit doctoraat er nooit gelegen. Sinds de eerste dag dat ik je ken, heb je mij 100 percent gesteund in alles wat ik wou bereiken en heb je mij daarbij steeds geholpen zoveel je kon. Dit jaar hebben we alletwee ons doctoraat afgewerkt en dat was zeker geen makkelijke periode. Bedankt om een glimlach op mijn gezicht te toveren tijdens momenten waarbij ik het compleet niet meer zag zitten, om mij te helpen alles te relativeren tijdens mijn stress- en paniekperiodes (en dat waren er veel) en om steeds tijd te maken voor het nalezen van mijn artikels en mijn doctoraat. Bedankt voor alles, ik zie je graag!

Karolien Scheerlinck
Gent 10/11/2011

Contents

1

Introduction

In many aspects of this world, there is one common principle, namely the search for an optimal state. Several examples can be given. At nanoscale, atoms that approach each other form a bond when this lowers the overall free energy of these atoms. The atoms interact in order to achieve an optimal state of minimal free energy [1]. Another example is the heating and then slowly cooling down of solid bodies. While the solid body is slowly cooled down, the molecules are trying to minimize their energy by forming an organized crystal lattice. In general, the adaptive strategy and evolutionary behaviour of organisms to a continuously changing environment can be seen as an optimization strategy guided by the search for a minimal energy state. This leads to the science of optimization, which is one of the oldest and most important sciences [2].

Optimization is more general than the search for a minimal energy state and refers to a branch of computational science that deals with finding acceptable solutions to given problems. These solutions are determined by the characteristics and requirements of the problem [3]. Optimization problems occur in diverse fields such as engineering [4, 5], manufacturing [6, 7], finance [8], medicine [9], computational art and music [10], physics [11], chemistry [12], *etc.* A large number of algorithms have been developed to solve this type of problems. Dependent on the characteristics of the problem, different algorithms are available. Therefore, in Part I of this dissertation, an overview is given of different categories of optimization. Next to the general concept of optimization, the different optimization algorithms used in this dissertation are discussed in detail.

In this dissertation, optimization problems are encountered in a variety of engineering applications related to the biosciences. Not all of these problems are optimization problems in their original formulations and tailor-made approaches to solve these problems can be found in the literature. However, it is often time-consuming and complex to develop a tailor-made solution method for every problem one is faced with. We therefore compare these methods with more general algorithms, based on universal concepts that are common to all optimization problems of a certain type or with certain characteristics. In particular, we will use the class of algorithms referred to as metaheuristics, a term that is explained in Chapter 2. These algorithms are generally applicable and we now examine whether they are equally capable of solving the problems at hand as the tailor-made approaches

constructed in the literature. If the original problem is not an optimization problem, we have to formulate an objective function so that an optimizer of this objective function is also a solution of the original problem. Whether this objective function is a simple function or a complex model, the metaheuristic algorithm is expected to find a suitable optimizer as long as certain general assumptions about the problem are fulfilled.

The flowchart in Figure 1.1 explains which parts and chapters of this dissertation are related and presents the order in which to read these chapters. Part I reviews some background material that is required for the work presented in this dissertation. Chapter 2, entitled 'General concepts of optimization', summarizes the different categories of optimization and introduces the concept of metaheuristics. We then discuss in more detail a number of metaheuristic optimization algoritms, both for continuous problems in Chapter 3 entitled 'Continuous optimization methods' and for discrete optimization problems in Chapter 4 entitled 'Combinatorial optimization methods'.

Part II discusses a subset selection problem, more specifically subset selection from a multi-experiment data set (Chapter 5). In this part, conventional subset selection methods are compared with an optimization approach based on combinatorial optimization methods, more specifically the metaheuristics Genetic Algorithms and Ant Colony Systems (Chapter 6). Therefore, for a complete understanding of this part, it is strongly recommended to first study Chapter 4. These subset selection methods are then applied to an agricultural case study (Chapter 7) concentrated around a large data set containing the concentration of 45 fatty acids in a large number of milk samples. These milk samples belong to multiple experiments. The objective is to select a subset of 100 milk samples that is informative for the total data set. At the same time the different experiments have to be sufficiently represented. When reading this part it is obvious that the recommended order is Chapter 5, followed by Chapters 6 and 7 and at last Chapter 8.

Part III discusses the calibration of a hydrologic model, more specifically a water and energy balance model (Chapter 11). A hydrologic model always consists of some model parameters that have to be determined before the model can be applied. In most cases, these model parameters can not be measured directly and an alternative strategy is required. When measured data for the output is available during a test period, we can try to find the set of model parameters for which the model best reproduces the measured output. The calibration is thus transformed into an optimization problem. As hydrologic models mostly result in a large number of output variables, we have to take into account all these variables to estimate the model parameters. However, the values of these output variables and the corresponding observations can have a different order of magnitude, which makes working with all these variables not straightforward (Chapter 9). A possible solution is the transformation of this problem to a multi-objective optimization problem,

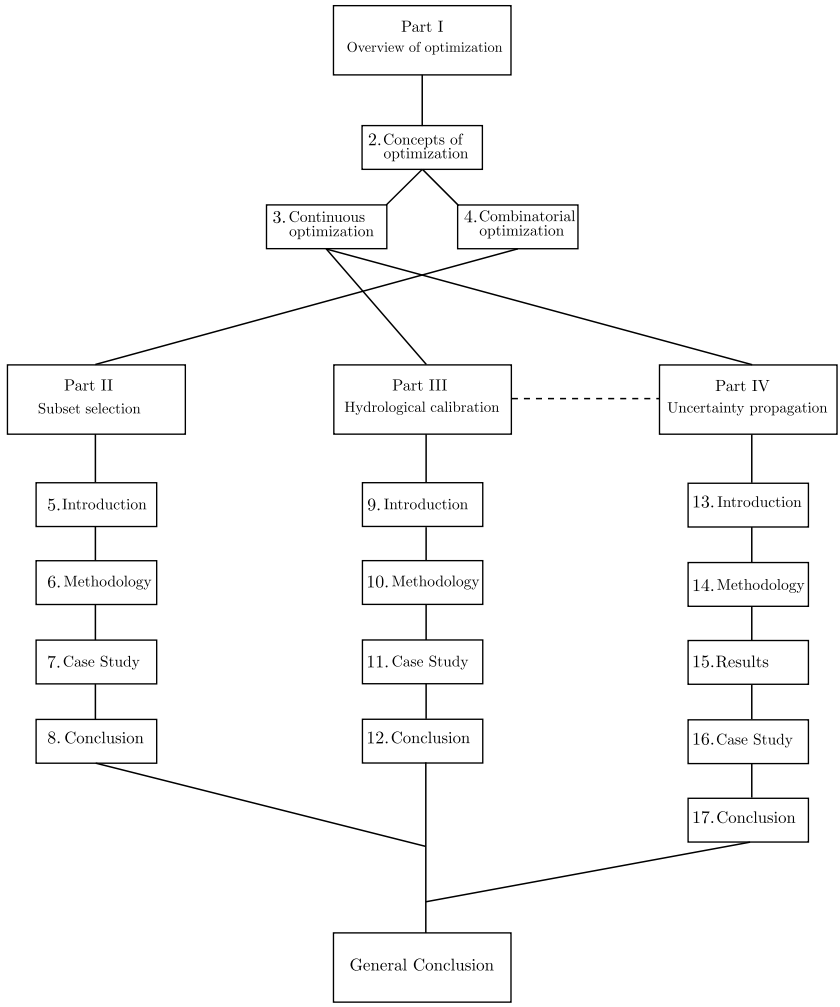


Figure 1.1: Flowchart that presents the connection between the different parts and chapters of this dissertation.

with as different objective functions the Root Mean Square Error (RMSE) of the different output variables, and construct a Pareto front. However, in case of a large number of output variables, this results in a high-dimensional Pareto front which makes this approach complicated. A problem specific solution is to work with the Multistart Weight-Adaptive Recursive Parameter Estimation (MWARPE) method. In this method, all variables are explicitly taken into account during the parameter update. A disadvantage of this method is that in MWARPE matrices have to be inverted with dimensionality equal to the number of observations, resulting in a restriction on the number of observations (Chapter 10) in the parameter estimation. It is also possible to merge the different objective functions into one objective

function and to optimize the latter objective function. In order to do this, the output variables should have the same order of magnitude and therefore the data should be rescaled. The objective of this part of this dissertation is to compare the MWARPE approach with the combination of the different objective functions into a single objective function. As optimization algorithm for the latter approach, we apply the metaheuristic Particle Swarm Optimization (PSO) (Chapter 10). Therefore, we recommend reading Chapter 3 before starting with this part. As in Part II, this part should be studied in the order Chapter 9, Chapter 10, Chapter 11 and finally Chapter 12.

The last part of this dissertation, Part IV, handles the propagation of uncertainty through mathematical equations, models, *etc.* (Chapter 13). Based on Zadeh's extension principle and the α-cut approach of Nguyen for non-interactive variables, we have also transformed this problem into an optimization problem (Chapter 14). The ability of PSO to solve complex continuous optimization problems was already proven in Part III, so that we also use PSO as optimization algorithm in this part. In order to reconfirm the power of this optimization algorithm, we compared this algorithm with a simple approach using a Gradient Descent approach based on Sequential Quadratic Programming and with another global optimization algorithm, namely Simplex Simulated Annealing (SIMPSA). This part is thus also based on continuous optimization and therefore it is necessary to first read Chapter 3. As Part III shows the capability of PSO, it is also interesting to read this part before going to Part IV. The objective of the current part is thus to develop a Fuzzy Calculator that makes uncertainty propagation through mathematical equations and models possible. First, we have to determine a suitable number of α-cuts. As it is difficult to determine the ideal number of α-cuts, we followed two approaches: either a fixed predetermined number is used, or an initially (very) small number is chosen that is subsequently increased according to a linearity criterion. Both a non-parallel and a parallel implementation are designed. The parallel version is restricted to work with PSO and employs communication to optimize its (internal) performance by exploiting the dependence between the various optimization problems (Chapter 14). The different configurations of the Fuzzy Calculator are evaluated on a set of benchmark functions (Chapter 15). Then, based on a generalization of Nguyen's α-cut approach for interactive variables, the Fuzzy Calculator is adapted to work with interactive variables as well. For this section, we restricted ourselves to the Fuzzy Calculator leading to the best results in the case of non-interactive variables and to interactivity described by the basic triangular norms (Chapter 15). Finally, the practical applicability of the Fuzzy Calculator is investigated in a case study (Chapter 16) making use of the inverse Integral Equation Model (IEM). This model relates the soil moisture to the backscatter values and roughness parameters. While backscatter values can easily be obtained from radar images, the determination of roughness parameters is more complex resulting in uncertainty. To describe the interactivity between

these parameters, the possibilistic Gustafson Kessel clustering algorithm is used. The Fuzzy Calculator is thus used to propagate the uncertainty of the roughness parameters through this model. As in previous parts, the recommended order to read this part is starting with Chapter 13, followed by Chapter 14, 15, 16 and 17.

Finally, the general conclusions, which can be drawn from this dissertation, are formulated. Obviously, this part can only be read after reading the total dissertation.

Part I

Overview of optimization

2

General concepts of optimization

This chapter outlines the general concepts of optimization problems [13]. Firstly, a categorization of optimization problems is given in Section 2.1. In this section, we briefly describe the different optimization problems that can be distinguished. Secondly, the theoretical concepts of continuous and discrete optimization are summarized in Sections 2.2 and 2.3. Finally, in Section 2.4, we briefly explain the term metaheuristics and illustrate when this branch of optimization algorithms will be applied.

2.1. Categorization of optimization problems

This dissertation will deal with optimization problems on many occasions. We always assume to have been provided with an objective function

$$f : \mathcal{S} \to \mathbb{K} \tag{2.1}$$

that maps input arguments from a general set $\mathcal{S}$ to a scalar output in $\mathbb{K}$. In order for optimization to be meaningful, $\mathbb{K}$ has to be a totally ordered set if we assume not to be dealing with multiple objective optimization. For simplicity, in this chapter we always assume to be dealing with a minimization problem. Typically, the output value is a real value — thus $\mathbb{K} = \mathbb{R}$ — and we are thus looking for an argument $\boldsymbol{x} \in \mathcal{S}$ that minimizes this output value. Maximization can be obtained by using $(-f)$ as objective function. Whereas the set $\mathcal{S}$ describes the type of input arguments $\boldsymbol{x}$ that are taken by f, there might be additional constraints on the allowed input arguments, which are also expressed using functions $c_i : \mathcal{S} \to \mathbb{R}$. A general optimization problem can then be defined as follows

$$\begin{aligned} &\min f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathcal{S} \\ &\text{subject to} \begin{cases} c_k(\boldsymbol{x}) = 0, & \text{for all } k \in \mathcal{E} = \{1, \ldots, m\} \\ c_l(\boldsymbol{x}) \geq 0, & \text{for all } l \in \mathcal{I} = \{m+1, \ldots, m+p\} \end{cases} \end{aligned} \tag{2.2}$$

with m the number of equality constraints and p the number of inequality constraints.

Two branches of optimization problems can be distinguished, depending on the properties of the set $\mathcal{S}$. In continuous optimization, $\mathcal{S}$ (Eq. (2.2)) can be mapped onto (a part of) $\mathbb{R}^n$. This means that we are dealing with an objective function f that depends on n real input parameters x_i $(i = 1, \ldots, n)$, which we denote as a vector $\boldsymbol{x} \in \mathbb{R}^n$, and outputs a real value $y = f(\boldsymbol{x}) \in \mathbb{R}$. Note that a complex input parameter can always be written using two independent real parameters. On the other hand, in discrete optimization, $\mathcal{S}$ (Eq. (2.2)) represents a countably infinite set. Within the class of discrete optimization problems, an important role is played by the class of combinatorial optimization problems. Here the input arguments represent permutations, combinations or selections and are thus further restricted. Specialized methods for this type of problem exist.

When additional constraints are present, we can define the *feasible region* Ω as

$$\Omega = \{\boldsymbol{x} \mid (\forall k \in \mathcal{E})(c_k(\boldsymbol{x}) = 0) \wedge (\forall l \in \mathcal{I})(c_l(\boldsymbol{x}) \geq 0)\}\,. \tag{2.3}$$

The feasible region thus contains all points $\boldsymbol{x}$ that satisfy the equality and inequality constraints. The optimization problem can then be reformulated as

$$\min f(\boldsymbol{x})\,, \quad \boldsymbol{x} \in \Omega\,. \tag{2.4}$$

A global solution $\boldsymbol{x}^*$ of an optimization problem is called a *minimizer* and should satisfy

$$f(\boldsymbol{x}^*) \leq f(\boldsymbol{x})\,, \quad \forall \boldsymbol{x} \in \Omega\,. \tag{2.5}$$

A general minimization problem can have any number of minimizers, ranging from zero to infinity. A necessary condition for a function to have at least one minimum is that the range of the function, *i.e.* the set $f(\Omega) = \{y \mid (\exists \boldsymbol{x} \in \Omega)(y = f(\boldsymbol{x}))\}$ is bounded from below. When the feasible region is finite or compact, this is always the case, if we assume that the objective function does not diverge in any point of Ω.

For continuous optimization problems, in case of a metrical space $\mathcal{S}$, there is a natural notion of a neighbourhood for every point $\boldsymbol{x}$, by defining

$$\mathcal{S}(\boldsymbol{x}, \epsilon) = \{\boldsymbol{y} \mid ||\boldsymbol{x} - \boldsymbol{y}|| < \epsilon\}\,, \tag{2.6}$$

with $||\cdot||$ the standard norm following from the metric. The optimization problem can then also have local solutions $\boldsymbol{x}^*$ that satisfy

$$(\exists \epsilon > 0)(\forall \boldsymbol{x} \in \mathcal{S}(\boldsymbol{x}^*, \epsilon) \cap \Omega)(f(\boldsymbol{x}^*) \leq f(\boldsymbol{x}))\,.$$

For discrete optimization problems, this natural notion of a neighbourhood cannot longer be used for the definition of a local solution, since for ϵ sufficiently small, the neighbourhood of a point contains only this point and every point would then

be a local solution. For this type of optimization problems, it depends on the application, or on the algorithm used, which notion of neighbourhood can be used in order to define local optima.

2.2. Theoretical concepts of continuous optimization

This section summarizes the theoretical foundations of multidimensional continuous optimization problems. Necessary and sufficient conditions for local optima of (twofold) continuously differentiable objective functions f are stated. It is useful to first treat the unconstrained case, where any point $\boldsymbol{x} \in \mathbb{R}^n$ is allowed. Constrained optimization will be studied afterwards. The contents of this section is based on [13].

2.2.1. Unconstrained optimization

The unconstrained continuous optimization problem is given by

$$\min f(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n \,. \tag{2.7}$$

If the function f is continuously differentiable at $\boldsymbol{x}^*$, then a necessary condition for $\boldsymbol{x}^*$ to be a local minimizer is that

$$\boldsymbol{\nabla}_{\boldsymbol{x}} f(\boldsymbol{x}^*) = [\frac{\partial f}{\partial x_1}(\boldsymbol{x}^*), \ldots, \frac{\partial f}{\partial x_n}(\boldsymbol{x}^*)]^{\mathrm{T}} = \boldsymbol{0} \,, \tag{2.8}$$

with $\boldsymbol{\nabla}_{\boldsymbol{x}} f(\boldsymbol{x})$ the gradient of the function f at the point $\boldsymbol{x}$. If the function f is continuously differentiable in $\mathbb{R}^n$, then we can locate all candidate local minimizers by solving the set of equations

$$\boldsymbol{\nabla}_{\boldsymbol{x}} f(\boldsymbol{x}) = \boldsymbol{0} \,. \tag{2.9}$$

Solutions $\boldsymbol{x}^*$ to this equation are called stationary points. They can be minimizers, maximizers or saddle points. If the function f is twofold continuously differentiable at a possible solution $\boldsymbol{x}^*$, then a necessary condition for this stationary point to be a minimizer is that

$$\nabla_{\boldsymbol{xx}} f(\boldsymbol{x^*}) = \begin{bmatrix} \frac{\partial^2 f}{(\partial x_1)^2}(\boldsymbol{x^*}) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(\boldsymbol{x^*}) & \cdots \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(\boldsymbol{x^*}) & \frac{\partial^2 f}{(\partial x_2)^2}(\boldsymbol{x^*}) & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \geq 0 \,, \tag{2.10}$$

with $\nabla_{\boldsymbol{x}x} f(\boldsymbol{x})$ the Hessian of the function f at the point $\boldsymbol{x}$. This implies that all eigenvalues of the Hessian $\nabla_{\boldsymbol{xx}} f(\boldsymbol{x^*})$ are greater than or equal to zero. The presence

of zero eigenvalues indicates that $\boldsymbol{x}^*$ cannot be shown to be a local minimizer by restricting to a second order Taylor expansion, and higher order expansions will be necessary. A sufficient (but not necessary) condition for $\boldsymbol{x}^*$ to be a minimizer is that $\nabla_{\boldsymbol{xx}} f(\boldsymbol{x}^*) > 0$.

2.2.2. Constrained optimization

A constrained continuous optimization problem can be formulated as

$$\min f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \mathbb{R}^n$$
$$\text{subject to} \begin{cases} c_k(\boldsymbol{x}) = 0, & \text{for all } k \in \mathcal{E} = \{1, \ldots, m\} \\ c_l(\boldsymbol{x}) \geq 0, & \text{for all } l \in \mathcal{I} = \{m+1, \ldots, m+p\} \end{cases} \tag{2.11}$$

with m the number of equality constraints and p the number of inequality constraints. Note that Ω denotes the feasible region, *i.e.* all points $\boldsymbol{x}$ that satisfy all constraints. For every point $\boldsymbol{x} \in \Omega$, we can define a set of indices, called the *set of active constraints* $\mathcal{A}(\boldsymbol{x})$ as

$$\mathcal{A}(\boldsymbol{x}) = \{i \in (\mathcal{E} \cup \mathcal{I}) \mid c_i(\boldsymbol{x}) = 0\} = \mathcal{E} \cup \{i \in \mathcal{I} \mid c_i(\boldsymbol{x}) = 0\}. \tag{2.12}$$

The active constraints are both the equality constraints and the inequality constraints for which $\boldsymbol{x}$ is at the boundary of the inequality.

If at a local minimizer $\boldsymbol{x}^*$ both the function and the active constraints are continuously differentiable, we can also formulate a set of necessary conditions. For that purpose, we define a Lagrangian

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_{i \in \mathcal{E} \cup \mathcal{I}} \lambda_i c_i(\boldsymbol{x}), \tag{2.13}$$

where the vector $\boldsymbol{\lambda}$ contains the Lagrange multipliers λ_i, for all $i \in \mathcal{E} \cup \mathcal{I}$. The Karush-Kuhn-Tucker (KKT) conditions state that when $\boldsymbol{x}^*$ is a local minimizer, there exists a Lagrange multiplier vector $\boldsymbol{\lambda^*}$ such that

$$\begin{cases} \boldsymbol{\nabla}_{\boldsymbol{x}} L(\boldsymbol{x^*}, \boldsymbol{\lambda^*}) = 0 & \\ c_i(\boldsymbol{x^*}) = 0, & \text{for all } i \in \mathcal{E} \\ c_i(\boldsymbol{x^*}) \geq 0, & \text{for all } i \in \mathcal{I} \\ \lambda_i^* \geq 0, & \text{for all } i \in \mathcal{I} \\ \lambda_i^* c_i(\boldsymbol{x^*}) = 0, & \text{for all } i \in \mathcal{E} \cup \mathcal{I} \end{cases} \tag{2.14}$$

The last line, known as the *complementarity* condition, states that the Lagrange multiplier $\lambda_i^* = 0$ for every inactive constraint $i \in \mathcal{I} \setminus \mathcal{A}(\boldsymbol{x}^*)$. It is thus no problem if the gradient of an inactive inequality constrained is not defined at $\boldsymbol{x}^*$. We can

redefine the Lagrange function as

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_{i \in \mathcal{A}(\boldsymbol{x})} \lambda_i c_i(\boldsymbol{x}) \,, \tag{2.15}$$

However, continuous differentiability of the function f and the active constraints c_i, $i \in A(\boldsymbol{x}^*)$, at the point $\boldsymbol{x}^*$ is not sufficient for the KKT conditions to hold. We need additional conditions that 'qualify' the constraints. One possible condition for which the KKT conditions will hold is known as the 'Linear Independent Constraint Qualification' (LICQ) and requires

$$\sum_{i \in \mathcal{A}(\boldsymbol{x}^*)} \alpha_i \boldsymbol{\nabla}_{\boldsymbol{x}} c_i(\boldsymbol{x}^*) = \mathbf{0} \Leftrightarrow \forall i \in \mathcal{A}(\boldsymbol{x}^*) : \alpha_i = 0 \tag{2.16}$$

This condition states that the gradients of the active constraints should constitute a linearly independent set of vectors. Other constraint qualifications are also possible. Without them, the Lagrange multiplier vector $\boldsymbol{\lambda}^*$ corresponding to a local minimizer $\boldsymbol{x}$ might not be unique, or might even not exist.

We will briefly try to motivate the origin of the KKT conditions. If the functions c_i encompassing the constraints are continuously differentiable, we can define a cone of feasible directions $\mathcal{F}_1(\boldsymbol{x})$, in which we can move away from a point $\boldsymbol{x} \in \Omega$ without leaving Ω, in the following way:

$$\begin{aligned} \mathcal{F}_1(\boldsymbol{x}) = \{\boldsymbol{d} \in \mathbb{R}^n \,|(\forall i \in \mathcal{E})(\boldsymbol{d}^{\mathrm{T}} \boldsymbol{\nabla}_{\boldsymbol{x}} c_i(\boldsymbol{x}) = 0) \\ \wedge (\forall i \in \mathcal{A}(\boldsymbol{x}) \cap \mathcal{I})(\boldsymbol{d}^{\mathrm{T}} \boldsymbol{\nabla}_{\boldsymbol{x}} c_i(\boldsymbol{x}) \geq 0)\} \,. \end{aligned}$$

If $\boldsymbol{x}^*$ is a local minimizer, the function f must not decrease when moving away from $\boldsymbol{x}^*$ in a direction $\boldsymbol{d} \in \mathcal{F}_1(\boldsymbol{x}^*)$. When taking an infinitesimal step from $\boldsymbol{x}^*$ in the direction of $\boldsymbol{d}$ proportional to a small constant $\eta > 0$, a first order Taylor expansion can be used, where terms of second and higher order in η (denoted as $O(\eta^2)$) are ignored. The function value in $\boldsymbol{x}^* + \eta \boldsymbol{d}$ can thus be approximated by

$$f(\boldsymbol{x}^* + \eta \boldsymbol{d}) \approx f(\boldsymbol{x}^*) + \eta \boldsymbol{d}^T \boldsymbol{\nabla}_{\boldsymbol{x}} f(\boldsymbol{x}^*) \geq f(\boldsymbol{x}^*) \,,$$

and we can infer that $\boldsymbol{d}^{\mathrm{T}} \boldsymbol{\nabla}_{\boldsymbol{x}} f(\boldsymbol{x}^*) \geq 0$, for all $\boldsymbol{d} \in \mathcal{F}_1(\boldsymbol{x}^*)$. At this point, LICQ is required in order to conclude from this condition that

$$\boldsymbol{\nabla}_{\boldsymbol{x}} f(\boldsymbol{x}^*) = \sum_{i \in \mathcal{A}(\boldsymbol{x}^*)} \lambda_i \boldsymbol{\nabla}_{\boldsymbol{x}} c_i(\boldsymbol{x}^*) \,, \tag{2.17}$$

with $\lambda_i \geq 0$, for all $i \in \mathcal{I} \cap \mathcal{A}(\boldsymbol{x}^*)$. This is precisely the content of the KKT conditions.

A point $\boldsymbol{x}^*$ that fulfills the Karush-Kuhn-Tucker conditions with corresponding Lagrange vector $\boldsymbol{\lambda}^*$ is a stationary point of the function $f(\boldsymbol{x})$ in Ω. Before trying

to impose conditions on the Hessian of f, we define the concept of *weakly active constraints* as those inequality constraints for which $\lambda_i^* = 0$, and *strongly active constraints* as the inequality constraints for which $\lambda_i^* > 0$. Finally, we define *strict complementarity* as $\lambda_i^* > 0$, for all $i \in \mathcal{I} \cap \mathcal{A}(\boldsymbol{x}^*)$. Thus, strict complementarity is satisfied if all active constraints are strongly active.

When examining the second order effect of variations with respect to a stationary point $\boldsymbol{x}^*$, it is sufficient to look in feasible directions $\boldsymbol{w}$ for which $\boldsymbol{w}^T \boldsymbol{\nabla}_{\boldsymbol{x}} f(\boldsymbol{x}^*) = 0$. We thus define the cone of directions

$$\begin{aligned}\mathcal{F}_2(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) = \{\boldsymbol{w} \in \mathcal{F}_1(\boldsymbol{x}) \mid & \big((\forall i \in \mathcal{E})(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{\nabla}_{\boldsymbol{x}} c_i(\boldsymbol{x}^*) = 0)\big) \\ & \wedge \big((\forall i \in \mathcal{A}(\boldsymbol{x}^*) \cap \mathcal{I})(\lambda_i^* > 0 \Rightarrow \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\nabla}_{\boldsymbol{x}} c_i(\boldsymbol{x}^*) = 0)\big) \\ & \wedge \big((\forall i \in \mathcal{A}(\boldsymbol{x}^*) \cap \mathcal{I})(\lambda_i^* = 0 \Rightarrow \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\nabla}_{\boldsymbol{x}} c_i(\boldsymbol{x}^*) \geq 0)\big)\}\,.\end{aligned}$$

$\mathcal{F}_2$ includes all directions $\boldsymbol{w}$ in which the gradient $\boldsymbol{\nabla}_{\boldsymbol{x}}$ is zero. If $\boldsymbol{x}^*$ is a local minimizer of $f(\boldsymbol{x})$ in Ω, f is twofold continuously differentiable in $\boldsymbol{x}^*$ and LICQ is satisfied, then we have as necessary condition that

$$\boldsymbol{w}^T \nabla_{\boldsymbol{xx}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) \boldsymbol{w} \geq 0, \qquad \text{for all } \boldsymbol{w} \in \mathcal{F}_2(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)\,.$$

The appearance of directions $\boldsymbol{w}$ for which the equality to zero is satisfied indicates that we have to investigate higher order terms in the Taylor expansion.

On the other hand, if $\boldsymbol{x}^*$ is a point that satisfies the KKT conditions, and $\boldsymbol{\lambda}^*$ is the corresponding Lagrange multiplier vector, and

$$\boldsymbol{w}^T \nabla_{\boldsymbol{xx}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) \boldsymbol{w} > 0, \qquad \text{for all } \boldsymbol{w} \in \mathcal{F}_2(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)\,,$$

then $\boldsymbol{x}^*$ is definitely a local minimizer. This sufficient (but not necessary) condition does not require that LICQ is fulfilled.

Finally, when strict complementarity holds, then the cone $\mathcal{F}_2(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$ is a vector space and we can find a basis matrix B with columns that span $\mathcal{F}_2(\boldsymbol{x}^*, \boldsymbol{\lambda}^*)$. We can then rephrase the necessary (sufficient) condition in terms of the positive semidefiniteness (positive definiteness) of the matrix $B^T \nabla_{\boldsymbol{xx}} L(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) B$.

2.3. Theoretical concepts of discrete optimization

For discrete optimization problems, there is no general definition of a neighbourhood of a point $\boldsymbol{x} \in \mathcal{S}$. We only have the defining relation in Eq. (2.5) of a global optimizer. In many applications — especially in combinatorial optimization problems [14, 15, 16] — the set Ω of feasible input arguments is finite, which ensures the existence of a global minimum, even when there are multiple minimizers $\boldsymbol{x}^*$ producing this minimal function value. For a finite set, we can in theory find the global solution

$\boldsymbol{x}^*$ by performing an exhaustive search, *i.e.* by evaluating the objective function f for all $\boldsymbol{x} \in \Omega$. We will always assume that we can easily construct a scheme to generate all feasible solutions $\boldsymbol{x} \in \Omega$. However, many interesting problems have an intrinsic notion of problem size, and are such that the number of feasible solutions $|\Omega|$ increases faster than any polynomial — often exponentially fast — in the problem size. The computation time of exact algorithms for obtaining the global minimizer $\boldsymbol{x}^*$ will often follow this "faster than any polynomial" scaling. A theoretical categorization of different types of problems is possible by converting them into decision problems. Converting an optimization problem to a decision problem is possible by asking the question "Is there an $\boldsymbol{x} \in \Omega$ such that $f(\boldsymbol{x}) < c$" with c some constant. If this decision problem has answer 'yes', then we can decrease the value of c and ask the question again, until we have singled out the global minimum. Decision problems are categorized in complexity classes. The complexity class NP contains all *nondeterministic polynomial time* problems, *i.e.* problems for which a given solution $\boldsymbol{x}$ can actually be checked to produce 'yes' by an algorithm with a computation time that is polynomial in the problem size. Within the class NP, we can find problems for which we can actually find solutions $\boldsymbol{x}$ that produce 'yes' in polynomial time. This subclass is labeled P. Though unproven, it is generally believed that P is a strict subset of NP and thus $\mathsf{P} \neq \mathsf{NP}$. The class NP then also contains problems which are much 'harder', in that one cannot find an algorithm that constructs solutions $\boldsymbol{x}$ of the decision problem in polynomial time. More generally, we can define any problem (not only decision problems but also e.g. optimization problems) to be NP-*hard* if they are 'at least as hard as the hardest problem' in NP. This definition requires that any problem in NP can be converted into this NP-hard problem by a transformation that is polynomial in the problem size. The set of NP-hard decision problems that are in NP are called the subset of NP-complete problems (Figure 2.1) [17]. We can

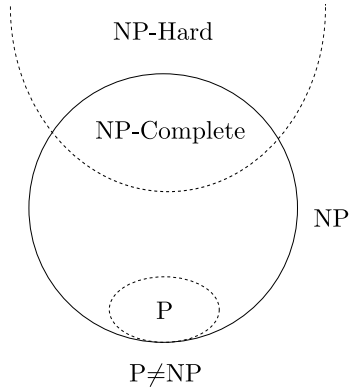


Figure 2.1: Venn diagram for P, NP, NP-Complete, and NP-Hard set of problems.

thus categorize combinatorial optimization problems as being NP-hard or not. For

NP-hard optimization problems, it is impossible to construct an algorithm that exactly finds the global optimizer $\boldsymbol{x}^*$ in a time that scales polynomial in the problem size [18, 19]. Problems that can be solved exactly in a time that scales polynomial are not NP-hard. Unfortunately, many interesting problems are NP-hard.

2.4. Metaheuristics

Section 2.2 summarizes the necessary and sufficient conditions that are satisfied by local minimizers of continuous optimization problems. In many cases, the number of local minimizers — or even the number of stationary points — is finite, and finding them would result in an easy selection of the global minimizer. Unfortunately, the necessary conditions generally constitute a highly complex set of non-linear equations for which no exact solution method is available. There are only few exceptions, such as convex optimization problems — where there is a single optimum — or problems with a quadratic objective function and linear constraints. In many applications, the function f itself is a complex model for which the gradient and Hessian cannot be explicitly calculated or which does not even satisfy the constraint of being continuously differentiable. Different algorithms for converging an initial guess $\boldsymbol{x}_0$ towards a local minimizer exist. Some but not all of them rely on continuous differentiability of the objective function. The local minimum $\boldsymbol{x}^*$ to which these algorithms converge depends on the initial solution $\boldsymbol{x}_0$, and there is no guarantee that all local minima, and thus the global minimum has been found. Most algorithms of this kind will iteratively create a path of solutions along which the objective function monotonically decreases. In that sense, they cannot escape from local minima and are local optimization algorithms. Examples include gradient descent and sequential quadratic programming, or the simplex algorithm as a derivative-free example.

For combinatorial problems that are NP-hard, one will also have to resort to approximate algorithms if the problem size is large. In some cases, such algorithms contain an intrinsic notion of a neighbourhood of candidate solutions $\boldsymbol{x}$, and such algorithms face the same problem as their continuous counterparts, *i.e.* they can get stuck in local optima.

In order to avoid these problems, several general approaches, often called metaheuristics, have been proposed. The word heuristic has its origin in the old Greek word 'heuriskein', which means the discovery of new strategies to solve problems. The suffix 'meta', also a Greek word, means 'upper level methodology'. A metaheuristic is thus a set of algorithmic concepts that can be used to define heuristic methods applicable to a wide set of different problems. The central idea of any metaheuristic algorithm is to iteratively generate new candidate solutions from old solutions. These algorithms make few or no assumptions about the problem being

optimized and they allow to tackle large-size problems by delivering satisfactory solutions in a reasonable time [18, 20].

Many definitions of metaheuristics are proposed in literature [21], we will cite here one of the most exhaustive [22]:

> *Metaheuristics are typically high-level strategies which guide an underlying, more problem specific heuristic, to increase their performance. The main goal is to avoid the disadvantages of iterative improvement and, in particular, multiple descent by allowing the local search to escape from local optima. This is achieved by either allowing worsening moves or generating new starting solutions for the local search in a more 'intelligent' way than just providing random initial solutions. Many of the methods can be interpreted as introducing a bias such that high quality solutions are produced quickly. This bias can be of various forms and can be cast as descent bias (based on the objective function), memory bias (based on previously made decisions) or experience bias (based on prior performance). Many of the metaheuristic approaches rely on probabilistic decisions made during the search. But, the main difference to pure random search is that in metaheuristic algorithms randomness is not used blindly but in an intelligent, biased form.*

The main reason for using metaheuristics is to try to construct a method that is able to escape from local minima and has a non-zero probability of converging to the global minimum. It is only legitimate to use metaheuristics to solve an optimization problem if it is not possible to solve the problem using an efficient exact algorithm, as is mostly the case for NP-hard optimization problems, where exact algorithms would require too much search time. However, metaheuristics are also used for P class problems with a large number of input variables, for P class problems with hard real-time algorithms, for NP-hard problems with moderate size and/or difficult structures of the input variables, or for optimization problems with time-consuming objective functions and/or constraints. For continuous problems, metaheuristic algorithms are used when derivative-based methods fail, because the objective function is discontinuous, is strongly nonlinear or ill-conditioned [20]. It is important to note that metaheuristics do not guarantee the optimality of the obtained solution.

As metaheuristic algorithms are developed in order to avoid getting trapped in local minima, the termination conditions of metaheuristic algorithms are more complex than simple convergence to a fixed point $\boldsymbol{x}^*$ or a fixed function value f^*. Depending on the algorithm, different termination conditions exist, such as a maximum CPU time, a maximum number of iterations, a maximum number of iterations without improvement, *etc.*

Different classification schemes exist for metaheuristic algorithms. The underlying principle can be nature-inspired or non-nature inspired. The algorithm can be

population-based or single-solution-based. It can employ a dynamic or a static objective function, one or multiple neighbourhood structures. Some algorithms use memory to store and exploit information from previous decisions, while others work instantaneous and only use the current candidate solution to generate the next guess. This section will be restricted to the difference between *single-solution-based metaheuristics* (S-metaheuristics) and *population-based metaheuristics* (P-metaheuristics).

In S-metaheuristics, the solution moves along a trajectory in the search space S of the problem. At every iteration, the current candidate solution is moved to a new solution in its immediate neighbourhood. Examples of such algorithms include Simulated Annealing, Simplex-Simulated Annealing and Tabu Search. P-metaheuristics start with the initialization of a population. At every iteration, a new population is generated that replaces the current population. P-metaheuristics differ in the way populations are generated and in the search memory used during the search. Many P-metaheuristics use nature-inspired rules to generate new populations. Some examples are evolutionary algorithms, Particle Swarm Optimization, Bee Colony Optimization and Ant Colony Optimization [20].

Metaheuristic concepts can also be employed to deal with constraints. Different strategies exist: reject strategies, penalizing strategies, repairing strategies, decoding strategies and preserving strategies [20]. In the reject strategies, infeasible solutions are discarded and only feasible solutions are kept during the search. Penalizing strategies penalize the infeasible solutions by extending the unconstrained objective function by a penalty function. Repairing strategies act by transforming an infeasible solution into a feasible one. In the decoding strategies, a mapping $\mathcal{R} \to \mathcal{S}$ is used that associates with each representation $r \in \mathcal{R}$ a feasible solution $\boldsymbol{x} \in \mathcal{S}$ in the search space. The whole optimization problem can then be defined in $\mathcal{R}$. Preserving strategies start with feasible initial solutions and will generate new solutions by only applying operations that preserve the feasibility.

3

Continuous optimization methods

This chapter describes a few continuous optimization algorithms, which will be applied when studying continuous optimization problems in the remainder of this dissertation (model calibration in Part III and uncertainty propagation in part IV). The first optimization algorithm, Sequential Quadratic Programming (Section 3.1), is a derivative-based algorithm that exploits the theoretical conditions of continuous optimization as outlined in Section 2.2. The second and third optimization algorithms, Particle Swarm Optimization and Simplex-Simulated Annealing (Sections 3.2 and 3.3), belong to the metaheuristic optimization algorithms. For Sequential Quadratic programming, we made use of the standard implementation of the programming environment Octave. The implementation of the Simplex-Simulated Annealing algorithm was taken from [23] and [24, 25] with the author's permission. On the other hand, Particle Swarm Optimization was implemented from scratch and modified to suit our goals as will be explored in later chapters.

3.1. Sequential Quadratic Programming

Sequential Quadratic Programming (SQP) is a very successful deterministic approach for solving continuous nonlinear constrained optimization problems. This optimization algorithm is based on the theoretical concepts described in Sections 2.2.1 and 2.2.2 of Chapter 2. As in Section 2.1, $\mathcal{S}$ is the set of all possible solutions, a possible solution of the optimization problem is denoted as $\boldsymbol{x}$ and the constraints, Lagrange multipliers and objective function are presented by c, λ and f. The key concept of SQP is to approximate, at every iteration k, the nonlinear optimization problem by a quadratic problem with linear constraints. Solving this quadratic problem defines a step $\boldsymbol{p_x}$ in which to look for a better approximation $\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \alpha \boldsymbol{p_x}$ of the minimizer. At the same time, the algorithm will also determine a better approximation $\boldsymbol{\lambda}(k+1)$ for the Lagrange multipliers.

It is possible that the objective function is bounded from below in the feasible region Ω, but that it will become unbounded in the region determined by the linear approximation of the nonlinear constraints. A way to include the nonlinear effects of the constraints is by using the full nonlinear Lagrangian $L(\boldsymbol{x}, \boldsymbol{\lambda}(k))$ rather than

the objective function $f(\boldsymbol{x})$ in the quadratic approximation of the optimization problem. Having obtained the solution $\boldsymbol{x}(k)$ and the corresponding Lagrange multiplier vector $\boldsymbol{\lambda}(k)$, the next step is to solve

$$\min \frac{1}{2}\boldsymbol{p}_{\boldsymbol{x}}^{\mathrm{T}}\nabla_{\boldsymbol{xx}}L(\boldsymbol{x}(k),\boldsymbol{\lambda}(k))\boldsymbol{p}_{\boldsymbol{x}} + \boldsymbol{\nabla}_{\boldsymbol{x}}L(\boldsymbol{x}(k),\boldsymbol{\lambda}(k))^{\mathrm{T}}\boldsymbol{p}_{\boldsymbol{x}}$$
$$\text{subject to} \begin{cases} c_i(\boldsymbol{x}(k)) + \boldsymbol{\nabla}_{\boldsymbol{x}}c_i(\boldsymbol{x}(k))^{\mathrm{T}}\boldsymbol{p}_{\boldsymbol{x}} = 0, \\ \qquad\qquad \text{for all } i \in \mathcal{E} = \{1,\ldots,m\} \\ c_j(\boldsymbol{x}(k)) + \boldsymbol{\nabla}_{\boldsymbol{x}}c_j(\boldsymbol{x}(k))^{\mathrm{T}}\boldsymbol{p}_{\boldsymbol{x}} \geq 0, \\ \qquad\qquad \text{for all } j \in \mathcal{I} = \{m+1,\ldots,m+p\} \end{cases} \tag{3.1}$$

Let us first consider the case where there are only equality constraints. Solving the quadratic problem is then quite easy, as it comes down to solving the linear system

$$\begin{bmatrix} \nabla_{\boldsymbol{xx}}L(\boldsymbol{x}(k),\boldsymbol{\lambda}(k)) & -\boldsymbol{\nabla}_{\boldsymbol{x}}\boldsymbol{c}(\boldsymbol{x}(k)) \\ (\boldsymbol{\nabla}_{\boldsymbol{x}}\boldsymbol{c}(\boldsymbol{x}(k)))^{\mathrm{T}} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_{\boldsymbol{x}} \\ \boldsymbol{p}_{\boldsymbol{\lambda}} \end{bmatrix} = \begin{bmatrix} -\boldsymbol{\nabla}_{\boldsymbol{x}}L(\boldsymbol{x}(k),\boldsymbol{\lambda}(k)) \\ -\boldsymbol{c}(\boldsymbol{x}(k)) \end{bmatrix} \tag{3.2}$$

with $-\boldsymbol{\nabla}_{\boldsymbol{x}}L(\boldsymbol{x}(k),\boldsymbol{\lambda}(k)) = -\boldsymbol{\nabla}_{\boldsymbol{x}}f(\boldsymbol{x}(k)) + \boldsymbol{\nabla}_{\boldsymbol{x}}\boldsymbol{c}(\boldsymbol{x}(k))\boldsymbol{\lambda}(k)$, and where we have symbolically denoted the Jacobian of the constraints as

$$\boldsymbol{\nabla}_{\boldsymbol{x}}\boldsymbol{c}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial c_1}{\partial x_1}(\boldsymbol{x}) & \frac{\partial c_2}{\partial x_1}(\boldsymbol{x}) & \cdots \\ \frac{\partial c_1}{\partial x_2}(\boldsymbol{x}) & \frac{\partial c_2}{\partial x_2}(\boldsymbol{x}) & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}. \tag{3.3}$$

We can then set $\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \boldsymbol{p}_{\boldsymbol{x}}$ and $\boldsymbol{\lambda}(k+1) = \boldsymbol{\lambda}(k) + \boldsymbol{p}_{\boldsymbol{\lambda}}$. This approach is equivalent to applying the Newton method for finding solutions of a set of nonlinear equations to the set of KKT conditions. The matrix in the left-hand side of Eq. (3.2) can be called the KKT Jacobian or KKT matrix. It can be shown that the KKT matrix is nonsingular in a neighbourhood of a minimizer $\boldsymbol{x}^*$ that satisfies the sufficient condition for a minimum and constraints satisfying the LICQ condition. The KKT matrix is however indefinite and there are different algorithms for solving this linear system optimally, depending on the number of (equality) constraints m and the number of degrees of freedom $n - m$.

When inequality constraints are also present, we have to use an algorithm for quadratic programming to solve the quadratic subproblem at every iteration. In particular, the chosen algorithm —active set methods are typically chosen— will yield a descent direction $\boldsymbol{p}_{\boldsymbol{x}}$. If $\nabla_{\boldsymbol{xx}}L(\boldsymbol{x}(k),\boldsymbol{\lambda}(k))$ is positive definite in the tangent space of the active constraints, the search direction follows from solving Eq. (3.2) with all active constraints at the current iterate $\boldsymbol{x}(k)$ included. If, however,

$\nabla_{\boldsymbol{xx}} L(\boldsymbol{x}(k), \boldsymbol{\lambda}(k))$ contains negative eigenvalues in the tangent space of the active constraints, then a direction $\boldsymbol{p_x}$ is sought that is both a negative curvature direction and a non-ascent direction.

To determine a new iterate $\boldsymbol{x}(k+1)$ from $\boldsymbol{x}(k)$, different strategies are possible. In the *trust-region* methods, the quadratic subproblem is modified in such a way that the step $\boldsymbol{p_x}$ is automatically limited to a region where the quadratic approximation is assumed to be good. One can then safely set $\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \boldsymbol{p_x}$. Typically, this also solves the problem of negative eigenvalues in the reduced Hessian. *Line-search* algorithms search a value of α such that $\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \alpha \boldsymbol{p_x}$ minimizes the objective function without violating the constraints. Often, rather than using simply the objective function, one uses a *merit function* to control the step size α. Merit functions also use information about the constraints and are supposed to help converging from a remote starting point to the global solution. They are also used in trust-region methods, to determine whether the trust region radius has to be modified. A typical merit function to be used in combination with SQP is the *augmented Lagrangian*, which is given by

$$L_A(\boldsymbol{x}, \boldsymbol{\lambda}; \mu) = f(\boldsymbol{x}) - \sum_{i \in \mathcal{E}} \lambda_i c_i(\boldsymbol{x}) + \frac{1}{2\mu} c_i(\boldsymbol{x})^2 \tag{3.4}$$

when only equality constraints are present. For inequality constraints, one has to introduce slack variables in the formulation of the augmented Lagrangian. One can show that for μ sufficiently small, the optimizer $\boldsymbol{x}^*$ will be an unconstrained strict local minimizer of $L_A(\boldsymbol{x}, \boldsymbol{\lambda}^*; \mu)$ and that the Hessian of L_A is positive definite.

Finally, there are also different methods for calculating or approximating the Hessian in the quadratic subproblem. Since this object contains all second derivatives of the objective function and the constraints, the computation of the Hessian is a very time costly operation (if no exact result is provided). The general strategy is to update the Hessian $\nabla_{\boldsymbol{xx}} L(\boldsymbol{x}(k+1), \boldsymbol{\lambda}(k+1))$ from the previous value $\nabla_{\boldsymbol{xx}} L(\boldsymbol{x}(k), \boldsymbol{\lambda}(k))$ using an update scheme such as the rank-two Powell-Symmetric-Broyden (PSB) update [13] or the rank-two Broyden-Fletcher-Goldfarb-Shanno (BFGS) update [13]. Some algorithms do not use the Hessian of the Lagrangian but rather the Hessian of the augmented Lagrangian $L_A(\boldsymbol{x}, \boldsymbol{\lambda}; \mu)$, which will always be positive definite for μ sufficiently small.

Clearly, a complete SQP algorithm is very complex, and many different variants and modifications are possible. We refer to [26, 13] for further information.

3.2. Particle Swarm Optimization

Particle Swarm Optimization is a population-based optimization algorithm, inspired by the social behaviour of group-forming animal species.

3.2.1. Biological background

The concept of group formation can be observed for many animal species. For some species, e.g. lions and baboons, the group has a social hierarchy with a leader on top. The behaviour of the individuals is then strongly determined by their place at the hierachical ladder. More interesting is the collective behaviour of individuals in decentralized, self-organizing systems. The individuals of these groups have no information about the global behaviour of the group or the environment. Based on local interactions with each other and the environment, the individuals are able to form groups and move together. These local interactions can lead to the development of complex behaviour and the accomplishment of complex objectives, a behaviour that is called swarm behaviour. Examples of such systems in nature are abundant: ant colonies, swarms of birds, schools of fish, *etc.*[27, 28].

Based on simple rules, the members of such swarms are able to move synchronously without collisions (see Figure 3.1). The movement of the total swarm is the result of keeping an optimal distance between the individuals, without following the orders of a leader or guidelines of a global plan. This behaviour can lead to many advantages, in tasks such as the protection against predators or the search for food.

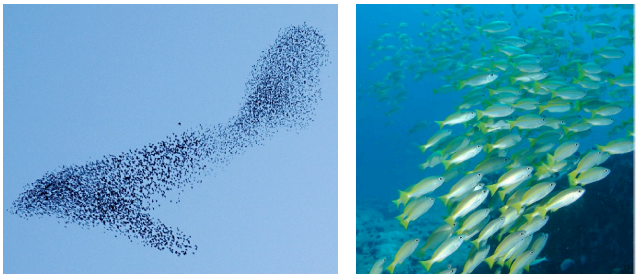
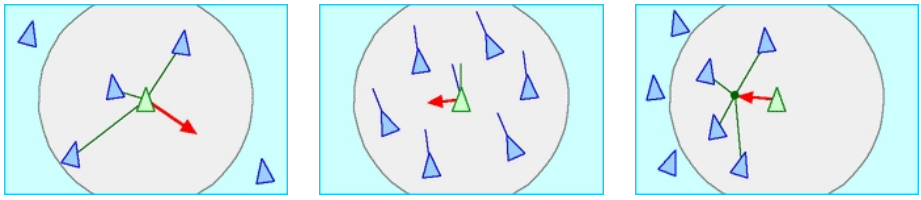


Figure 3.1: Examples of synchronized movements: swarms of birds (left) and schools of fish (right).

3.2.2. The origin of Particle Swarm Optimization

Many scientists have studied the synchronized movements made by different animal groups. In 1986, Reynolds made a computer model, named Boids, of coordinated

animal movements like flocking birds and schooling fish [29]. These simulations showed that the basis of swarm behaviour follows from local interactions, characterized by simple rules. The model consists of 3 simple principles that describe the movement of an individual based on the positions and velocities of the neighbouring individuals. These principles are called separation, alignment and cohesion. Separation moves the individuals away from each other in order to avoid local obstructions and collisions (Figure 3.2(a)). Alignment is responsible for moving the individuals in the same direction as their neighbours. The velocity of the individuals is then adapted to the velocity of their neighbours (Figure 3.2(b)). Cohesion moves the individuals to the center of their neighbours, such that the individuals remain close to the other swarm members (Figure 3.2(c)).



(a) (b) (c)

Figure 3.2: The 3 principles of the computer model Boids: separation (a), alignment (b) and cohesion (c).

Heppner & Grenander independently developed a similar swarm model and added an attractor —a common attraction point for all members of the swarm— to the model. These simulation models have a lot of applications (e.g. games, animation movies, optimization) [30].

Particle Swarm Optimization (PSO) was first introduced by Kennedy for the simulation of social behaviour and was later proposed as an optimization method [28]. Particle Swarm Optimization is an optimization algorithm that categorizes under Swarm Intelligence, which is a general name for artificial intelligence techniques based on the collective behaviour that exists in decentralized, self-organizing systems such as the ones discussed above. All swarm intelligence methods are thus inspired by the social behaviour of insects and other animals. These systems consist of a population of individuals that interact locally with each other and the environment. These interactions can indeed lead to complex behaviour and the accomplishment of certain goals.

3.2.3. Social network structures

PSO is also based on local interactions between the particles of the swarm, namely the particles in the swarm learn from their neighbours and move similarly as their best neighbour. In other words, the successful neighbours have more influence than

the less successful ones. Therefore, the performance of PSO strongly depends on the structure of the social network. More specifically, the information flow through the network depends on the connection between the particles of the network, the amount of clustering and the average shortest distance between two particles. A strong connection leads to a fast flow of information and therefore to a faster convergence. The disadvantage is that a strong connection results in a higher susceptibility to local optima, because the search space is less thoroughly explored than with lesser connected networks. Lesser connected networks with a lot of clusters can have the problem that there will be little information flow between the different clusters [3].

Different network structures are developed for PSO (Figure 3.3). The first structure is the 'Star' (Figure 3.3(a)), in which all particles are connected and each particle can communicate with the other particles. Therefore, each particle is attracted by the best particle of the total swarm. This is also known as the 'global best' PSO. Another network structure is the 'Ring', where each particle communicates with m neighbours, where in most cases $m = 2$. As the information flows slower through the network, a larger part of the search space is explored. However, the convergence will be slower. This structure is also known as the 'local best' PSO.

It is also possible to isolate the particles. One particle then serves as a central point and all information flows through that particle. This particle compares the performance of the particles of the neighbourhood and adapts his position to that of the best neighbour and communicates this position to the other particles. This structure is called the 'Wheel' structure (Figure 3.3(c)). Other examples of social network structures are the 'Pyramid' structure, the 'Four cluster' structure and the 'Vonn-Neumann' structure (Figures 3.3(d), (e) and (f) respectively).

Every network structure allows to define for each particle i in the population a neighbourhood $\mathcal{E}_i$, containing the indices of the particles with which each particle i can communicate. Thus $\mathcal{E}_i$ lists the indices of all the particles that are connected to i by the social network structure.

3.2.4. The algorithm

The PSO algorithm is a population-based metaheuristic, which starts with the initialization of a population of N particles with randomly chosen position and velocity vectors. The position of each particle represents a candidate solution of the optimization problem. In an n-dimensional search space the position and velocity of the ith particle, with $i = 1, \ldots, N$, are denoted by n-dimensional vectors $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ and $\mathbf{v}_i = (v_{i1}, v_{i2}, \ldots, v_{in})$, respectively. In a next step, the objective function f is evaluated for each particle i and this value is assigned as a *fitness* value f_i to the particle. For each particle i a vector $\mathbf{p}_i = (p_{i1}, p_{i2}, \ldots, p_{in})$ is defined that points to the best position that particle i has reached up to this

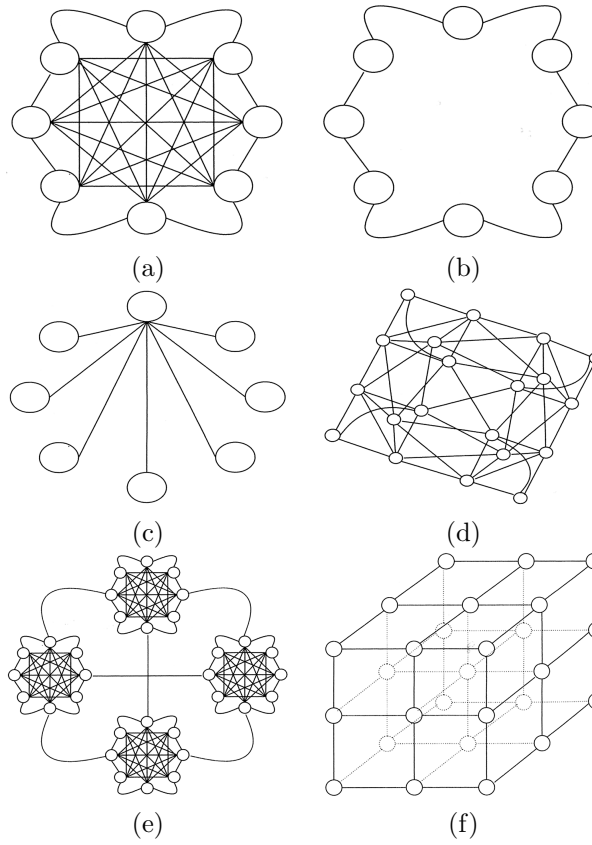


Figure 3.3: Different network structures: (a) star, (b) ring, (c) wheel, (d) pyramid, (e) four clusters and (f) Von-Neumann [3].

point in the iteration cycle; it is also called the personal optimum of this particle. In other words, it is the position with the best fitness value this far in the particle's trajectory. In the neighbourhood $\mathcal{E}_i$ of particle i, the particle that reached the best fitness function value until this point will be identified and labeled g_i. This fitness corresponds to the function value at the position given by the vector $\mathbf{p}_{g_i}$ [3, 31].

At each iteration step (from step k to step $k+1$) the position and velocity of each particle will be updated through the following equations (Figure 3.4):

$$\begin{aligned}\mathbf{v}_i(k+1) = \mathbf{v}_i(k) &+ c_1 \cdot r_1(k) \cdot [\mathbf{p}_i(k) - \mathbf{x}_i(k)] \\ &+ c_2 \cdot r_2(k) \cdot [\mathbf{p}_{g_i}(k) - \mathbf{x}_i(k)] \,, \qquad (3.5)\end{aligned}$$

$$\mathbf{x}_i(k+1) = \mathbf{x}_i(k) + \mathbf{v}_i(k+1) \,. \qquad (3.6)$$

The positive constants c_1 and c_2 are the so-called cognitive and social parameters.

The factors $r_1(k)$, $r_2(k)$ are random numbers between 0 and 1, and are regenerated in each iteration step.

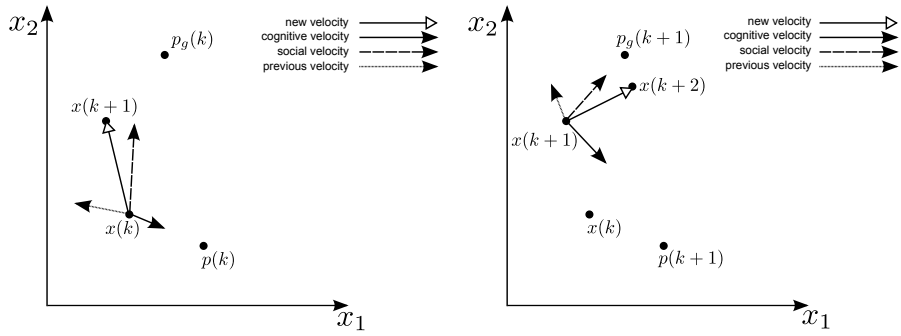


Figure 3.4: Velocity and position update for one particle in two dimensions (left: iteration step j, right: time step $j+1$.

The first part of Eq. (3.5), $\mathbf{v}_i(k)$, is the momentum part that states that the velocity cannot change abruptly, and is based on the current velocity. The second part, $c_1 \cdot r_1(k) \cdot [\mathbf{p}_i(k) - \mathbf{x}_i(k)]$, is the 'cognitive' or personal part, indicating that the particle learns from its own experience and fitness. The effect of this part of the equation is that the particles are attracted to their own best position. The third part, $c_2 \cdot r_2(k) \cdot [\mathbf{p}_{g_i}(k) - \mathbf{x}_i(k)]$, is the social part and represents the cooperation with the other particles of the neighbourhood or the learning from the flying experience of the neighbourhood. The particles are attracted to the globally best position of the neighbourhood. The contribution of the cognitive and social component is weighed by a stochastic quantity $c_1 \cdot r_1(k)$ and $c_2 \cdot r_2(k)$. The update of the position is given by Eq. (3.6).

After the velocity and the position are updated, a new fitness value for the particles is evaluated as $f_i(k+1) = f(\boldsymbol{x}_i(k+1))$, and it is further checked whether $\boldsymbol{p}_i$ needs to be adapted:

$$\mathbf{p}_i(k+1) = \begin{cases} \mathbf{p}_i(k), & \text{if } f_i(k+1) \geq f(\mathbf{p}_i(k))\,, \\ \mathbf{x}_i(k+1), & \text{if } f_i(k+1) < f(\mathbf{p}_i(k))\,. \end{cases} \tag{3.7}$$

The best position within particle i's neighbourhood $\mathcal{E}_i$ is memorized with the index g_i:

$$g_i = \arg\min_{t \in \mathcal{E}_i} \Big[f(\mathbf{p}_t(k+1)) \Big], \tag{3.8}$$

The steps described above are repeated until a certain stopping criterion is met, usually a minimal change in fitness of the best N_{best} particles ($1 \leq N_{\text{best}} \leq N$) or a maximal number of iterations K. The pseudocode of this algorithm can be found

in Algorithm 1.

Algorithm 1: A basic PSO algorithm

```
Data: Parameters
Result: Best solution
Initialize a population of particles at random;
Evaluate the particles of the population;
while stopping criterion is not reached do
    for each particle of the population do
        Calculate new velocity and position;
        Evaluate fitness of each particle;
        Select personal best solution;
        Select global best solution;
```

A disadvantage of updating the velocity as described in Eq. (3.5) is that velocities on the one hand may become too high and therefore cause particles to pass good solutions, or on the other hand may become too small such that the search space will be explored insufficiently. We thus have to control the velocity and the details of the control mechanism will influence the exploration-exploitation ratio. Exploration is the capability to test different regions in the search space in order to have a fair probability of locating the global optimum. Exploitation, on the other hand, is the capability to concentrate the search around a promising candidate solution and precisely locate the optimum. Giving preference to exploration leads to a thorough inspection of the search space and a robust localization of the optimum, with as disadvantage a large number of function evaluations before convergence is obtained. On the other hand, giving preference to exploitation results in a fast and accurate convergence to a possibly local optimum. Therefore, the ratio between these contradictory objectives is very important. In Particle Swarm Optimization, these objectives are directly related to how the velocity is controlled. Different mechanisms exist to control the velocity, namely the application of a maximum velocity $\mathbf{v}_{\max}$ or the introduction of an inertia weight w or of a constriction parameter χ.

Restriction of the velocity

In the first applications of PSO, it was noticed that the velocity can quickly explode, in particular for particles that are located at a large distance from the personal best and the global best position of the neighbourhood. As a consequence, the particles make large changes in position and are located outside the boundaries of the search space. To control the global exploration of the particles, the velocity has to be restricted between certain boundaries. Therefore, a maximum velocity $\mathbf{v}_{\max}$ is determined. If the velocity component $|v_{id}|$ of particle i exceeds the maximum

velocity $v_{d,\max}$ in direction d, the velocity v_{id} is set to $\text{sign}(v_{id})v_{d,\max}$ [3, 31].

The value of $\boldsymbol{v}_{\max}$ is very important as it controls the exploration-exploitation ratio. Large values of $\boldsymbol{v}_{\max}$ facilitate exploration; small values on the other hand favour exploitation. A suitable value of $\boldsymbol{v}_{\max}$ has to be determined, in order to create a balance between exploration and exploitation of the search space. To ensure this balance, a fraction of the size of the domain of the search space is chosen as value of $\boldsymbol{v}_{\max}$:

$$\boldsymbol{v}_{\max} = \delta(\boldsymbol{x}_{\max} - \boldsymbol{x}_{\min}) \tag{3.9}$$

where $\boldsymbol{x}_{\max}$ and $\boldsymbol{x}_{\min}$ contain the lower and upper bound for each component of the search space (*i.e.* $x_{d,\min} \leq x_d \leq x_{d,\max}$) and $\delta \in [0, 1]$. The value of δ is problem dependent.

Restricting the velocity has also disadvantages. Restricting the velocity does not only change the step size of the particles, but also the direction of movement of the particles (Figure 3.5), for example by reducing the maximum velocity in time or by the introduction of an inertia weight w.

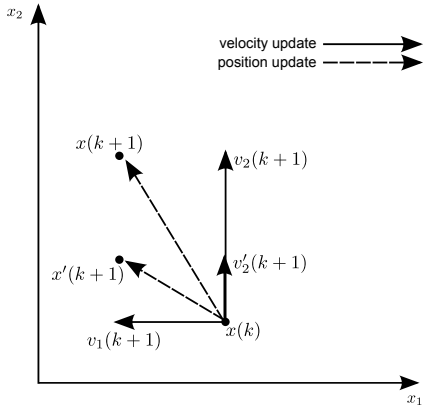


Figure 3.5: Change in direction of movement after velocity restriction.

Inertia weight

The inertia weight w [3, 31] is an additional parameter added to the equation of the velocity update:

$$\begin{aligned}\boldsymbol{v}_i(k+1) = w \cdot \boldsymbol{v}_i(k) + c_1 \cdot r_1(k) \cdot [\boldsymbol{p}_i(k) - \boldsymbol{x}_i(k)] \\ + c_2 \cdot r_2(k) \cdot [\boldsymbol{p}_g(k) - \boldsymbol{x}_i(k)] \ ,\end{aligned} \tag{3.10}$$

with $0 < w < 1$. The inertia weight slows down the velocity of the particle at the previous iteration step and consequently controls the impact of the previous

velocity on the new velocity. In that way, w regulates the ratio between exploration and exploitation. A large (small) w simplifies exploration (exploitation). The ideal value of w leads to a balance between exploration and exploitation and results in a reduction of the number of iterations needed to find the optimal solution.

Constriction parameter

An alternative for the inertia weight w is the constriction parameter χ [3, 31]. The equation of the velocity update is then:

$$\begin{aligned}\boldsymbol{v}_i(k+1) = \chi(k)[\boldsymbol{v}_i(k) + c_1 \cdot r_1(k) \cdot [\boldsymbol{p}_i(k) - \boldsymbol{x}_i(k)] \\ + c_2 \cdot r_2(k) \cdot [\boldsymbol{p}_g(k) - \boldsymbol{x}_i(k)]]\,. \end{aligned} \tag{3.11}$$

The objective of the constriction parameter is also to balance the exploration-exploitation ratio. The difference between w and χ is that a mathematical model is developed for χ, which takes into account the random factors $r_1(k)$ and $r_2(k)$. The following relation appears to be ideal [3, 31]:

$$\chi(k) = \frac{2\kappa}{|2 - \phi(k) - \sqrt{\phi(k)(\phi(k) - 4)}|} \tag{3.12}$$

with $\phi(k) = c_1 r_1(k) + c_2 r_2(k)$. This expression is only properly defined for $\phi(k) \geq 4$. If $\phi(k) < 4$, we set $\phi(k) = 4$, namely $\chi(k) = \kappa$. The coefficient κ lies in the interval $[0, 1]$, small (large) values lead to faster (slower) convergence and little (much) time spend for exploration.

3.2.5. Handling constraints

Another important aspect to be considered is the way of handling the boundaries of the search space and thus of taking possible constraints into account. Two main approaches are available to handle the constraints that limit the search space. The first one exists in including the constraints in the objective function using penalty functions. This can be accomplished using the general techniques that are also used by other algorithms. The second approach, on the other hand, consists of dealing with the constraints and the objective function separately. This approach has some advantages, since no additional parameters need to be introduced in the algorithm, and there is no limit to the number or format of the constraints [32, 33]. This approach can be implemented in several distinct ways. One possibility is to neglect the boundaries and allow particles to be positioned outside the search space. A disadvantage of this approach is that the global best particle can possibly be

located outside the search space. Another approach is to allow particles to cross the boundaries, but to disallow particles outside the boundaries to become the globally best particle. It is also possible to set the velocity of the particle outside the search space equal to zero, or to reinitialize the particles that are located outside the search space. We can also rescale the velocity of the particle outside the search space, such that the particle is positioned on the boundary. Or we can position the particle on the boundary and set the velocity to zero or reflect the velocity. In order to accomplish the reflection of the velocity, the sign of the components of the velocity, in the dimensions where the boundary is crossed, is changed.

3.2.6. Possible stopping criteria

When selecting a certain stopping criterion, two aspects have to be considered:

- PSO should not converge too early, as this would lead to suboptimal solutions.
- For preventing large computational costs, the number of function evaluations should not be too high. There should be a lot of evaluations in areas where the fitness is low and few evaluations in areas where the fitness is high.

The following stopping criteria, each having their own disadvantage, are possible:

- A maximal number of iterations: when the maximal number of iterations is too small, the algorithm will stop before the optimal solution is reached.
- An acceptable solution: the algorithm is stopped when the error is lower than a predetermined value ϵ. However, for this criterion, we need information about the optimum.
- Lack of improvement: The disadvantage of this stopping criterion is that it is not always easy to find an objective quantity and corresponding tolerance level that measures the lack of improvement. Improvement can be measured in different ways. When the mean change in position of the particles is small, we can assume that the swarm has converged. An alternative is to stop the algorithm when the velocity of the particles is close to zero, then the change in position will also be minimal.

3.2.7. Description of the parameters

As for all heuristic optimization algorithms, the performance of the algorithm largely depends on the choice of the parameter values. The parameters that have to be determined are the population size N, the cognitive and social parameters c_1 and c_2, the size of the neighbourhood, the number of iterations K.

The size of the population determines the initial diversity of the swarm. When the particles are uniformly initialized, the diversity will increase when a higher number of particles is present. A larger swarm will explore a larger part of the search space per iteration and less iterations will be necessary to find a good solution. Too large populations will lead to a larger computational cost and the particles will degrade to a parallel random search. In literature [3], it is shown that with a population size of 10 to 30 particles, the optimal solution is often found for a wide range of problems. Nevertheless, the perfect population size is problem dependent.

The size of the neighbourhood is responsible for the range of the social interactions in the swarm. In smaller neighbourhoods there will be less social interactions than in large neighbourhoods, therefore convergence is slower in small neighbourhoods than in large neighbourhoods. Although there is a slower convergence, the convergence is more reliable in small neighbourhoods and the particles are less susceptible to local optima.

The number of iterations needed to obtain a good solution is problem dependent. A balance has to be made between the computational cost and the necessary accuracy of the solution.

Together with the random parameters r_1 and r_2, the cognitive c_1 and social c_2 parameters control the stochastic influence of the cognitive and social components of the velocity of the particle. In other words, these parameters determine the relative influence of the personal and global optimum. The value of these parameters is problem dependent. When only $c_1 > 0$, the particles can be interpreted as independent hill climbers, namely each particle searches for the best position in his neighbourhood. When only $c_2 > 0$, the total swarm is attracted to one point. When c_1 and c_2 are small, the particles are only weakly influenced by the personal and global best position and are allowed to explore the complete search space. On the other hand, large values of c_1 and c_2 cause an increase in velocity with abrupt movements to better regions.

3.3. Simplex-Simulated Annealing

The Simplex-Simulated Annealing (SIMPSA) algorithm [34] is an optimization algorithm based on a combination of the nonlinear Simplex algorithm [35] and the Simulated Annealing algorithm [36]. In this section, we will briefly describe the nonlinear Simplex algorithm and the Simulated Annealing algorithm in order to then present the Simplex-Simulated Annealing algorithm.

3.3.1. Nonlinear Simplex

The nonlinear Simplex algorithm is a derivative-free algorithm for finding the local minimum of the function f in the neighbourhood of a randomly chosen starting point $\boldsymbol{x_0} = (x_{0,1}, \ldots, x_{0,n})$ around which a simplex, *i.e.* a polytope determined by $n+1$ vertices, is created. The n vertices $\boldsymbol{x}_i$ $(i = 1, \ldots, n)$ are created by shifting $\boldsymbol{x}$ along each of the coordinate axes separately. The function values $f_i = f(\boldsymbol{x}_i)$ at the vertices of the simplex, $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_n$, are compared and the simplex is encouraged to move away from the worst vertex. The latter is performed by evolving the simplex at each iteration through reflections, expansions and contractions in one or all directions. More specifically, let us assume that the vertices are ordered such that $\boldsymbol{x}_0$ is the best vertex and $\boldsymbol{x}_n$ is the worst vertex. The aim is to define a new simplex by moving $\boldsymbol{x}_n$. Let us define $\boldsymbol{x}_\mathrm{r}$ as the reflection of $\boldsymbol{x}_n$ through the centroid $\boldsymbol{x}_\mathrm{o} = \sum_{i=0}^{n-1} \boldsymbol{x}_i/n$ of the remaining vertices:

$$\boldsymbol{x}_\mathrm{r} = \boldsymbol{x}_\mathrm{o} - (\boldsymbol{x}_n - \boldsymbol{x}_\mathrm{o}) = 2\boldsymbol{x}_\mathrm{o} - \boldsymbol{x}_n \,. \tag{3.13}$$

If $f_0 \leq f(\boldsymbol{x}_\mathrm{r}) < f_{n-1}$, then we replace $\boldsymbol{x}_n$ by $\boldsymbol{x}_\mathrm{r}$. If $f(\boldsymbol{x}_\mathrm{r}) < f_0$, then we can hope to find an even better solution by expanding $\boldsymbol{x}_\mathrm{r}$ to $\boldsymbol{x}_\mathrm{e}$ which is given by

$$\boldsymbol{x}_\mathrm{e} = \boldsymbol{x}_\mathrm{o} - \gamma(\boldsymbol{x}_n - \boldsymbol{x}_\mathrm{o}) = (1+\gamma)\boldsymbol{x}_\mathrm{o} - \gamma\boldsymbol{x}_n \,, \tag{3.14}$$

with $\gamma > 1$ ($\gamma = 2$ in a typical implementation). If $f(\boldsymbol{x}_\mathrm{e}) < f(\boldsymbol{x}_\mathrm{r})$, then we replace $\boldsymbol{x}_n$ by the expanded point $\boldsymbol{x}_\mathrm{e}$, else by the reflected point $\boldsymbol{x}_\mathrm{r}$. This was under the assumption that $f(\boldsymbol{x}_\mathrm{r}) < f_{n-1}$. If $f(\boldsymbol{x}_\mathrm{r}) > f_{n-1}$, then we will try to contract by computing $\boldsymbol{x}_\mathrm{c}$ as

$$\boldsymbol{x}_\mathrm{c} = \boldsymbol{x}_\mathrm{o} + \rho(\boldsymbol{x}_n - \boldsymbol{x}_\mathrm{o}) = (1-\rho)\boldsymbol{x}_\mathrm{o} + \rho\boldsymbol{x}_n \,, \tag{3.15}$$

with $\rho < 1$ ($\rho = 1/2$ in a typical implementation). If $f(\boldsymbol{x}_\mathrm{c}) < f_n$, then we replace $\boldsymbol{x}_n$ by $\boldsymbol{x}_\mathrm{c}$. If not, we decide to reduce the whole simplex by setting $\boldsymbol{x}_i = \boldsymbol{x}_0 + \sigma(\boldsymbol{x}_i - \boldsymbol{x}_0)$ for all $i = 1, \ldots, n$ (thus for all points but the best) with $\sigma < 1$ ($\sigma = 1/2$ in a typical implementation). Convergence is stopped if the edges of the simplex are shorter than a given tolerance value. As the simplex method uses downhill movements in case of a minimization problem, it is a local optimization algorithm.

3.3.2. Simulated Annealing

In contrast to the nonlinear Simplex algorithm, the Simulated Annealing algorithm [36, 37] is a global optimization algorithm based on the physical thermal process of annealing. When a metal is first melted by heating it, and then slowly cooled down, it will eventually be frozen in a minimal energy state. Thermal equilibrium at a given temperature is characterized by a Boltzmann distribution function of

the energy states $P(E) \sim \exp(-E/KT)$. At thermal equilibrium the energy is probabilistically distributed among all different energy states E. This probability distribution shows that the system can go uphill as well as downhill, but the lower the temperature, the smaller the probability of going uphill. These ideas are captured by Metropolis (1953) [37] into numerical equations to search for the minimum of a function. Before starting the algorithm, some parameters have to be initialized, namely the initial temperature ($T_{\max}$), the number of iterations (K), the cooling ratio (τ) and the freezing temperature ($T_{\min}$). If the current solution after k iterations is given by $\boldsymbol{x}(k) = (x_1(k), \ldots, x_n(k))$, then, at iteration $k+1$, a point $\boldsymbol{x}(k+1) = (x_1(k+1), \ldots, x_n(k+1))$ is randomly selected in the neighbourhood of $\boldsymbol{x}(k)$, and accepted with probability $p(k+1)$, which is given by:

$$p(k+1) = \begin{cases} \exp\left(\frac{-(f(\boldsymbol{x}(k+1)) - f(\boldsymbol{x}(k))}{T(k+1)}\right), & \text{if } f(\boldsymbol{x}(k+1)) > f(\boldsymbol{x}(k)) \\ 1, & \text{if } f(\boldsymbol{x}(k+1)) < f(\boldsymbol{x}(k)) \end{cases} \tag{3.16}$$

with $T(k+1)$ the current system temperature.

3.3.3. Simplex-Simulated Annealing

Simulated Annealing was initially developed as a discrete optimization algorithm. In case of continuous optimization problems, we need a strategy to generate next possible solutions in the neighbourhood of the current solution. This leads to the development of the Simplex-Simulated Annealing (SIMPSA) algorithm. In the SIMPSA algorithm, the nonlinear Simplex algorithm and the Simulated Annealing algorithm are combined. The Simplex algorithm is used to determine the next possible solution of the problem. Simulated Annealing is used to allow wrong-way movements [34]. The same parameters as for Simulated Annealing are required, except for the initial annealing temperature, which is optimized by the algorithm itself. The ideal value for the initial temperature $T(k)$ is estimated to be solving:

$$\zeta = \frac{m_1 + m_2 \exp(\frac{-\triangle f^+}{T(k)})}{m_1 + m_2}, \tag{3.17}$$

where the acceptance ratio ζ is typically set to 95%, $m_0 = 100 \cdot n$, m_1 represents the number of successful moves, m_2 represents the number of unsuccessful moves and $\triangle f^+$ the average increase in cost for the m_2 unsuccessful moves. In order to apply Eq. (3.17), a preliminary high temperature is estimated by multiplying the absolute value of the objective function corresponding to the starting solution by a large positive value. As cooling schedule, the Aarst and van Laarhoven [38] scheme is used:

$$T(k+1) = \tau \cdot T(k) = \frac{T(k)}{1 + \frac{T(k)\ln(1+\delta)}{3\sigma}}\,, \tag{3.18}$$

with k the current iteration, τ the cooling ratio, δ the parameter that controls the cooling rate and σ the standard deviation of all cost configurations at the current temperature. In the original Simulated Annealing algorithm, the cooling schedule is only after a large number of iterations enforced, this because equilibrium has to be reached at each temperature level. However, to reduce the computational cost, it is also possible to apply the cooling schedule at the moment an improved solution is obtained. This adaptation is applied in the SIMPSA algorithm and is known as the non-equilibrium variant of the SIMPSA algorithm.

The first step of the SIMSPA algorithm consists of generating the initial simplex. For unconstrained optimization problems, this simplex is generated as follows:

$$\boldsymbol{x}_i = \boldsymbol{x}_0 + (0.5 - r)2|x_{0,i}|\boldsymbol{e}_i\,, \tag{3.19}$$

with $\boldsymbol{x}_0$ a random point in the search space, r a random number between 0 and 1 and $\boldsymbol{e}_i$ the unit vector in direction i. For constrained problems, the initial simplex is generated by

$$\boldsymbol{x}_i = \boldsymbol{x}_0 + (0.5 - r)K(k) \cdot (x_{\min,i} - x_{\max,i})\boldsymbol{e}_i\,, \tag{3.20}$$

with $\boldsymbol{x}_{\min}$ and $\boldsymbol{x}_{\max}$ the boundary constraints, $K(k)$ a variable factor and k the current global iteration. Important to note is that Eq. (3.20) only guarantees feasibility when the initial solution vector $\boldsymbol{x}_0$ obeys all boundary constraints.

In order to incorporate the Simulated Annealing property of allowing wrong way movements, in case of a minimization problem, the function values of the vertices of the simplex are randomly perturbed proportional to the temperature control parameter by adding a positive logarithmically distributed value. In the same way, from the function value of every new replacement point $\boldsymbol{x}_h^*$ a similar random quantity is subtracted. This is presented by following equations:

$$f(\boldsymbol{x}_i)_{\text{perturbed}} = f(\boldsymbol{x}_i) - T\ln(r)\,; \qquad i = 1, \ldots, n+1 \tag{3.21}$$

$$f(\boldsymbol{x}_h^*)_{\text{perturbed}} = f(\boldsymbol{x}_h^*) + T\ln(r)\,, \tag{3.22}$$

with $f(\boldsymbol{x}_i)$ the function value for vertex i, $f(\boldsymbol{x}_h^*)$ the function value of the replacement point $\boldsymbol{x}_h$ and $f(\boldsymbol{x})_{\text{perturbed}}$ the perturbed function value and T the temperature control parameter. Then, the original simplex algorithm is performed on these perturbed function values.

3.3.4. Handling constraints

During the run of the algorithm, we also have to deal with constraints. This is done by replacing the points that do not satisfy the constraints by randomly generated points centered around the current best vertex by using Eq. (3.20), with $\boldsymbol{x}_0$ denoted as the best vertex. It is still possible that infeasible points are replaced by infeasible points, but now they are centered around the best vertex and can be forced to obey all boundary constraints. All infeasible points are then penalized by assuming a very large positive value in case of a minimization problem. If all points in the simplex are penalized, it is still possible to make a quantitative comparison between these points, by making the random perturbation proportional to the temperature control parameter presented in Eqs. (3.21) and (3.22) [34].

3.3.5. Possible stopping criteria

The next aspect to be discussed is the termination criterion of the SIMPSA algorithm. The algorithm includes two convergence tests. The first one is inherent to the simplex method [39] and is a measure of collapse of the centroid. The second one is based on an averaged gradient of the objective function with respect to the number of function evaluations. When a maximum number of iterations is reached or when the change in the objective value is less than a predefined tolerance, SIMPSA is stopped [34].

3.3.6. Description of the parameters

The performance of this algorithm also depends on the choice of the parameter values, which are problem dependent. The parameters that have to be chosen are the cooling ratio τ, the freezing temperature $T_{\min}$ and a maximum number of iterations. In the beginning of the algorithm, the initial temperature T, determined by the algorithm itself, will be high. This means that exploration is favoured with respect to exploitation. During the algorithm, this temperature will decrease, which will change the exploration-exploitation ratio. At the end, at the minimum temperature $T_{\min}$ exploitation is favoured with respect to exploration. The cooling ratio determines how fast the algorithm makes the transition of exploration to exploitation. A too small cooling ratio τ will result in a faster convergence, but the algorithm is then more susceptible to local optima. On the other hand, a too large cooling ratio τ will result in a slow convergence, with the risque of overstepping the global optimum. The freezing temperature $T_{\min}$ determines the exploration-exploitation ratio at the end of the algorithm. As at the end of the algorithm, exploitation is important for the convergence, a small freezing temperature $T_{\min}$ is preferred. For the maximum number of iterations a balance has to be made between the computational cost and the required accuracy of the solution.

4

Combinatorial optimization methods

As mentioned in Chapter 2, combinatorial optimization problems are a class of discrete optimization problems, where the input arguments encode permutations, combinations or variations. In case this type of optimization problem is NP-hard, we have to rely on metaheuristics, to which we refer in this setting as combinatorial optimization. In this chapter, two combinatorial optimization algorithms are presented. As in Chapter 3, we restrict ourselves to the algorithms that are used in further chapters of this dissertation (in particular to the problem of subset selection in Part II), namely Genetic Algorithms and Ant Colony Systems. Both algorithms are metaheuristics and were implemented to fit our needs. Nevertheless, we here discuss the general version of these algorithms and postpone the discussion of our modifications to the relevant part (Part II).

4.1. Genetic Algorithms

Genetic algorithms belong to the class of evolutionary algorithms, a particular kind of nature-based algorithms. Evolutionary algorithms use concepts inspired by evolutionary biology such as mutation, selection and crossover. First, we will discuss the biological background of Genetic Algorithms.

4.1.1. Biological background

Living organisms consist of one or more cells. Each cell contains genetic material, which is called the genome. More specifically, each cell is composed of a set of chromosomes consisting of different genes, representing blocks of DNA. Each gene encodes a trait —possible settings for a trait are called alleles— and has its own position on the chromosome, which is also known as the locus.

The particular set of genes possessed by an individual is called the genotype. The characteristics and qualities of an individual are known as the phenotype of the individual.

When reproductions (recombinations) between organisms occur, a combination of genes of the parents leads to the genes of the children, a process which is defined

as crossover. During the copying of these genes errors can occur, *i.e.* it is possible that the genes are slightly changed, which is called mutation. This mutation is responsible for the preservation of the diversity in the population.

In nature, the reproduction of parents is controlled by survival of the fittest. Fit individuals have more possibilities for food, water and partners. Therefore, fit individuals have higher reproduction possibilities. These concepts are also included in Genetic Algorithms [40].

4.1.2. The algorithm

A genetic algorithm is a population-based optimization algorithm. This algorithm starts at $k = 0$ with an initial population $P(k)$ of N chromosomes. Each chromosome i ($i = 1, \ldots, N$) encodes a candidate solution $\boldsymbol{x}_i$ of the optimization problem and is assessed by the objective function f, resulting in a fitness $f_i = F(f(\boldsymbol{x}_i))$, where F is any monotonically decreasing function. A good choice of F will depend on the range of function values $f(\boldsymbol{x})$. The optimization process runs through a number of iterations K which are appropriately called generations. In each generation a new population $P(k)$ is created, according to the following recipe. Firstly, a population of chromosomes is initialized. Secondly, the chromosomes are evaluated by the objective function. Thirdly, parents are chosen using some selection procedure. Fourthly, children are created through crossover and mutation of the selected parents. These children constitute the new population. These steps are repeated until a stopping criterion is reached, e.g. a maximal number of generations. [41, 40]. The pseudocode of the basic algorithm can be found in Algorithm 2.

Algorithm 2: A basic Genetic Algorithm

```
Output: Best solution
k ← 0;
Initialize population P(0) at random;
while stopping criterion is not reached do
    foreach individual in P(k) do
        Evaluate the chromosome;
    Select parents through some selection procedure;
    Reproduce children through crossover and mutation;
    k ← k + 1 ;
```

4.1.3. Representation

Depending on the problem, the chromosomes can encode possible solutions $\boldsymbol{x}$ in the search space $\mathcal{S}$ in different ways depending on the type of problem and the

structure of the solution space. The binary representation is the first developed representation and is ideal for problems that can be formulated in terms of a set of boolean decision variables such as subset selection problems. The chromosome is then a string of binary values, which is also called a bit-string. The length of the chromosome is problem dependent. In this representation, all possible bit-strings have to represent valid solutions and it is important that all possible solutions can be represented as a bit-string. The permutation representation is used for order based problems. For discrete problems that are not of combinatorial nature, there is also an integer representation where the genes can take values out of a set integers. And finally, Genetic Algorithms can be applied to continuous optimization using the real space representation where the genes can take values in a real interval. We will restrict the further discussion to the binary representation, as this is the optimal choice for the applications in this dissertation and in a wide range of problems for which Genetic Algorithms performs well [40].

4.1.4. Parent selection

For the creation of the children, we have to select parents from the population. Different parent selection procedures exist, such as fitness-based selection, rank-based selection and tournament selection [41, 40].

In fitness-based parent selection, the probability of selecting individual i is $\frac{f_i}{\sum_{g=1}^{N} f_g}$. These probabilities can be plotted on a roulette wheel, where each probability is then presented by a segment of the roulette wheel (Figure 4.1). In this way, a random selection is made similar to how the roulette wheel is rotated. The parent that is selected by rotating the wheel corresponds to the parent where the pointer of the wheel ends. In order to select N parents, the roulette wheel is rotated N times independently.

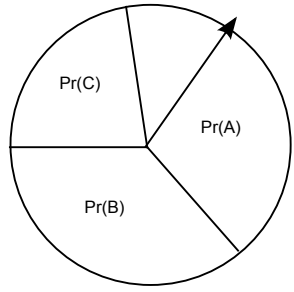


Figure 4.1: Fitness-based parent selection

The advantage of this representation is its simplicity. Better individuals are strongly favoured and take over the population very quickly, which can often be a drawback. When the difference in fitness between the individuals is rather small, the selection

is more or less random. The biggest disadvantage of this strategy is that the roulette wheel is used N times independently, resulting in a high variance in the number of children attributed to each individual. A solution is provided by 'Stochastic Universal Sampling', where N proportionally distributed pointers are plotted on the roulette wheel and the wheel is then used only one time (Figure 4.2). In this way, the N parents are simultaneously selected.

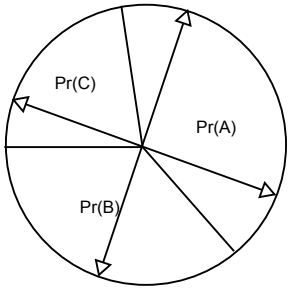


Figure 4.2: Stochastic Universal Sampling

Rank-based selection is based on the disadvantages of the fitness-based selection procedure. Firstly, the individuals of the population are sorted according to their fitness. Secondly, selection probabilities are calculated based on the rank of the individuals. Then, the same principle is used as in the fitness-based selection procedure.

A completely different strategy is to use tournament selection. Of two randomly selected chromosomes, the one with the highest fitness is kept as a suitable parent. This process is repeated until the number of parents equals the number of chromosomes in the population. Not every chromosome will be a parent, and some chromosomes will act as parent more than once. The advantages of this procedure are its simplicity and the invariance under rescaling of the fitness. However, the high variance in the number of children attributed to each individual is sometimes disadvantageous.

4.1.5. Recombination

Recombination consists of crossover and mutation [40]. Crossover is the genetic operator that performs the recombination of the genes of the parents to produce the children. First, a crossover probability r_c is determined. Then, for each couple of parents a random number between 0 and 1 is created. If this random number is smaller than or equal to the crossover probability, the children are created through crossover of the parents. On the other hand, if this random number is larger than the crossover probability, the children are exact copies of the parents. There exist different crossover operators, the most important ones are: one-point crossover, M-point crossover and uniform crossover. In one-point crossover, two parents are

randomly chosen. Then, a random point of the chromosome is chosen as breaking point and the tails of the parents are exchanged. The newly formed chromosomes are the children.

In M-point crossover, two parents are randomly chosen. The chromosomes of the parents are broken in M segments and the children are created by alternatingly taking parts of the two parents. In uniform crossover, again two parents are randomly chosen. In this crossover procedure, the genes are treated separately. More specifically, each gene is randomly chosen from one of the two parents

Mutation is a random change in the genes of the chromosomes. First, the mutation probability r_m has to be determined. For each gene of the chromosome, a random number between 0 and 1 is created. If the random number is smaller than or equal to the mutation probability, the corresponding gene is mutated, which will correspond to a bit flip in the binary representation. If the random number is larger than the mutation probability, the original value of that gene is kept.

4.1.6. Handling constraints

There are three standard methods to deal with constraints and infeasible solutions in GA [41, 42]. The best solution is of course to use a representation that ensures that all solutions are feasible. When this is practically impossible, one can try to design a repair operator which guarantees the transformation of an infeasible solution to a feasible solution. If neither of these are possible, one can resort to the application of a penalty function to penalize the fitness of an infeasible solution without distorting the fitness landscape. In the second part of this dissertation, we will discuss a modification of Genetic Algorithm's standard crossover and mutation operators in order to transform parents that satisfy a specific constraint into children that satisfy this same constraint.

4.1.7. Description of the parameters

Before the algorithm can be used, different parameters have to be determined. These parameters, *i.e* the population size N, the crossover probability r_c, the mutation probability r_m and the number of generations K, are problem dependent. The larger the population size the larger the explored part of the search space. Too large populations, however, will lead to a large computational cost. The crossover probability is a measure of interaction between the parents, a large interaction will lead to a larger exploration-exploitation ratio. Mutation is responsible for keeping the diversity in the population. However, as we want the algorithm to converge, we do not want too much randomness in the population, and the mutation probability should not be too high. For the number of generations, we have to make a balance between the necessary accuracy and the computational burden [40].

4.2. Ant Colony Systems

Just as Particle Swarm Optimization (Section 3.2), Ant Colony Optimization (ACO) falls under swarm intelligence. ACO is a metaheuristic algorithm and takes inspiration from the foraging behaviour of some ant species, which deposit pheromones on the ground in order to indicate favourable paths that should be followed by the other members of the colony. A similar mechanism is used to solve optimization problems with ACO.

4.2.1. Biological background

A french entomologist Pierre-Paul Grassé [43, 44] discovered that some species of termites react to certain stimuli. These reactions act as new significant stimuli for both the insect that produced them and the other insects in the colony. This type of communication is described by the term stigmergy, which is an indirect, non-symbolic form of communication passed on by the environment and can only be accessed by the insects that visit the place were the information was released. Stigmergy can also be observed in some colonies of ants. The ants walking to and from the food source deposit on the ground a substance called pheromone. The other members of the colony observe the presence of the pheromones and tend to follow the paths where the pheromone concentration is higher. The pheromone depositing and following behaviour of ants is investigated thoroughly with experiments such as the 'double bridge' [45].

ACO was initially proposed by Colorni, Dorigo and Maniezzo [46] as an optimization algorithm to solve combinatorial optimization problems. In ACO, a number of artificial ants build solutions to the considered optimization problem and exchange information on the quality of these solutions through a communication strategy that is similar to the one used by real ants. The original ACO algorithm is known as Ant Systems (AS), which was then followed by a number of different algorithmic variants that tried to improve the performance of the AS algorithm.

4.2.2. The algorithm

ACO is a population-based metaheuristic inspired by the behaviour of real ants. It is applicable to problems where the set of possible solutions $\mathcal{S}$ can be mapped to a graph. The vertices of the graph represent parts or components of the solution, and the edges of the graph represent the path that the ants can follow in order to construct a solution. ACO is thus a constructive metaheuristic, where the artificial ants will extend an initially empty partial solution sequence by adding solution components to it. The addition of solution components will continue until the solution sequence is complete. This is called the construction phase. After

all ants within a single iteration have completed their solution sequence $\boldsymbol{x}$, the pheromones are updated. These principles are the same for all ACO variants. Initially ACO was applied to real problems in graph theory, such as the traveling salesman problem [47]. Pheromones are deposited on either vertices or edges, and a higher concentration of pheromones at a certain vertex (edge) increases the probability for this vertex (edge) to be part of a solution in the next iteration. The concentration of pheromones is thus chosen inversely proportional to the objective function that has to be minimized, which is typically related to the total path length in graph problems.

However, the ACO metaheuristic can be generalized to any combinatorial optimization problem for which a constructive heuristic can be defined. In particular, any problem that is formulated in terms of n discrete decision variables X_d $(d = 1, \ldots, n)$ can be studied using ACO. In such problems, every decision variable X_d can take a value from a set of m_d possible values $\mathcal{X}_d = \{x_d^1, \ldots, x_d^{m_d}\}$. The construction of solutions thus boils down to the successive assignment of values $x_d^{j_d}$ to the decision variables X_d, with $j_d \in \{1, \ldots, m_d\}$. To every possible assignment $X_d = x_d^{j_d}$, we associate a vertex $v_d^{j_d}$ of a graph. The construction of a solution is then mapped to a directed graph problem by connecting the vertex $v_d^{j_d}$ for every $j_d = 1, \ldots, m_d$ to every vertex $v_{d+1}^{j_{d+1}}$ for every $j_{d+1} = 1, \ldots, m_{d+1}$ by a unidirectional edge, as illustrated in Figure 4.3. This scheme to map the problem of decision variables to a graph is not unique, as it depends on the order in which a value is assigned to the different decision variables. The consequences of this observation are discussed in the next section.

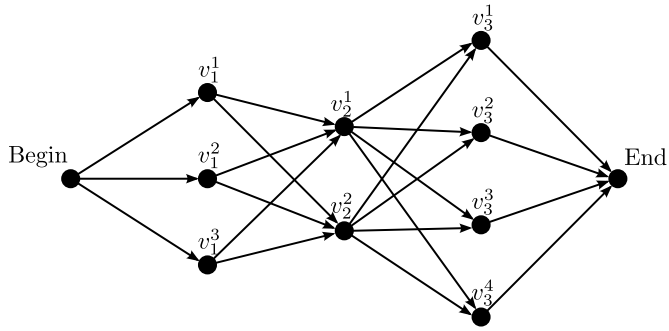


Figure 4.3: Directed graph representing the possible decisions the ants can make during the construction of solutions.

In the construction phase, the ants construct solutions by each traversing the graph from 'Begin' to 'End' along the directed edges. The partial solutions $\boldsymbol{x}^p$ encoded by the ants at 'Begin' are empty ($\boldsymbol{x}^p = \langle\rangle$). In every step d, a solution component $X_d = x_d^{j_d}$ is added to the partial solution $\boldsymbol{x}^p$, depending on which vertex $v_d^{j_d}$ the ant passes. To decide to which vertex $v_d^{j_d}$ with $j_d = 1, \ldots, m_d$ the ants traverse, they use a probabilistic model depending on the pheromone concentration along the path.

For this kind of problem, one typically adopts an algorithm where the ants deposit pheromones on the vertices and not on the edges. We thus define a pheromone trail parameter $\tau_d^{j_d}$ assigned to every vertex $v_d^{j_d}$. The probability of adding a solution component $X_d = x_d^{j_d}$ to a partial solution $\boldsymbol{x}^p$ is given by [18]:

$$p(x_d^{j_d} \mid \boldsymbol{x}^p) = \frac{[\tau_d^{j_d}]^\alpha [\eta(x_d^{j_d})]^\beta}{\sum\limits_{x_d^t \in R(\boldsymbol{x}^p)} [\tau_d^t]^\alpha [\eta(x_d^t)]^\beta}\,, \qquad \text{for all} \quad x_d^{j_d} \in R(\boldsymbol{x}^p)\,. \tag{4.1}$$

Here $\eta(x_d^{j_d})$ represents the heuristic information, which is a simple measure of the *a priori* desirability of adding this component given the current partial solution. In most applications, this value does not change during the run of the algorithm. $R(\boldsymbol{x}^p)$ is the set of possible solution components that can be used to extend the partial solution $\boldsymbol{x}^p$. Typically, this set contains all values $x_d^{j_d}$ for $j_d = 1, \ldots, m_d$, unless some values $x_d^{j_d}$ can no longer be assigned to X_d due to constraints based on previous assignments in the partial solution. The exponents α and β can be used to control the influence of the *a priori* heuristic information and of the pheromone concentration.

When the construction phase is finished and all ants have constructed a complete solution, each ant will individually update the pheromone concentration depending on the fitness of its solution. Then, a new iteration is started. Initially, all pheromone trail parameters $\tau_d^{j_d}(1)$ in the first iteration are chosen equal, which should correspond to having an equal probability for all solutions. However, as discussed in the next section, this is not always the case. Let us assume that there are N ants within iteration k and that ant i has constructed a solution $\boldsymbol{x}_i = (x_1^{j_{1,i}}, \ldots, x_n^{j_{n,i}})$ with fitness $f(\boldsymbol{x}_i)$. Every ant i deposits a number of pheromones to the vertices $v_d^{j_{d,i}}$ along which it has traversed the graph, that is given by $F(f(\boldsymbol{x}_i))$, where f is the objective function that has to be minimized, and F is any monotonically decreasing function. A good choice for F will depend on the range of function values $f(\boldsymbol{x})$ of the objective function. The total update of the pheromene trail parameters $\tau_d^{j_d}$ from iteration k to iteration $k+1$ is then given by [18]:

$$\tau_d^{j_d}(k+1) = (1-\rho)\,\tau_d^{j_d}(k) + \rho \sum_{\{i \mid x_d^{j_{d,i}} = x_d^{j_d}\}} F(f(\boldsymbol{x}_i))\,, \tag{4.2}$$

for all $j_d = 1, \ldots, m_d$ and $d = 1, \ldots, n$. In this equation, ρ represents the evaporation rate and models that pheromones fade away over time. This evaporation is required for the initial pheromone concentration along bad paths to decay, so that in the end all pheromones are distributed along good paths and the ants converge to the best solution. Equation (4.2) illustrates that solution components $x_d^{j_d}$ belonging to solutions $\boldsymbol{x}_i$ with lower values of the objective function, will receive higher pheromone updates. The pseudocode of the AS algorithm can be found in

Algorithm 3.

Algorithm 3: A basic ACO algorithm

```
Set parameters;
Initialize pheromones(T);
while no convergence do
    Construct solutions x ;
    Update pheromones τ;
```

4.2.3. Representations and bias

In this section we discuss in more detail the construction and representation of solutions to combinatorial optimization problems, and the possible pitfalls of using these representations in combination with the ACO paradigm. ACO was originally developed for the traveling salesman problem [47], where one is interested in the optimal permutation of a set of n elements $\mathcal{I} = \{I_1, \ldots, I_n\}$, such that e.g. the total travel distance is minimized if the elements I_d for $d = 1, \ldots, n$ represent different cities that a salesman has to travel to. Following the general scheme of the previous section, we can identify the decision variable X_d in step d with the next item to select. Hence, every decision variable X_d can take a value in the set $\mathcal{X}_d = \mathcal{I} = \{I_1, \ldots, I_n\}$, with of course the additional constraint that no item can be selected that was already selected before and is thus already present in the partial solution $\boldsymbol{x}^p$. To every possible solution of the problem corresponds a unique path that can be taken by the ants and no bias is present. Note that we can represent this construction as a directed graph using the general construction of the previous paragraph, but that we can also associate it to the undirected graph in Figure (4.4).

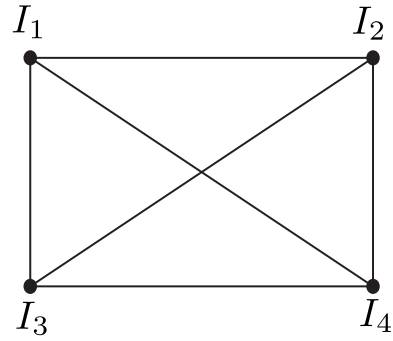


Figure 4.4: Undirected graph representing the construction of permutations from the set $I = \{I_1, \ldots, I_4\}$. In every step of the construction, ants can travel from their current position to every other element of the set that they have not visited before. This additional constraint cannot be expressed on the graph.

The same construction of solutions can be applied to subset selection problems. Possible examples include *knapsack* problems [48, 49, 50]. A knapsack problem refers to the problem of maximizing the value of a bag (knapsack) by filling the bag with a number of samples, without exceeding the maximum weight of the knapsack. For example, in the 0/1 dimensional knapsack problem, each item I_d of the set I has a value z_i and a weight w_i. The problem is then to select a subset of items which maximizes the sum of the values of the items and does not exceed a given weight constraint. Another example of a subset selection problem will be studied in the second part of this dissertation. In subset selection problems, we are interested in the optimal combination (without repetition) of elements from the set $\mathcal{I} = \{I_1, \ldots, I_n\}$ instead of in a permutation of these elements. Hence, we do not necessarily need to select all elements, and the order in which the elements are selected does not matter for the value of the objective function. As before, we can construct solutions by identifying X_d with the next item to select. Unlike in the traveling salesman problem, a solution sequence can be complete before all n items have been selected. The construction will stop when no more items can be selected without violating the constraints or without increasing the objective function or when all items have been selected. This representation of solutions is therefore called the variable length representation (VLR). This representation was proposed by Hinterding [51]. In the VLR, each item of the set $I_d \in I$ is assigned one pheromone trail parameter τ_d and a heuristic information parameter η_d. The probabilistic decision rule (Eq. (4.1)) for adding an item I_d to a partial solution $\boldsymbol{x}^p$ is then replaced by:

$$p(I_d \mid \boldsymbol{x}^p) = \frac{[\tau_d]^\alpha [\eta_d]^\beta}{\sum\limits_{I_l \in R(\boldsymbol{x}^p)} [\tau_l]^\alpha [\eta_l]^\beta}\,, \qquad \text{for all } I_d \in R(\boldsymbol{x}^p)\,. \tag{4.3}$$

However, in the VLR a representation bias is present [48, 52]. Since the order of selection is now unimportant, every solution corresponds to different paths that can be taken by the ants, with the number of paths equal to the number of permutations of the elements in the subset. When solutions with a different number of selected items coexist, the solutions with a larger number of selected items will be over-represented in the search space, since more permutations are possible. Because of this bias, larger sequences will be favoured independently of the fitness values of these solutions.

A second representation for the subset selection problem is the bit string representation (BSR) (Figure 4.5) [53], which corresponds to how solutions are encoded in the chromosomes of GA [54]. This time we have exactly n decision variables X_d $(d = 1, \ldots, n)$ that can take possible values in $\mathcal{X}_d = \{x_d^0 = 0, x_d^1 = 1\}$, where the value $x_d^1 = 1$ indicates that item I_d is selected for the subset while the value $x_d^0 = 0$ indicates that item I_d is not selected. The decision variables are assigned values in a fixed order, starting with decision variable X_1 and ending with decision variable

X_n. Important to note is that in this representation, at each step d of the solution construction process, only two solution components, more specifically the inclusion or exclusion of item I_d, compete for selection. To extend the solution with a given solution component, we apply the following probabilistic rule

$$p(x_d^{j_d} \mid \boldsymbol{x}^p) = \frac{[\tau_d^{j_d}]^\alpha [\eta_d^{j_d}]^\beta}{\sum\limits_{x_d^k \in R(\boldsymbol{x}^p)} [\tau_d^k]^\alpha [\eta_d^k]^\beta}\,, \qquad \text{for all } x_d^{j_d} \in R(\boldsymbol{x}^p)\,. \tag{4.4}$$

In contrast to the VLR, the BSR is non-redundant in the ACO framework, meaning that there will be no bias due to overrepresentation of some parts of the solution space. Every possible subset corresponds to exactly one path that can be taken by the ants.

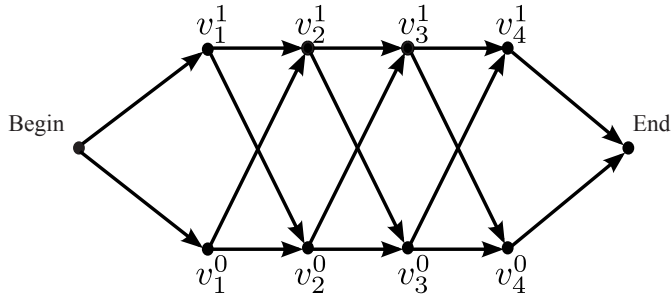


Figure 4.5: Bit string representation for the selection of items from a given set $I = \{I_1, \ldots, I_4\}$

However, in the BSR, a construction bias is present which is specific to the way ants construct these solutions in ACO. On the other hand, GA, which uses the same representation to encode solutions to the subset selection problem in its chromosomes, does not suffer from this bias. To explain this type of bias, we make use of a tree representation of the search space for the knapsack problem (Figure 4.6). The branches leading to infeasible solutions that violate the constraints are represented by dashed lines. The root node of the tree represents the empty partial solution. As the tree is traversed from top to bottom, the partial solution is extended with solution components. Each node of the tree thus represents a partial solution $\boldsymbol{x}^p$ and the leaves of the tree represent the possible complete solutions. For each node, the partial solution is presented between brackets $\langle\rangle$. Above each partial solution, the conditional probability of constructing this partial solution from the partial solution in the node above is presented in the first iteration of the ACO algorithm, when all pheromone trail parameters $\tau_d^{j_d}$ are equal. Note that we would like to obtain equal probabilities for every possible solution when all pheromone trail parameters are equal.

Figure 4.6 illustrates a negative bias. More specifically, this figure shows that the different feasible solutions do not have an equal selection probability —the

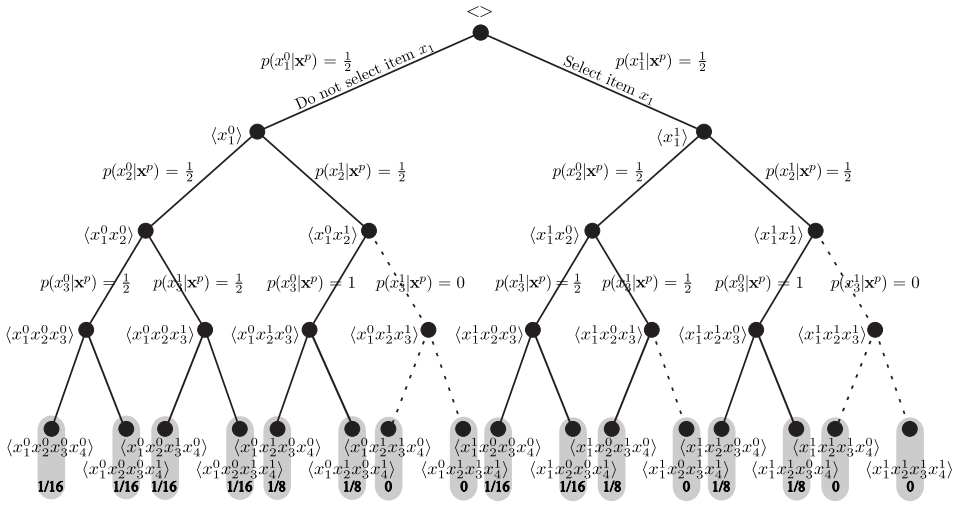


Figure 4.6: Possible tree representation of a simple knapsack problem for the BSR. The dashed lines represent branches leading to infeasible solutions

selection probability can be obtained by multiplying the conditional probabilities leading to the leaf— even though this was precisely attempted by choosing the pheromone trail parameters all equal in the first iteration of the algorithm. This bias is a consequence of a forced choice corresponding to an increased conditional probability for branches that originate from a vertex that also contains branches that produce infeasible solutions and can thus not be chosen. Clearly, the unequal selection probabilities of the different solutions depend on the order in which the different solution components are assigned. A different assignment order would result in a different distribution of the unequal probabilities over the different solutions.

As was first proposed by Verwaeren *et al.* [55], we should modify the probabilistic decision rule in order to prevent this bias. As mentioned above, the nodes of the tree represent partial solutions $\boldsymbol{x}^p$. The number of leaves or terminal nodes in a subtree $\boldsymbol{x}^p$ is denoted as $N(\boldsymbol{x}_p)$. Each leaf corresponds to a complete solution sequence that can be feasible or infeasible. The number of feasible solutions in a subtree is denoted $N^a(\boldsymbol{x}^p)$. Using these new variables, we develop a variable heuristic information parameter $\eta(x_d^{j_d})$:

$$\eta(x_d^{j_d}) = \frac{N^a\left(\langle x_1^{j_1}, \ldots, x_d^{j_d}\rangle\right)}{N^a\left(\langle x_1^{j_1}, \ldots, x_{d-1}^{j_{d-1}}\rangle\right)}\,. \tag{4.5}$$

The combination of Eqs. (4.4) and (4.5) leads to the following probabilistic decision

rule

$$p\left(x_d^{j_d} \mid \langle \boldsymbol{x}^p \rangle\right) = \begin{cases} \frac{N^a\left(\langle x_1^{j_1},\ldots,x_d^0\rangle\right)\tau_d^0}{N^a\left(\langle x_1^{j_1},\ldots,x_d^0\rangle\right)\tau_d^0 + N^a\left(\langle x_1^{j_1},\ldots,x_d^1\rangle\right)\tau_d^1}\,, & \text{if} \quad j_d = 0\,, \\ \frac{N^a\left(\langle x_1^{j_1},\ldots,x_d^1\rangle\right)\tau_d^1}{N^a\left(\langle x_1^{j_1},\ldots,x_i^0\rangle\right)\tau_d^0 + N^a\left(\langle x_1^{j_1},\ldots,x_d^1\rangle\right)\tau_d^1}\,, & \text{if} \quad j_d = 1\,, \end{cases} \tag{4.6}$$

where we now restrict to the choice $\alpha = \beta = 1$. As the sum of the probability of going right $\left(p\left(x_d^1 \mid \langle \boldsymbol{x}^p \rangle\right)\right)$ and the probability of going left $\left(p\left(x_d^0 \mid \langle \boldsymbol{x}^p \rangle\right)\right)$ in the tree is equal to one, we can conclude that the probabilities are valid. When the probabilistic decision rule in Eq. (4.6) is used and all pheromones are equal, each leaf in the tree representing a feasible solution will be reached with equal probability $\frac{1}{N^a(\langle\rangle)}$.

However, when using this probabilistic decision rule, problems occur when the pheromones are updated. For example, when we look at the tree in Figure 4.6, we can see that under node $\langle x_1^0 \rangle$ 6 feasible leafs are present and under node $\langle x_1^1 \rangle$ 5 feasible leafs are present. Therefore, in case of equal pheromones, the probability of not selecting item 1 ($X_1 = x_1^0$) is $\frac{6}{11}$ and the probability of selecting item 1 ($X_1 = x_1^1$) is $\frac{5}{11}$. Although these probabilities are different, we do not want that these different selection probabilities are reflected in the pheromone updates τ_1^1 and τ_1^0. If this were the case, it would correspond to an *a priori* expectation that solutions containing $X_1 = x_1^0$ have a better function value for all possible objective function. In order to solve this problem, we introduce the concept of a 'guided choice'. The extension of a partial solution $\boldsymbol{x}^p$ with a solution component $x_d^{j_d}$ is guided if $N^a(\langle x_1^{j_1},\ldots,x_d^0\rangle) \neq N^a(\langle x_1^{j_1},\ldots,x_d^1\rangle)$. This problem can be resolved by using following pheromone update rule

$$\begin{aligned} \tau_d^{j_d}(k+1) = (1-\rho)\,\tau_d^{j_d}(k) \\ + \rho \sum_{\{i \mid x_d^{j_{d,i}} = x_d^{j_d}\}} 2\left(1 - \frac{N^a(\langle x_1^{j_{1,i}} x_2^{j_{2,i}} \ldots x_d^{j_{d,i}}\rangle)}{N^a(\langle x_1^{j_{1,i}} x_2^{j_{2,i}} \ldots x_{d-1}^{j_{d-1,i}}\rangle)}\right) F(f(\boldsymbol{x}_i))\,. \end{aligned} \tag{4.7}$$

A disadvantage of the adapted probabilistic decision rule (Eq. (4.6)) is that the complete tree describing the search space of the problem has to be known in advance, which is for some problems practically the same as doing an exhaustive search of the complete search space. This disadvantage is not present when the number of solutions beneath the different nodes is exactly known without constructing these solutions, an example of which will be encountered in Part II, Chapter 7. Important to note is that when the pheromone concentrations are not equal for all samples, there is still bias present due to the fixed order of the samples. This bias is known as assignment order bias. However, as this assignment order bias is strongly related to the construction bias, solutions for the construction bias will also partially solve this assignment order bias. Another possible solution is to

randomize the order of the samples to be selected for each ant.

4.2.4. Handling constraints

Several methods exist to deal with constraints and infeasible solutions. Firstly, some implementations prevent infeasible solutions from occurring. At every construction step, only components that can lead to feasible solution sequences can be added to the current partial solution. This approach was described in Section 4.2.3 for the case of subset selection problems. Secondly, we can also make use of repair operators. All combinations of solution components are then allowed during the solution construction process. At the end of the construction procedure, sequences resulting in infeasible solutions will be mapped to nearby feasible solutions. However, avoiding infeasible solutions can result in the introduction of bias. For example, when using the repair operators, the mapping of infeasible solutions to nearby feasible solutions can introduce redundant representations. This bias is known as construction bias [48].

4.2.5. Description of the parameters

As in the above described algorithms, here also we have to determine some parameters. These parameters are the population size N, the number of iterations K, the evaporation parameter ρ and the exponent α and β. Again, we have to make a balance between the necessary accuracy and the computational cost, which is reflected in the population size and the number of iterations. The evaporation parameter reflects the exploitation-exploration ratio. A small evaporation parameter results in a lot of exploration and little exploitation and therefore minimizes the risk of convergence to local optima. However, when the evaporation parameter is too small, no convergence will be present. On the other hand, a too large evaporation parameter will result in convergence to a local optimum. The exponents α and β control the influence of the heuristic information and the pheromone concentration. The optimal values of these parameters are problem dependent.

Part II

Subset selection from multi-experiment data sets

5

Introduction

This part of the dissertation deals with the problem of subset selection. Subset selection has become an important problem since advances in experimental techniques have strongly increased the possibilities of data acquisition and data sharing. Although the availability of a large amount of data is beneficial, the post processing that has to be applied to these data might involve highly complex laboratory analysis steps and are often highly cost and time expensive. Therefore, it is not possible to apply these expensive and time-consuming analysis steps to the total data set. Restricting the analysis to a subset of the total data set might then offer a solution.

We assume to be dealing with data sets consisting of a large number of individual samples which are described by numerical variables and can thus be represented as vectors in some n-dimensional vector space. In addition, we allow to have samples originating from E different experimental settings. This poses a new problem, namely to develop a method to select an optimal subset of samples from a multi-experiment data set. Ideally, the best subset is the one which leads to the best result in the further analyses. Even if these analyses allow for a result that is easy to quantify, testing different subsets is infeasible if these analyses are cost and time expensive. Therefore, we endeavor to select an optimal subset by imposing suitable conditions on the statistical nature of the subset. Of utmost importance is that the subset is as informative as the total data set, *i.e.* that all the variability of the total data set is also present in the subset. In addition, it is important that the number of selected samples in each experiment is roughly proportional to the size of that experiment, although some experiments might require relatively more data points in a representative subset and slight deviations should be permitted. Another way to look at this problem is from the point of view of distributions. While for some applications it is useful to know which regions of the sample space are more densely crowded by samples, all these different samples will not provide new information in the further analytical steps. We thus want a subset of samples which contains an equal number of samples from every region in space where samples can possibly occur. The distribution of this subset should thus, for each variable, be a flattened version of the original distribution, *i.e.* a more uniform distribution. Put differently, we want the subset to have the same variability over a smaller number of samples. This corresponds to higher variances. We can thus reformulate the subset selection problem as an optimization problem,

where the objective is to maximize the variance of each variable for the selected subset.

It is clear that a random selection of samples does not result in an optimal subset, as a random subset inherits the distribution of the total data set. Various specific algorithms are available for selecting an optimal subset of samples. A well-known and often used method is the Kennard and Stone algorithm [56, 57]. A more advanced version of the Kennard and Stone algorithm is the Optimisable k-Dissimilarity Selection algorithm (OptiSim) [58], which is considerably faster when a large subset must be selected [56]. For less uniformly distributed data sets, techniques can be based on clustering methods such as k-means clustering. A representative number of samples is then selected out of each cluster [56]. In contrast to the deterministic Kennard and Stone algorithm, the latter two subset selection algorithms are stochastic methods, *i.e.* the result can differ when the algorithm is performed multiple times. While these algorithms are not optimization methods, they also try to accomplish the goal of obtaining a subset which is more uniformly distributed. The essential ingredients of each of these methods are recapitulated in Chapter 6.

By formulating the subset selection problem as an optimization problem, we can apply the combinatorial optimization algorithms introduced in Chapter 4 to the subset selection problem. For Genetic Algorithms, the binary character of the fundamental entities can be perfectly mapped to samples being selected or not. Genetic Algorithms have been applied to the optimal subset selection problem before, but with different modifications and different objective functions than the ones introduced in this thesis. A major difference in [59] is the non-binary representation of the chromosomes. However, precisely this binary representation is our motivation for applying Genetic Algorithms, rather than other heuristic optimization techniques, to the optimal subset selection problem. Instance selection — the selection of representative samples from a large data set — is an important aspect in a wider range of problems. The application of evolutionary algorithms to the problem of instance selection has already been discussed in great detail in [60, 61]. However, the context and the goal of instance selection in these papers is strongly different, and not directly transferable to class of problems that is aimed at in this part of the thesis. In order to compare Genetic Algorithms with another biologically inspired algorithm, we also solve the subset selection problem with Ant Colony Systems.

Chapter 6 thoroughly describes the different subset selection methods used in this part of the dissertation. In Chapter 7, these subset selection methods are applied to a case study consisting of a data set that contains the concentration of a number of fatty acids in a large number of milk samples, stemming from multiple experiments. The objective is to select a subset of milk samples in which each of the different experiments is sufficiently represented. Chapter 7 thus evaluates and compares

the different algorithms. At last, in Chapter 8 a conclusion about this part of the dissertation is formulated.

6

Methodology

This chapter deals with the methodology connected to subset selection problems. Firstly, a definition of a subset selection problem is given (Section 6.1). Secondly, in Section 6.2 the different subset selection algorithms are discussed.

6.1. Definition of the subset selection problem

In a general subset selection problem, the objective is to select an optimal subset out of a total set $\mathcal{I}$ of n samples $\{I_1, \ldots, I_n\}$. The definition of optimal differs according to the problem at hand and is often characterized by an objective function which has to be optimized by the subset. Possible subsets are denoted as elements $\boldsymbol{x} \in \mathcal{S} = 2^{\mathcal{I}}$, where $2^{\mathcal{I}}$ denotes the powerset of $\mathcal{I}$, *i.e.* the set containing all possible subsets of $\mathcal{I}$. In addition, there is often one or more constraints that restrict the subsets that are allowed. The objective function and the constraints depend on the application at hand. A well-known example is the knapsack problem. In the 0/1 dimensional knapsack problem, the objective is to select a subset of samples that maximizes the sum of the values of these samples. At the same time, the sum of the weigths of the samples in this subset is upper bounded by a certain value.

In this part of this dissertation, another kind of subset selection problem is tackled. The objective is to select a subset of a given number n_{subset} of samples that is representative of the total set of samples. For this type of subset selection problem various problem-specific algorithms were developed to select an optimal subset of samples, such as the Kennard and Stone algorithm, the k-means clustering based algorithm and the OptiSim algorithm. These are not optimization algorithms. In the remainder of this section, we define a quantitative objective function that captures the essence of the qualitative requirement for the subset. This allows us to also apply general combinatorial optimization algorithms, as described in Chapter 4, to the aforementioned problem. The subset selection problem is then a specific type of combinatorial optimization problem that can be written as

$$\max_{\boldsymbol{x} \in 2^{\mathcal{I}}} f(\boldsymbol{x}), \qquad \text{subject to} \quad \#\boldsymbol{x} = n_{\text{subset}}\,. \tag{6.1}$$

where the function $f : 2^{\mathcal{I}} \to \mathbb{R}$ is the objective function to be optimized and $\#\boldsymbol{x}$ counts the number of samples in the subset $\boldsymbol{x}$.

Finally, we construct a suitable objective function to quantify how representative a subset is of the total data set. In the case where the samples solely consist of numerical data, we can represent them as vectors $\boldsymbol{X} \in \mathbb{R}^D$, where D represents the number of dimensions of the samples, *i.e.* the number of variables that were measured for each sample. As mentioned in the introduction, the aim is to have a subset with a smaller number of samples that contains the same variability as the total data set in each variable $i = 1, \ldots, D$. This should result in an increase of the variances. A well-considered fitness function is then given by

$$\kappa = \min_{i=1}^{D} \frac{\mathrm{var}_{i,\mathrm{sel}}}{\mathrm{var}_i} \,. \tag{6.2}$$

The abbreviation 'sel' in the subscript indicates that these variances are calculated using the values X_i of variable i of all samples $\boldsymbol{X}$ in the subset, as opposed to the variances calculated using the total data set. This ratio is calculated for all variables $i = 1, \ldots, D$ of the data separately. Subsequently, the minimum of these D ratios is computed. This fitness function is to be maximized.

In case of multi-experiment data sets, we can try to steer the partitioning of the selected samples over the different experiments. Without predetermining the number of samples per experiment, it is possible to use metaheuristics for discrete optimization. To this end, we adapt the fitness function as follows:

$$\kappa^* = \min_{e=1}^{E} \left(\min_{i=1}^{D} \frac{\mathrm{var}_{i,e,\mathrm{sel}}}{\mathrm{var}_{i,e}} \right) , \tag{6.3}$$

with e the index indicating the experiment and E the number of experiments. Now, the numerator of each ratio in the right-hand side is calculated using the values X_i of all samples $\boldsymbol{X}$ in the subset belonging to the experiment e. In the denominator, the values X_i of all samples $\boldsymbol{X}$ belonging to experiment e in the total data set are used. In order to have $\mathrm{var}_{i,e,\mathrm{sel}} > 0$ for all values of i and e, at least two samples of each experiment should be selected in the optimal subset. Since optimization algorithms try to maximize this fitness function, they naturally impose this minimal selection of two samples from each experiment.

6.2. Subset selection algorithms

While Genetic Algorithms and Ant Colony Systems are described in full generality in Chapter 4, this chapter explains specific modifications that were introduced to make them fully compatible with the optimal subset selection problem. For Genetic Algorithms (Section 6.2.4), we introduce new mutation and crossover operators

that allow to fix the total number of selected samples. For Ant Colony Systems (Section 6.2.5), the computational cost to solve this class of subset selection problems is considerable and a parallel implementation is constructed. In addition, we discuss how Genetic Algorithms and Ant Colony Systems allow to take the multi-experiment aspect of the data set into account, by constructing an appropriate fitness function. Such an approach is not possible with the deterministic methods, unless the number of samples to retain from each experiment is fixed in advance. Except for the Kennard and Stone algorithm (Section 6.2.1), the only straightforward way to take the different experiments into account, is to treat the experiments independently.

6.2.1. The Kennard and Stone algorithm

The Kennard and Stone algorithm [56, 57] requires the definition of a distance measure on the set of samples and aims at sequentially selecting samples that are uniformly distributed over the range of the total data set. The starting point is the sample that is closest to the mean. Each newly added sample fulfills the requirement that it is located as far as possible from the set of already selected samples. This requirement uses the notion of distance between a point and a set of samples, which is defined as the minimum over all distances between the single point and each of the samples in that set. The candidate sample with the largest distance is then added to the set of selected samples. This procedure is repeated until a predetermined number of selected samples is obtained [57]. A slightly modified version of the Kennard and Stone algorithm has been developed to select two or more independent subsets [62]. The pseudocode for the standard Kennard and Stone algorithm can be found in Algorithm 4.

Algorithm 4: Kennard and Stone algorithm

Data: Total set of n samples
Result: Set of n_{subset} selected samples
Calculate distance between all the samples;
Calculate distance between all the samples and the mean;
Select as first sample the one closest to the mean;
Select as second sample the one that is most distant from the first;
Define the set of candidate samples as the remaining samples;
while *number of selected samples* $< n_{subset}$ **do**
    Calculate distance between candidate samples and the set of selected samples;
    Select as next sample the one for which this distance is maximal;
    Increase number of selected samples with one;
    Remove the selected sample from the set of candidate samples;

For two samples $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^D$, we can define the Euclidean distance:

$$d_E(\boldsymbol{X}, \boldsymbol{Y}) = \|\boldsymbol{X} - \boldsymbol{Y}\| = \sqrt{(\boldsymbol{X} - \boldsymbol{Y})^T(\boldsymbol{X} - \boldsymbol{Y})} \tag{6.4}$$

and the Mahalanobis distance:

$$d_M(\boldsymbol{X}, \boldsymbol{Y}) = \sqrt{(\boldsymbol{X} - \boldsymbol{Y})^T Q^{-1}(\boldsymbol{X} - \boldsymbol{Y})}\ , \tag{6.5}$$

where Q is the covariance matrix, calculated using the total data set. In contrast to the Euclidean distance, the Mahalanobis distance takes into account the geometrical form of the variations present in the data set, and is therefore data set dependent.

In case of multi-experiment data sets, it is not straightforward to take the different experiments into account. When the number of samples to be selected from each experiment is fixed, there are two possibilities: the first possibility is to divide the data set into separate data sets, one for each experiment, and to apply the Kennard and Stone algorithm on these data sets independently, selecting the requested number of samples from each experiment. The second possibility is to adapt the Kennard and Stone algorithm. In this case, the algorithm has to keep track of the number of already selected samples for each experiment. When the predetermined number of samples is obtained for a certain experiment, it is no longer possible to select samples from that experiment. The pseudocode for the adapted Kennard and Stone algorithm can be found in Algorithm 5.

6.2.2. The k-means clustering based algorithm

As mentioned in the introduction, a clustering technique such as k-means clustering can be used when the data set is less uniformly distributed. k-means clustering is a stochastic clustering technique based on the Euclidean distance [63]. The pseudocode can be found in Algorithm 6. We set the number of clusters k equal to the number of samples we want to select ($k = n_{\text{subset}}$). After the data is clustered, we select from each cluster the sample that is closest to the cluster center. These samples constitute our group of selected samples. As in the case of the Kennard and Stone algorithm, when dealing with multi-experiment data sets, we can only take the different experiments into account when the number of selected samples from each experiment is fixed in advance. For this algorithm, the best solution is to subdivide the data set and to apply the algorithm on the different data sets independently.

Algorithm 5: Kennard and Stone algorithm for multi-experiment data

Data: Total set of n samples
Result: Set of n_{subset} selected samples
Determine how many samples have to be selected from each experiment e;
Calculate distance between all the samples;
Calculate distance between all the samples and the mean;
Select as first sample the one closest to the mean;
Select as second sample the one that is most distant from the first;
Define the set of candidate samples as the remaining samples;
while *number of selected samples* $< n_{subset}$ **do**
    Calculate distance between candidate samples and the set of selected samples;
    Select as next sample the one for which this distance is maximal;
    Find to which experiment e this sample belongs;
    Increase number of selected samples for the corresponding experiment with one;
    if *number of selected samples exp.* e = *predetermined number of selected samples exp.* e **then**
        Remove all samples belonging to experiment e from set of candidate samples;
    else
        Remove selected sample from set of candidate samples;

Algorithm 6: The k-means clustering based algorithm

Data: Total set of n samples
Result: n_{subset} clusters
Make at random n_{subset} clusters;
Calculate cluster centers (the mean of each cluster);
while *cluster centers at time* $k+1 \neq$ *cluster centers at time* k **do**
    Calculate Euclidean distance between each sample and the cluster centers;
    Unite each sample with the cluster for which this distance is minimal;

6.2.3. The OptiSim algorithm

The OptiSim algorithm aims at constructing an optimal subset by selecting samples uniformly over the total data set [56]. OptiSim requires some user-defined input parameters: a threshold ϵ, which is the required minimum distance between two selected samples (both the Euclidean and Mahalanobis distance can be used), the number of samples $k = n_{\text{subset}}$ that have to be selected and the size S of a temporary subset that is created to render the computation less time consuming.

The algorithm runs through a number of iterations, in which four different sets of samples are used. The set of available samples ($= A$) contains all samples at the start of the algorithm. The final result is the set of selected representative samples ($= B$). Furthermore, a temporary subset of S candidate samples ($= \mathcal{S}_{\text{temp}}$) and a recycle bin of candidate samples is constructed. The first sample that is selected is the one closest to the mean and is immediately removed from the set of available samples and moved to B. In each iteration step, a temporary subset of S candidate samples is constructed. Samples are randomly chosen and removed from the set of available samples. If they are separated by at least a distance ϵ from the set of already selected samples, they are added to the temporary subset. These steps are repeated until the number of samples in the temporary subset reaches the user-defined value S. The sample in this temporary subset that is most distant from the set of already selected samples is then selected as next sample. The other candidate samples in the temporary subset are moved to the recycle bin. This process is iterated until n_{subset} samples are selected. If at any time the number of available samples reaches zero, the content of the recycle bin is moved to the set of available data. The pseudocode for this algorithm is given in Algorithm 7.

Algorithm 7: OptiSim algorithm

```
Data: Total set of n samples (= A), parameters S and ε
Result: Set of n_subset selected samples (= B)
Calculate distance between all the samples and their mean;
Take as first sample the one closest to the mean and move it to B;
while size of B < n_subset do
    while size of S_temp < S do
        Select a random sample from A;
        Remove sample from A;
        if distance to B > ε then
            Add sample to S_temp;
        if number of available samples = 0 then
            Move recycle bin to A
    Move the sample from S_temp that is most distant from B, to B;
    Move the remaining samples from S_temp to the recycle bin;
```

The two algorithm-specific parameters are the temporary subset size S and the threshold distance ϵ. Suitable values for these parameters were estimated in [56]. The default value for S is between 5 and 25 percent of the original data set size n. When using the Euclidean distance, a good value for ϵ can be found by filling the volume V of the data set with n_{subset} spheres with radius $r = \epsilon/2$. The volume of the data set is then given by [56]:

$$V = n_{\text{subset}} \frac{\sqrt{\pi^D}}{\Gamma(D/2+1)} \left(\frac{\epsilon}{2}\right)^D , \tag{6.6}$$

where D is the dimensionality of the data set and Γ is Euler's Gamma function. The volume V of the data set can be calculated as [56]

$$V = \prod_{i=1}^{D} \text{range}(X_i) , \tag{6.7}$$

where $\text{range}(X_i)$ represents the range covered by the data set along the ith dimension.

Rewriting Eqs. (6.6) results in a maximum value for ϵ:

$$\epsilon = 2 \left[\frac{\Gamma(D/2+1)V}{n_{\text{subset}}\sqrt{\pi^D}} \right]^{1/D} , \tag{6.8}$$

where we use Eq. (6.7) for the volume V. However, this formula can only serve as an estimate for the order of magnitude of ϵ. The reason is that even with a densest packing configuration, which is highly unlikely, the filling factor of the spheres in the total volume decreases exponentially as 2^{-D} [64]. A general cure for this problem is to replace the factor 2 in the expression above with an arbitrary constant c:

$$\epsilon = c \left[\frac{\Gamma(D/2+1)V}{n_{\text{subset}}\sqrt{\pi^D}} \right]^{1/D} . \tag{6.9}$$

A value $c = 1$ corresponds to the densest packing configuration. Therefore, for the calculation of ϵ, we have chosen for a value $0 < c < 1$.

As for the Kennard and Stone algorithm and the algorithm based on k-means clustering, the number of selected samples from each experiment has to be fixed in advance, in order to take the different experiments into account. The only possible way to select these samples without compromising the mechanism of OptiSim, is to subdivide the data set and to apply the OptiSim algorithm independently on the different data sets corresponding to the different experiments.

6.2.4. Genetic Algorithms

Genetic algorithms are extensively described in Section 4.1. For clarity, we briefly recall the most important concepts. The optimization process runs through a number of generations. A genetic algorithm starts with the initialization of a population of chromosomes. For the subset selection problem, the binary representation is the most adequate representation for the chromosomes. The length of the chromosomes then equals the total number of samples n in the data set. A sample is selected for the subset when it has value one and is not selected when it has value zero in the chromosome.

Each chromosome is assessed by a fitness function, for which we can choose Eq. (6.2), when not taking into account the different experiments, or Eq. (6.3) when taking into account the different experiments. We thus do not have to predetermine the desired number of samples per experiment. In each generation a new population is created through the selection of parents using tournament selection. Children are then created through crossover and mutation of the parents, as explained in the next paragraph. These children constitute the new population. The best chromosome of the previous population can be preserved as a member of the new population, a strategy that is called elitism [40]. The pseudocode of the algorithm can be found in Algorithm 2 in Section 4.1.

The standard crossover operators, such as uniform crossover, one-point crossover and m-point crossover [65], do not guarantee that the number of selected samples is fixed to the value n_{subset}. As we want this number to equal a specified constant, we must modify the crossover operator. How this is done, is illustrated in Algorithm 8.

Algorithm 8: Adapted crossover

Data: Parents
Result: Children
P_1 = samples that are selected in parent 1;
P_2 = samples that are selected in parent 2;
$I = P_1 \cap P_2$;
$C = (P_1 \setminus I) \cup (P_2 \setminus I)$;
Take for child 1 at first the samples from I followed by a random selection of half of the samples from C;
Remove this random part from C;
Take for child 2 at first the samples from I and for the remainder the remaining samples from C;

The mutation is also altered. Whenever a one is changed into a zero another zero is changed to one, or vice versa. In this way the number of selected samples remains fixed.

Finally, we remark that we did not take into account the presence of bias for Genetic Algorithms (where it is often called deception) [66]. As the results in

Chapter 7 show, it is likely that no bias is present for the type of subset selection problem discussed in this case study.

6.2.5. Ant Colony Systems

Section 4.2 gives a detailed description of Ant Colony Systems. In the general AS algorithm, a population of artificial ants starts with an initially empty partial solution sequence $\boldsymbol{x}^p = \langle\rangle$ and add solution components to it in each iteration k, according to the probabilistic rule in Eq. (4.1) (Chapter 4, Section 4.2). This continues until the solution sequence is complete. When this construction phase is finished, the pheromones τ are updated by Eq. (4.2) (Chapter 4, Section 4.2).

For the subset selection problem, we can use both the variable length representation (VLR) and the bit string representation (BSR). Since each sample of the total set of n_{subset} selected samples has the same weight, the VLR is not plagued by representation errors. The number of permutations is the same for every solution, namely $n_{\text{subset}}!$. For the BSR, the representation error is always absent. However another kind of negative bias can be detected, which can be resolved by introducing some modifications as explained in Section 4.2.3. The VLR, BSR and adapted BSR will be compared in Chapter 7.

As fitness function, we use Eq. (6.2) when not taking into account the different experiments and Eq. (6.3) when taking into account the different experiments. As in Genetic Algorithms, we do not need to predetermine the desired number of samples per experiment.

As the employment of ACO to this particular subset selection problem is computationally very demanding, we have developed a parallel version of ACO. This parallel algorithm was implemented using the Message Passing Interface (MPI) of Octave. MPI is a library of routines that can be used to create parallel programs. A parallel program makes use of two or more processes. Depending on the application, several paradigms of parallel programs exist [67]. In this application, we have chosen for a master-slave paradigm. Initially, only one process is used, this process is called the master process, which controls the program. The master process is thus the manager and decides which tasks have to be completed by the other processes, which are called the slave processes. The slave processes then send the results of the completed tasks back to the master process. The program is finished when the master process sends a 'timetoquit' to the slave processes. The pseudocode of the

parallel ACO is presented in Algorithm 9.

Algorithm 9: A parallel ACO algorithm

```
Initialize MPI;
Set parameters;
if master process then
    while no convergence do
        Send assignment to construct solution x to slaves;
        Receive solutions constructed by slaves;
        Update best solution ;
        Update pheromones τ ;
    Send timetoquit;
else
    Timetoquit is false;
    while timetoquit is false do
        Receive messages from master;
        Construct solution x;
        Send solution x to master;
```

7

Case Study

In this chapter, the subset selection algorithms, described in full detail in Chapter 6, are applied to a case study. Section 7.1 describes the data set used in this case study. This data set consists of the concentration of 45 fatty acids in a large number of milk samples belonging to multiple experiments. Section 7.2 presents the results of the different subset selection algorithms and discusses them thoroughly. In order to do this, we make a distinction between the results obtained when not taking the different experiments into account (Section 7.2.2) and the results obtained when taking the different experiments into account (Section 7.2.3). In Section 7.2.4, this analysis is repeated by restricting the data to the most important fatty acids only. Section 7.2.5 discusses the statistical significance of the differences in the results obtained with the different algorithms. Then, in Section 7.3 the distribution of the optimal subset is compared with the distribution of the total data set.

7.1. Data description

This section describes the details of an example in agriculture where it is beneficial to apply a very expensive post processing step to a small subset of the original data set. This problem was presented to us by the research group of professor V. Fievez of the Department of Animal Production of Ghent University and will be used throughout this chapter to test the different subset selection methods that were introduced in the previous chapter. The original data set, which was provided to us by the Department of Animal Production, consists of measurements of the concentration of $D = 45$ fatty acids in a large number $n = 1033$ of milk samples that belong to $E = 6$ different experiments. The concentrations of the fatty acids are expressed in terms of mass percentage. These fatty acids have been identified as methylated fatty acids, after extraction and methylation according to [68], on a Hewlett-Packard 6890 gas chromatograph (Hewlett-Packard Co., Brussels, Belgium) with a CP-Sil88 column for fatty acid methyl esters (100 m $\times$ 0.25 mm $\times$ 0.2 μm; Chrompack Inc., Middelburg, the Netherlands). The temperature program was set according to [68]. We refer to this identification method as the simplified reference method.

It has been shown that milk fatty acids have the potential to monitor nutrients

produced during digestive processes [68, 69] and diagnose metabolic disorders such as acidosis and ketosis [70]. Milk fatty acids of particular importance in this respect are odd and branched chain fatty acids and trans-isomers of C18:1 and cis/trans C18:2. These fatty acids are called the priority fatty acids here. However, although the highly polar cyanoalkyl polysiloxane stationary phase, that was used to obtain the current data set, is most widespread, it cannot resolve all milk fatty acids. Indeed, some trans C18:1, cis C18:1 and cis/trans C18:2 isomers are only partially resolved and overlap of trans C16:1 with branched chain C17:0 fatty acids has been reported [71]. Moreover, identification of several fatty acids is also challenging due to the limited availability of standards. In parallel research the objective is to construct the best possible calibration equations for the odd and branched chain fatty acids and the trans-isomers of C18:1 and cis/trans C18:2 using the spectra of raw milk or milk fat from different spectrophotometrical techniques (such as mid-infrared (MIR), near-infrared (NIR) and Raman spectroscopy). Hence, the separation of the different fatty acids and their isomers needs to be as high as possible. Therefore, the best possible reference data is needed, which gives a more accurate representation of the actual amounts of the fatty acids in the samples.

However, for budgetary reasons we have to create a subset of $n_{\text{subset}} = 100$ reference samples on which a detailed milk fatty acid reference analysis is to be performed using different GC-settings, which might include following methods: 1) according to [68], a Hewlett-Packard 6890 gas chromatograph (Hewlett-Packard Co., Brussels, Belgium) with a CP-Sil88 column for fatty acid methyl esters (100 m $\times$ 0.25 mm $\times$ 0.2 μm; Chrompack Inc., Middelburg, the Netherlands) but using different temperature programs, allowing to resolve a larger set of individual fatty acids [72]; 2) according to [73] a Hewlett-Packard 6890 gas chromatograph (Hewlett-Packard Co., Brussels, Belgium) with a Solgel-wax column (30 m $\times$ 0.25 mm $\times$ 0.25 μm; SGE Analytical Science, Victoria, Australia), which does not allow separation of cis and trans C18:1 isomers, but results in an improved resolution of branched chain and some trans-mono-unsaturated fatty acids and 3) a comprehensive two-dimensional GC, which is a multi-dimensional separation technique, allowing separation on two GC columns with different separation mechanisms [71], which should allow improved separation and identification of several milk fatty acid methyl esters. The reference data set created using these gaschromatographic approaches will be used further to assess the prediction of odd and branched chain and the trans/cis isomers of the C18:1 and C18:2 fatty acids of interest. This prediction will be based on several spectrophotometrical techniques for the analysis of the selected milk samples and will be used to improve chemometrical methods applied for this prediction.

In order to have a wide range of concentrations of several milk fatty acids of interest, milk samples of six experiments are considered, in which cows were subjected to different diets. The total data set contains $m = 1033$ milk samples. In experiment 1

[74], milk samples were obtained throughout the lactation period from 20 cows, divided in two equal groups with limited or ad libitum access to the compound feed. Experiment 2 [75] and 3 (unpublished results) consisted of a 6- and 12-weeks trial in which twelve and four cows were subjected to a gradual increase of quickly fermentable carbohydrates in the compound feed. In experiment 4 [76], three cows in mid lactation were used to examine milk fatty acid composition responses to micro algae feeding during 20 days. Milk samples of experiment 5 [77] were derived from 16 cows during the first 12 weeks after parturition. Eight cows were offered a standard diet, whereas the other eight animals received a compound feed with micro algae, similar to the test feed of experiment 3. In experiment 6 (unpublished results), three different linseed sources (extruded, rolled and rumen by-pass) were fed to three groups of six cows during a period of six weeks. Since the subset of samples has to be representative for the total data set, it is important that a sufficient number of samples are selected from each experiment. A good guideline is to have the number of selected samples in each experiment roughly proportional to the size of that experiment, but this should not be a strict rule.

As mentioned in Section 6, it is important that the subset of $n_{\text{subset}} = 100$ samples possesses the same variability as the total data set. Because the data stems from different experiments, it is recommended to have all experiments represented in the subset. Therefore, a good objective is to maximize the variance of each variable for the selected subset. The presence of outliers might pose an issue with this objective function. Because we want to maximize the variances, it is likely that these outliers will be selected. Since each of the experiments have been checked thoroughly before publication or internal reporting (of the unpublished results), extreme values which remain in the current data set are not false measurements but contain valuable information. Accordingly, in our case, the data are assumed to be free from outliers. This is also verified by calculating the deviation between the mean for the subset and the total data set for each variable. When the distribution of the subset is not shifted into a particular direction with respect to the original distribution, we can assume that no outliers are selected in the subset. Indeed, if an outlier would be part of the subset, it would have a larger effect on the mean of each fatty acid for the subset due to the smaller number of samples.

7.2. Results and discussion

This section presents the results and discusses our observations in detail. Results are always displayed using a pair of graphs as explained in Section 7.2.1. In Sections 7.2.2 and 7.2.3, we discuss the results obtained using all fatty acids, even though we only plot data for a few fatty acids on the graph for clarity. We have also repeated the analysis for a data set that is restricted to include only these few fatty acids of particular interest. The results of that analysis are very similar, as

stated in Section 7.2.4. Finally, in Section 7.2.5, we test the statistical significance of our findings.

7.2.1. Representation of the results

The following sections display the resulting subsets selected by each of the algorithms discussed in the previous chapter. The results of each algorithm will be illustrated using a pair of graphs which always display the same content. Both graphs display quantities which are calculated for each variable $i = 1, \ldots, n$ and for each of the six experiments separately, *i.e.* these quantities are calculated using the values X_i belonging to a certain experiment. The first graph demonstrates the gain (> 1) or loss (< 1) of variance when only the selected samples are used. This number is given by the ratio of the variance of each fatty acid for the selected subset to the variance of the corresponding fatty acid for the total data set. The minimum of all these values is precisely κ^*, as defined in Section 6.2.4, and serves as a good quantification of our subset. This value will thus be included in the tables. It is important to notice that if the concentration of a fatty acid is unknown for all the samples of a certain experiment, it is impossible to calculate the variance of this missing fatty acid, both for the total set of samples and for the optimal subset. In this case the ratio $\text{var}_{i,l,\text{sel}}/\text{var}_{i,l}$, as needed in the calculation of κ^*, is set to one. When the concentration of a fatty acid is zero for all the samples of a certain experiment, the variance of this fatty acid for the total set of samples and for the optimal subset is also zero. In this case the ratio $\text{var}_{i,l,\text{sel}}/\text{var}_{i,l}$ is also set to one. The second graph displays how much the mean of each fatty acid for the selected samples deviates from the mean of the corresponding fatty acid for all the samples. This difference is standardized by dividing it by the standard deviation of all the samples. The difference is negligible when its absolute value is small compared to one. Both quantities are easy to interpret. As explained before, our objective is an increase in variance with a preservation of the mean for each variable. The increase in variance ensures that extremal points and points in the outer region of the original data set are more likely to be selected in the subset, while the preservation of the mean ensures that the subset is still homogeneously distributed over the set of all samples and not shifted to a particular direction. As discussed above, a strong shift of the mean might also indicate the presence of outliers in the data set. As we impose to maximize the variance of each fatty acid for the selected subset, outliers will certainly be selected if they are present.

When no samples are selected from a certain experiment, this experiment will not be displayed on the graphs. If only one sample is selected from a certain experiment, all variances calculated using this selected sample will amount to zero. Since this cannot be represented on a logarithmic scale, this experiment will also be omitted from the graphs presenting loss or gain of variance.

In order to make a statistical comparison between the different subset selection algorithms, the stochastic techniques k-means, OptiSim, Genetic algorithms and Ant Colony Systems were repeated 50 times. In the graphs, a single generic solution is presented and in the tables the mean of the 50 repetitions is presented. As was already stated at the beginning of this section, only a few fatty acids of particular interest are represented on the graphs but all data is taken into account in the comparative tables.

7.2.2. Without distinction between the experiments

In this section, we discuss the results obtained with the different subset selection techniques as presented in Section 6. The techniques applied to obtain the results in this subsection were not informed about the subdivision of the total data set in six different experiments. In the next subsection, we will show how taking the different experiments into account improves the results.

The Kennard and Stone algorithm

Euclidean distance (ED): Figure 7.1 illustrates that the variances of some fatty acids are smaller for the selected subset than for the total data set when not taking the different experiments into account. Table 7.1 shows that only one sample is selected from experiment 4, which explains why the value of κ^* is zero and why this experiment is not presented in Figure 7.1. The value of κ indicates that, even when no dinstinction is made between the experiments, for at least one fatty acid a strong loss of variance is observed. Finally, Figure 7.2 demonstrates that the means of some fatty acids for the selected subset deviate strongly from the corresponding means computed using the total data set.

Mahalanobis distance (MD): When the Mahalanobis distance is used, samples from each experiment are selected (Table 7.1). However, this only slightly improves the values of κ and κ^* with respect to the Euclidean case (Figure 7.3). The deviations from the means are on average larger (Figure 7.4). We can argue that the use of the Mahalanobis distance is not a proper choice for the optimal subset selection problem. The definition of the Mahalanobis distance is such that the data is transformed, and directions of high variance can no longer be distinguished from directions of low variance. The Mahalanobis distance makes it harder to detect the extremal values which are responsible for a large part of the variability of the data set and which should be included in the set of selected samples.

The k-means clustering based algorithm

With this technique samples are proportionally selected from each of the experiments (Table 7.1). The deviations from the means between the selected subset and the total data set are rather small in most cases (Figure 7.6). However, for certain experiments, the variance of a large number of fatty acids is smaller for the selected subset than for the total data set (Figure 7.5). This can also be noticed in Table 7.1 (κ and κ^*).

The OptiSim algorithm

Samples are selected from each experiment (Table 7.1). The variances of some fatty acids are much lower for the subset selected with the OptiSim algorithm than for the total data set, as indicated by the low values of κ and κ^* (Figure 7.7). The means of the fatty acids for the selected samples are also strongly shifted for some fatty acids (Figure 7.8).

Genetic Algorithms

To determine the optimal values for the population size, the number of generations and the crossover and mutation probability of Genetic Algorithms, we performed an exhaustive search. We examined a crossover probability r_c of 0.3, 0.6 and 0.9; a mutation probability r_m of 0.01, 0.001 and 0.0001; a population size N of 50, 100, 150 and 200 and a number of generations K of 200, 300, 400 and 500. Because of the presence of mutation in genetic algorithms, we cannot expect the whole population to converge to a single point, unless the mutation probability is gradually turned to zero. However, we choose to fix the number of generations. Figure 7.9 (a) illustrates that the best objective function value is obtained with a number of generations $K = 400$ or $K = 500$ and a mutation probability $r_m = 0.0001$. Figure 7.9 (b) shows that a crossover probability $r_c = 0.9$ in combination with a mutation probability $r_m = 0.0001$ leads to the best objective function value. Figures 7.9 (c) and (d) indicate that a population size $N = 200$ in combination with a number of generations $K = 400$ or $K = 500$ results in a better objective function value. At last, Figures 7.9 (e) and (f) illustrate that a crossover probability $r_c = 0.9$ gives better results than a lower crossover probability. Therefore, the chosen parameter values are $r_c = 0.9$, $r_m = 0.0001$, $K = 400$ and $N = 200$.

As mentioned in Section 6.2.4, with the standard crossover and mutation operator, it is not possible to fix the size of the selected number of samples. Because we wanted a subset of roughly 100 samples, we adapted these standard operators. However, to investigate if it is not possible to get a higher variance with a different subset size, we also executed Genetic Algorithms with the standard crossover and mutation operator.

Fitness function κ (adapted operators): The optimization of the fitness function κ results in the absence of samples from experiment 4. This fitness function has a value above one, which indicates there is no loss of variance when the different experiments are not taken into account (Table 7.1). This is not a surprise, because the function κ was optimized. However, the variances of some fatty acids are significantly smaller for the selected subset than for the total data set when the different experiments are considered separately (Figure 7.10). For most fatty acids, the deviation between the mean for the selected subset and the total data set is rather small (Figure 7.11).

Fitness function κ (standard operators): With the standard operators, the optimal subset consists of 117 samples. This, however, is surely not the best one we can obtain, since the value of κ that we obtained with the adapted operators is higher, although the number of samples is restricted to a fixed value of 100 in that case. In this case, the variances of some fatty acids are also significantly smaller for the selected subset than for the total data set when the different experiments are considered separately (Figure 7.12). For most fatty acids, the deviation between the mean for the selected subset and the total data set is again rather small (Figure 7.13).

Ant Colony Optimization

When we want to apply the ACO algorithm to this subset selection problem, we have to select an appropriate representation of the solution space. As mentioned in Chapter 4 (Section 4.2.3), there are two possible representations for the subset selection problem. The first representation is the variable length representation (VLR), which seems to be suitable as in this case study no representation error is present (all solutions have the same length). However, in a first test configuration it seems that the ants are not able to converge and therefore the algorithm does not stop. A possible explanation is that the different milk samples are closely positioned to each other in the 45-dimensional space of concentrations of fatty acids, so that different clusters of milk samples with nearly identical fatty acid concentrations are formed and it does not matter which sample is selected in such a cluster. Ants that have selected different samples of the same cluster will have a similar objective function value and the samples selected by these ants will have a similar pheromone concentration. Therefore, these ants will not converge to a unique solution. Figure 7.14 illustrates the distance between the different milk samples in this space. The black lines separate the different experiments. In correspondence to our findings with the Kennard and Stone algorithm, we used the Euclidean distance as this allows for the best detection of extreme distances between samples that are worth selecting. However, as is shown by Figure 7.14,

several groups of milk samples with little separation are present. More specifically the samples belonging to the same experiment and the samples of experiments 1 and 4 are positioned very close to each other, resulting in very small distances among them.

Consequently, we assume that the bit string representation, with the adapted probabilistic decision rule and the adapted pheromone update, leads to better results. The bias originating from the fixed order of the samples to be selected is a favourable effect in this case. If this bias were resolved, we would face the same problem as with the variable length representation. Because of the fixed order of the milk samples, the ants always select the same samples out of the different clusters of highly correlated samples and convergence is possible. To illustrate that the results are better with the adapted probabilistic decision rule and the adapted pheromone update rule, we will give an example of the results with the standard probabilistic decision and pheromone update rule.

As with GA, we have to determine some parameters for the ACO algorithm, namely the population size N, the initial pheromone concentration τ_0 and the evaporation rate ρ. As the algorithm runs until the ants are converged (as convergence tolerance, we choose tol= 0.001), no number of iterations has to be determined. In order to determine these parameters, we performed an exhaustive search. We tested the ACO algorithm with a population size N of 10, 20, 30 and 40, an initial pheromone concentration τ_0 of 100, 500 and 1000 and an evaporation rate ρ of 0.1, 0.5 and 0.9. Figure 7.15(a) illustrates that an evaporation rate of 0.1 in combination with a population size of 30 or 40 gives the best results. Figure 7.15(b) shows that an initial pheromone concentration of 1000 leads to higher values of the objective function than an initial pheromone concentration of 100 or 500. Figure 7.15(c) confirms our conclusions of Figures 7.15(a) and (b). We therefore choose a population size $N = 30$, an initial pheromone concentration $\tau_0 = 1000$ and an evaporation rate $\rho = 0.1$.

Fitness function κ (original AS algorithm): Figures 7.16 and 7.17 illustrate that, in the case of the original AS algorithm and no distinction between the different experiments, samples are only selected from experiments 1 and 2. Figure 7.16 shows that there is a strong loss of variance. This is also indicated by the value of κ in Table 7.1. The deviation between the mean for the selected subset and the total data set is rather small (Figure 7.17).

Fitness function κ (adapted AS algorithm): When using the AS algorithm with the adapted probabilistic update rule and the adapted pheromone update rule, samples are selected from every experiment (Figure 7.18 and 7.19). However, again a strong loss of variance is present for the selected subset of samples (Figure 7.18).

Figure 7.19 illustrates that the deviation between the mean for the selected subset and the total data set is very strong for some fatty acids.

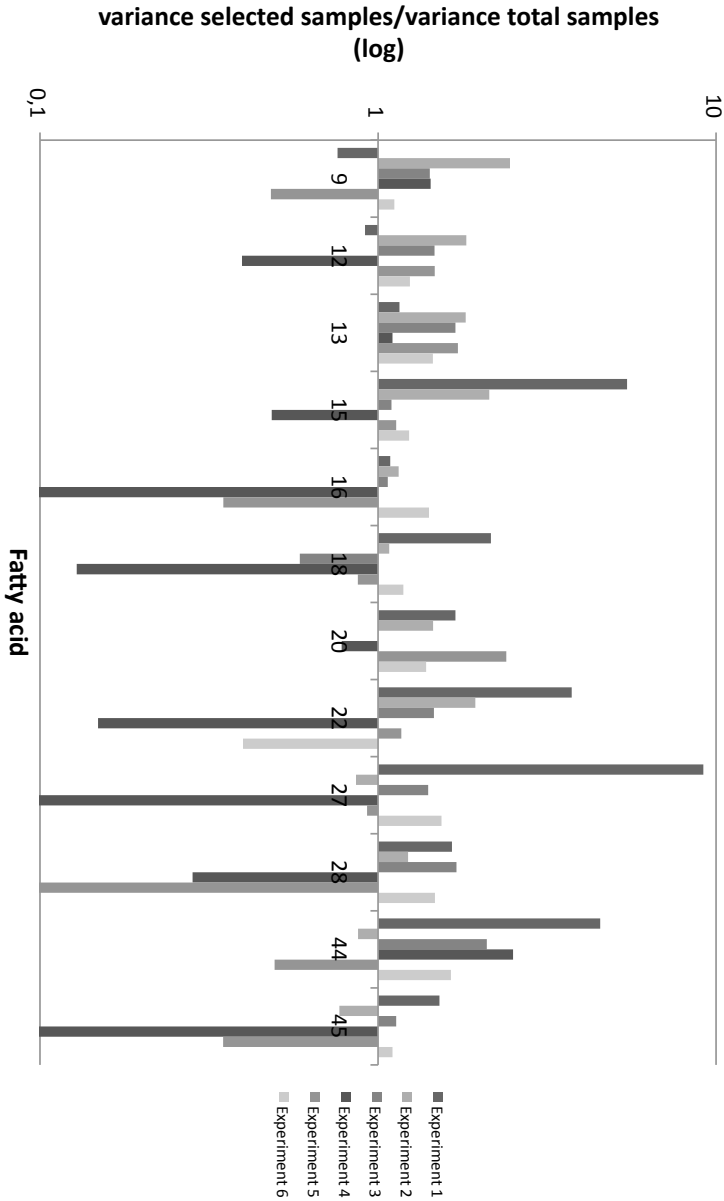


Figure 7.1: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (Kennard and Stone algorithm, Euclidean distance). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

Figure 7.2: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation for each fatty acid per experiment (Kennard and Stone algorithm, Euclidean distance). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

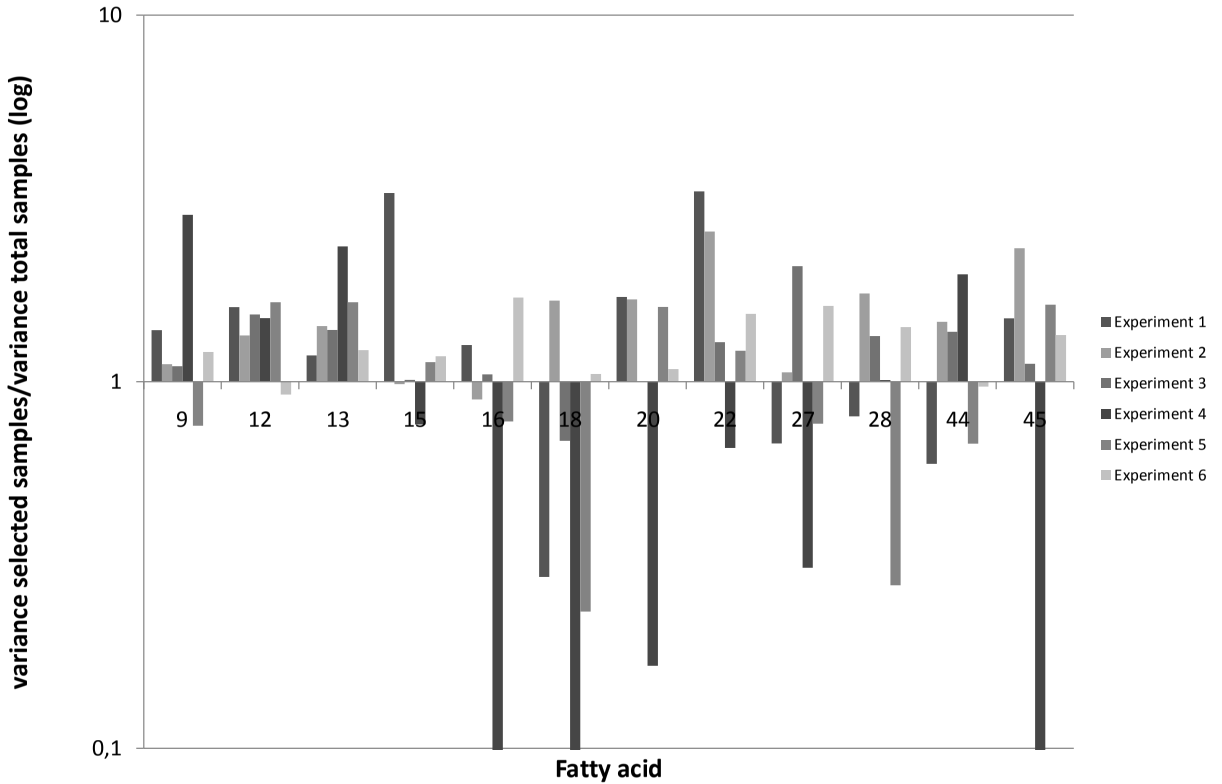


Figure 7.3: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (Kennard and Stone algorithm, Mahalanobis distance). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

Figure 7.4: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation for each fatty acid per experiment (Kennard and Stone algorithm, Mahalanobis distance). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

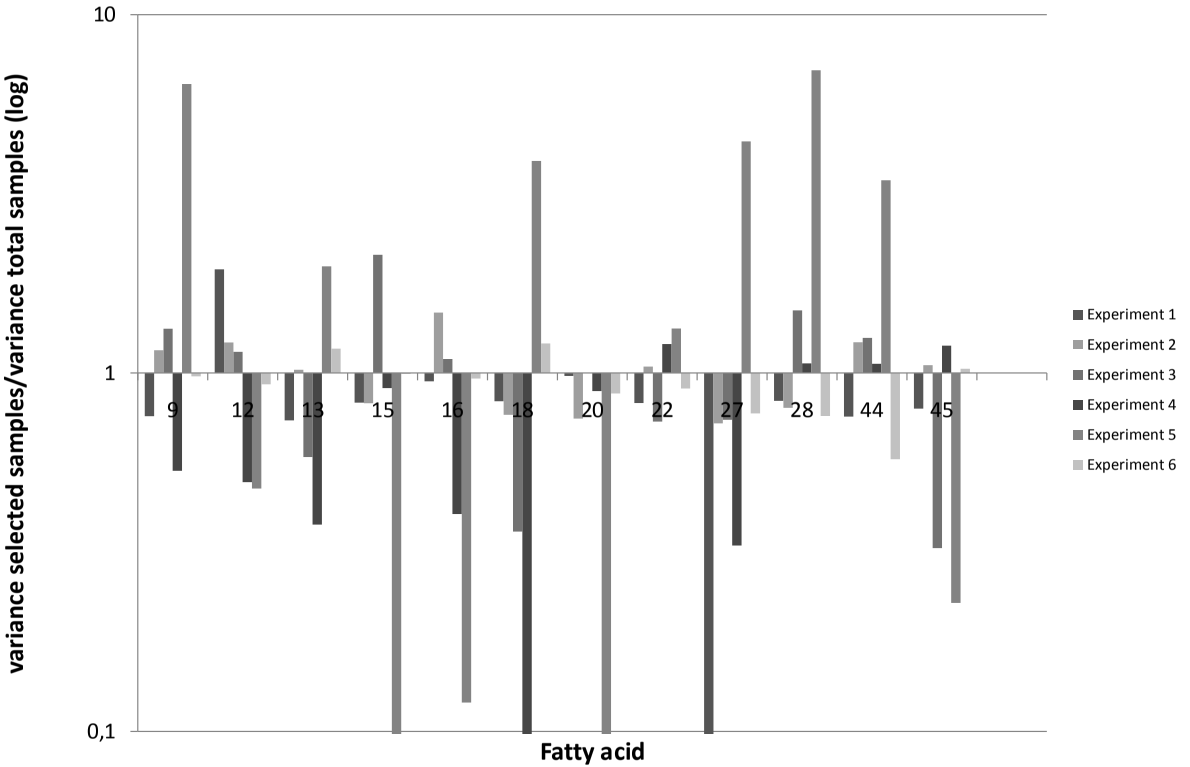


Figure 7.5: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (k-means clustering based sample selection). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

Figure 7.6: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (k-means based sample selection). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

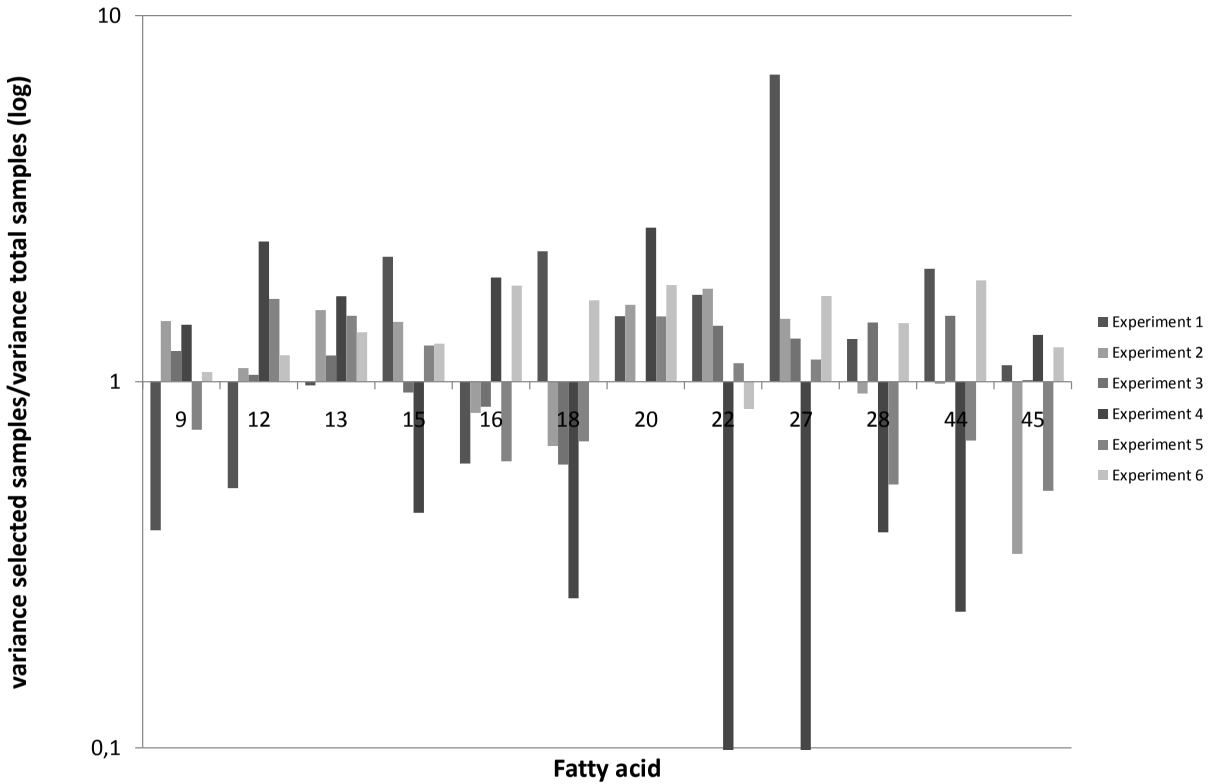


Figure 7.7: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (OptiSim clustering based sample selection). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

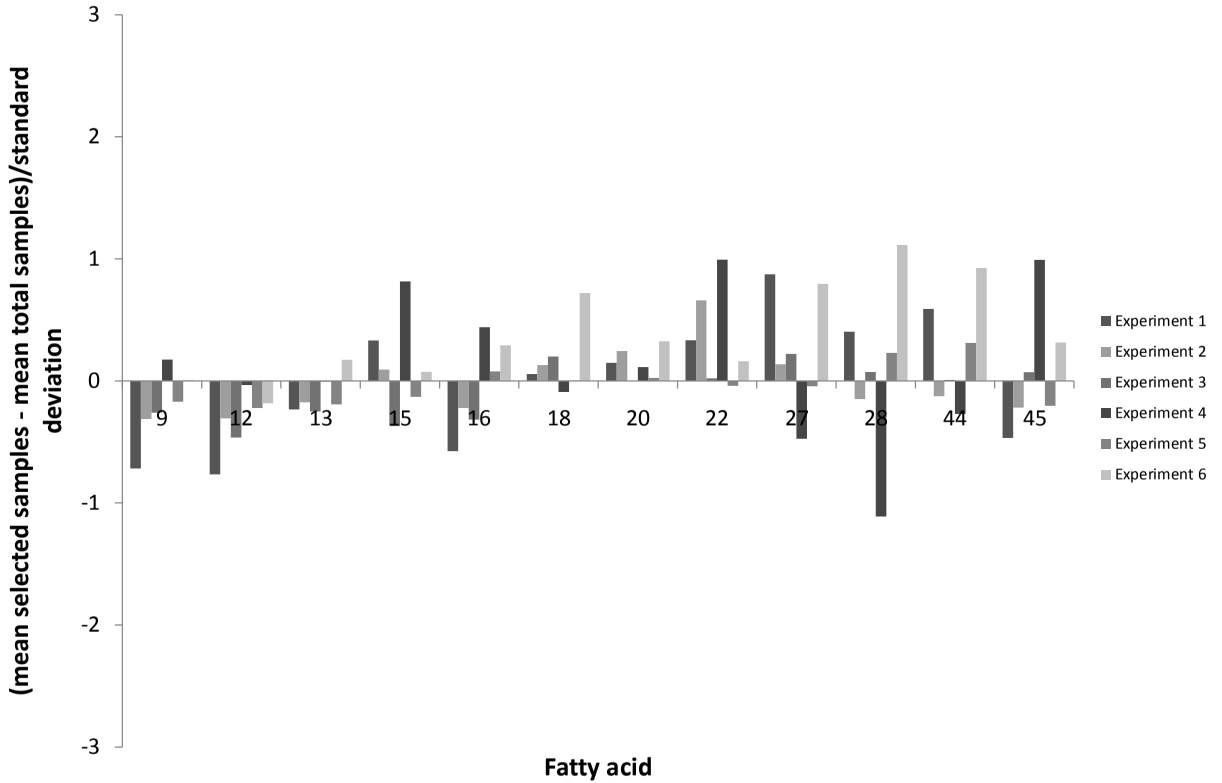


Figure 7.8: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (OptiSim based sample selection). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

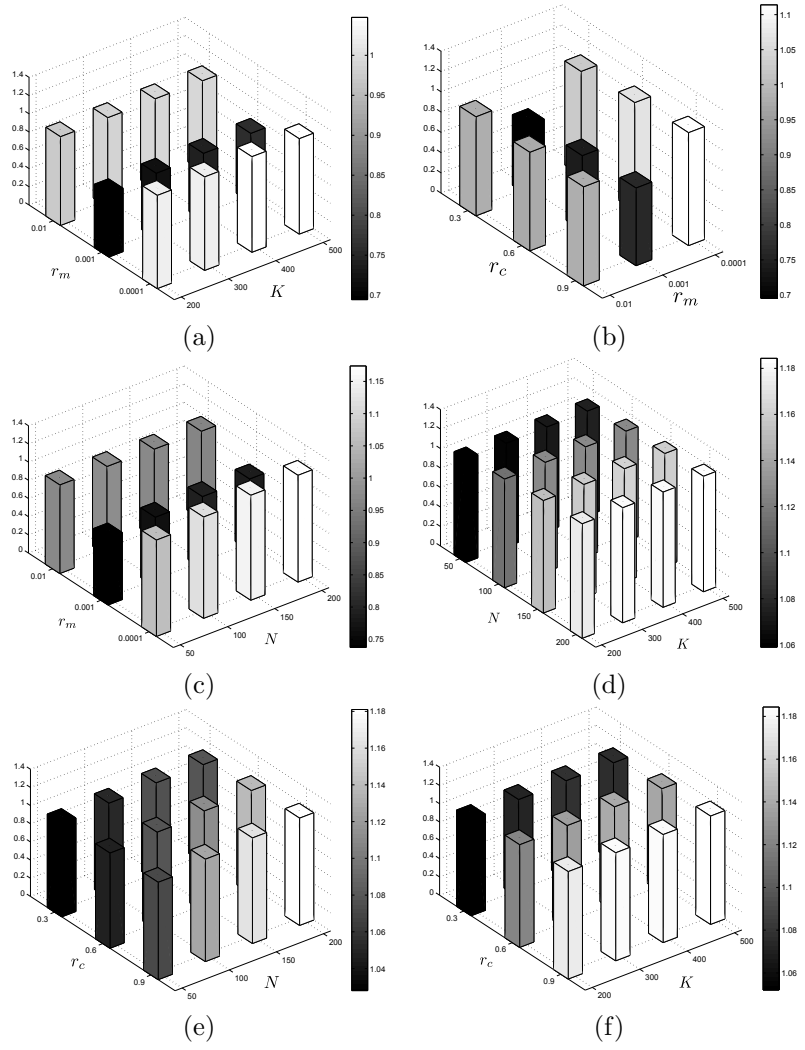


Figure 7.9: Dependence of the objective function value on different values of the GA parameters: (a) number of generations K versus mutation probability r_m with a fixed crossover probability $r_c = 0.6$ and population size $N = 50$, (b) crossover probability r_c versus mutation probability with fixed number of generations $K = 200$ and population size $N = 100$, (c) mutation probability r_m versus population size N with a fixed number of generations $K = 200$ and a fixed crossover probability $r_c = 0.9$, (d) number of generations versus population size with a fixed crossover probability $r_c = 0.9$ and a fixed mutation probability $r_m = 0.0001$, (e) crossover probability versus population size with a fixed mutation probability $r_m = 0.0001$ and number of generations $K = 400$ and (f) crossover probability versus number of generations with a fixed mutation probability $r_m = 0.0001$ and population size $N = 200$

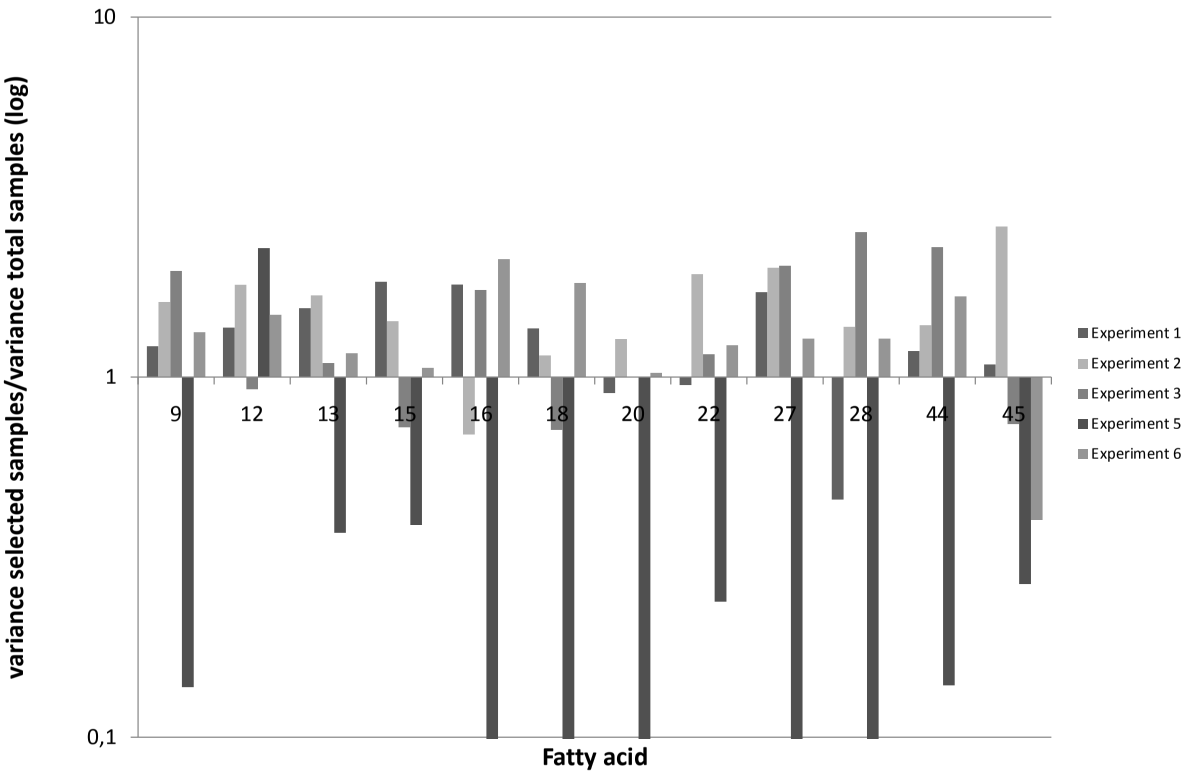


Figure 7.10: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (Genetic Algorithms (without distinction between the different experiments) (AO)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

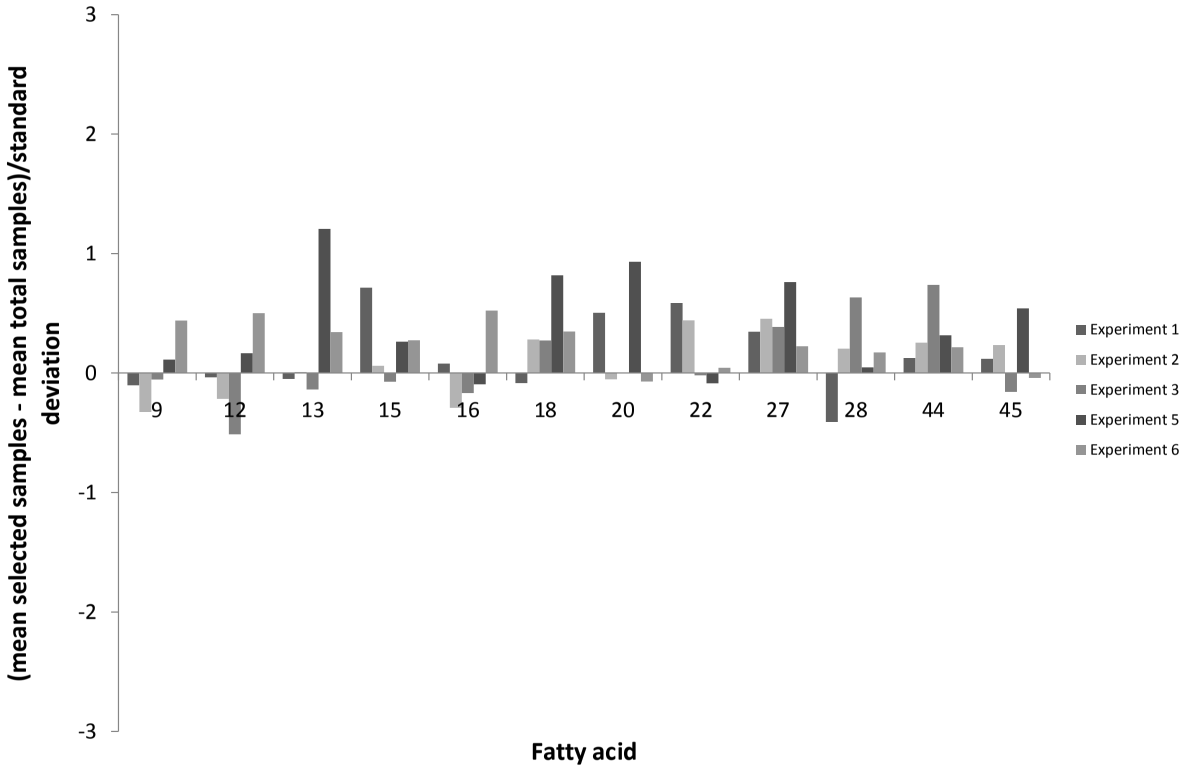


Figure 7.11: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (Genetic Algorithms (without distinction between the different experiments) (AO)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

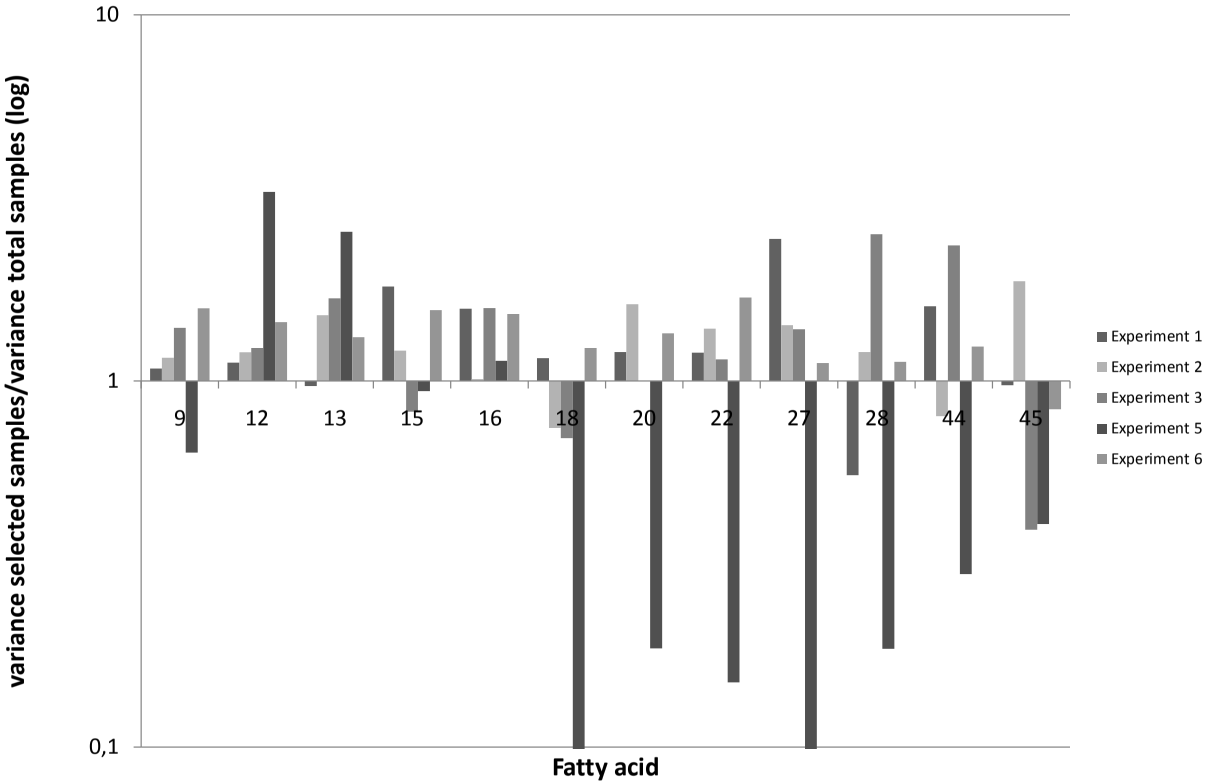


Figure 7.12: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (Genetic Algorithms (without distinction between the different experiments) (SO)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

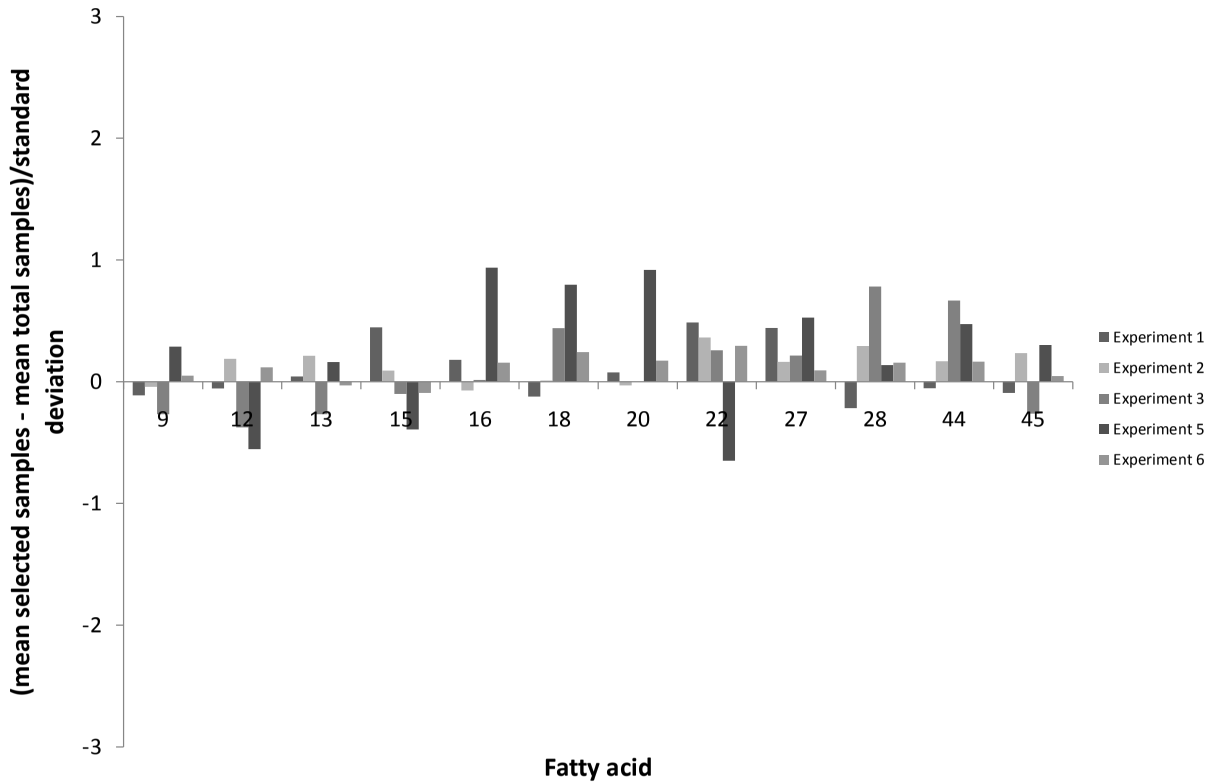


Figure 7.13: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (Genetic Algorithms (without distinction between the different experiments) (SO)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

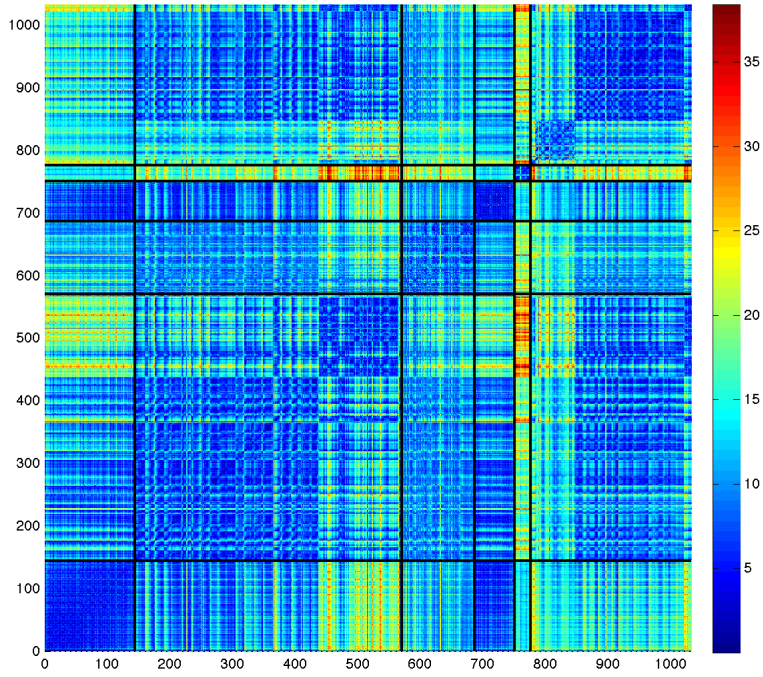


Figure 7.14: Euclidean distance between the different milk samples.

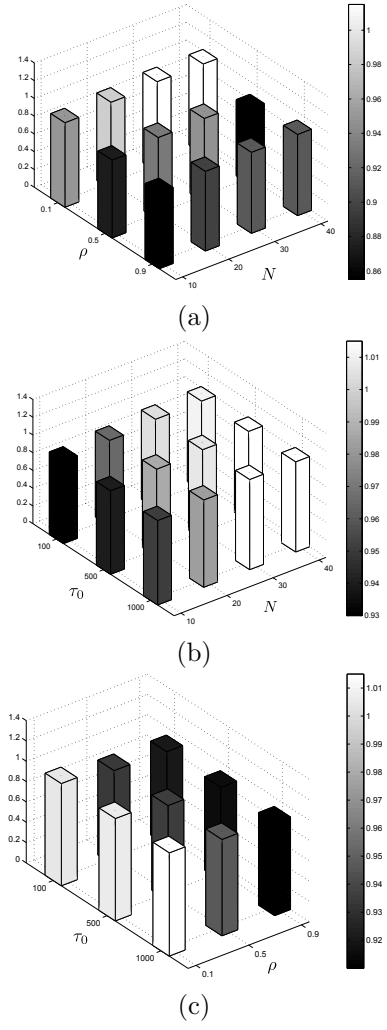


Figure 7.15: Dependence of the objective function value on different values of the ACO parameters: (a) evaporation rate ρ versus population size N with an initial pheromone concentration of $\tau_0 = 1000$, (b) population size N versus initial pheromone concentration τ_0 with a fixed evaporation rate $\rho = 0.1$, (c) evaporation rate ρ versus initial pheromone concentration τ_0 with a fixed population size $N = 30$

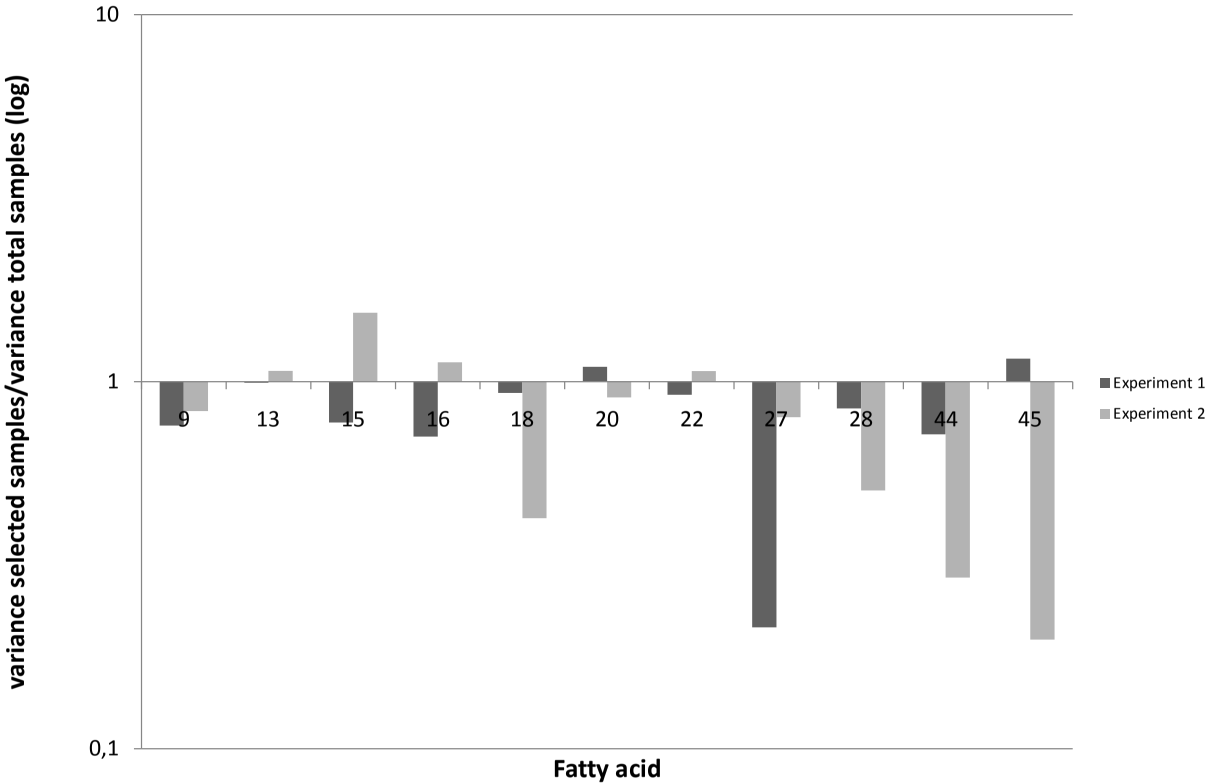


Figure 7.16: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (original AS algorithm (OA) (without distinction between the different experiments)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

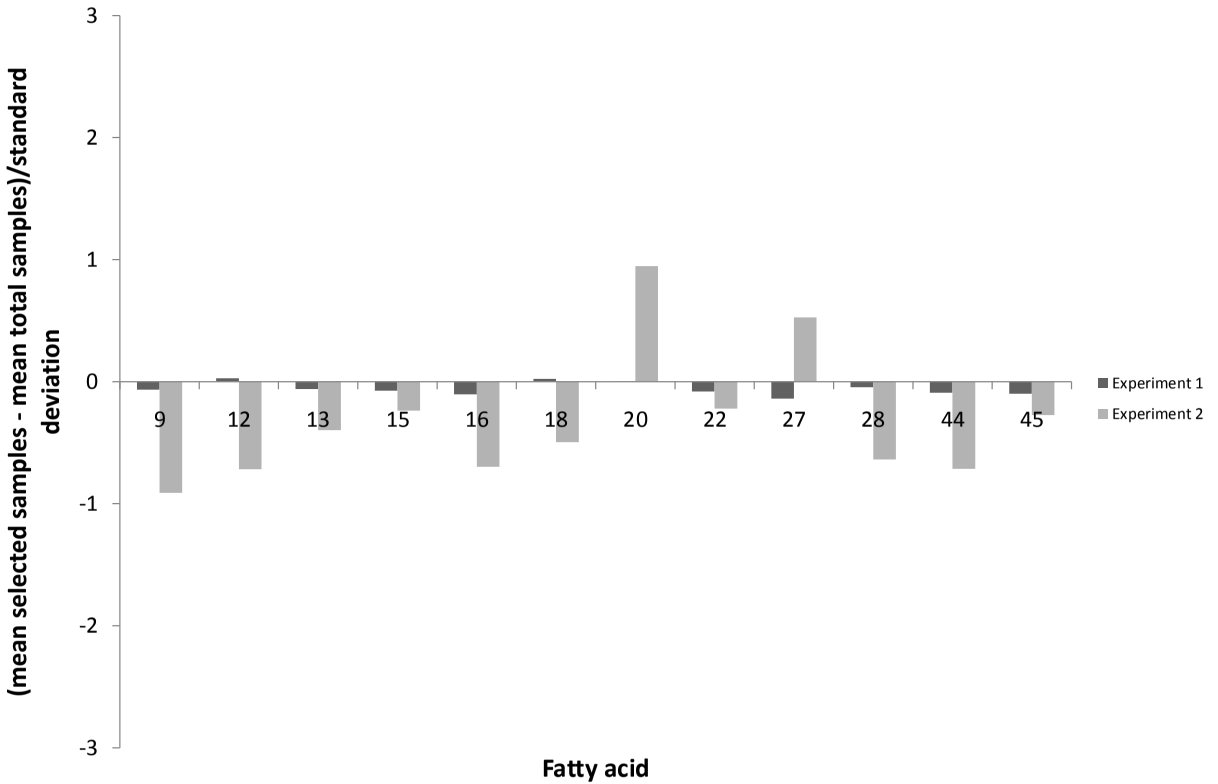


Figure 7.17: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (original AS algorithm (OA) (without distinction between the different experiments)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

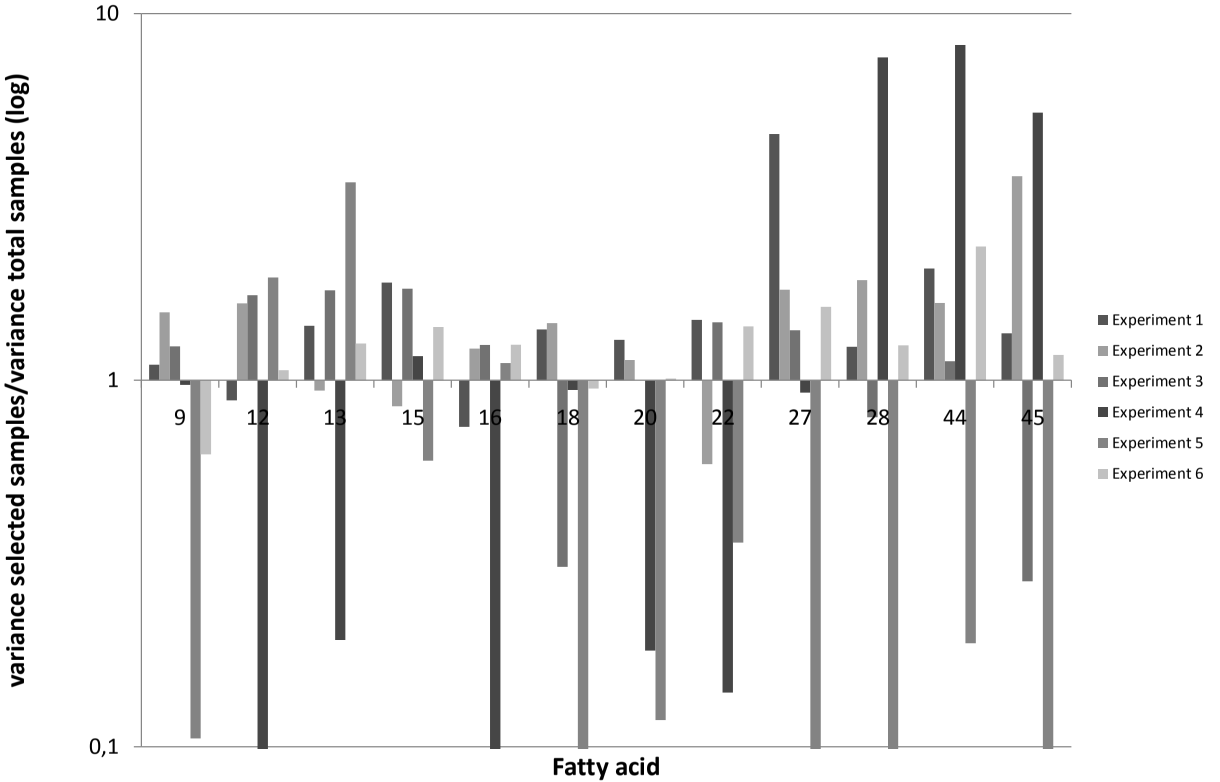


Figure 7.18: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (adapted AS algorithm (AA) (without distinction between the different experiments)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

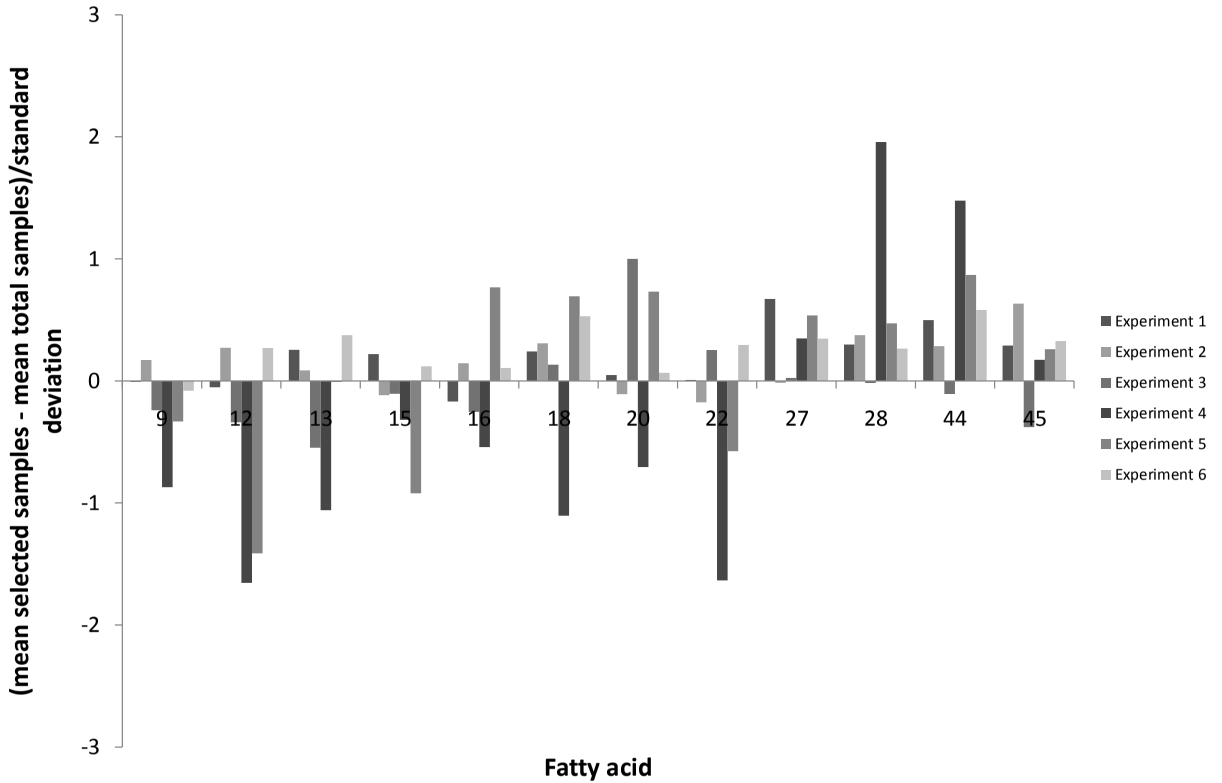


Figure 7.19: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (adapted AS algorithm (AA) (without distinction between the different experiments)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

Table 7.1: Summary of the results of selecting a subset of samples out of 1033 samples for all fatty acids. For each method the number of samples selected from each experiment is presented. The results of the Kennard and Stone algorithm are presented with the Euclidean Distance (ED) and with the Mahalanobis Distance (MD) as distance measure. For Genetic Algorithms (GA) the results of both adapted operators (AO) and standard operators (SO) are listed.

	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6	κ	κ^*
All fatty acids								
Total set of samples	144	426	117	63	26	257	1	1
ED	8	32	19	1	8	32	0.4	0
MD	14	35	15	3	9	24	0.7	0.0001
k-means	14	40	12	6	2	26	0.5	0.0081
OptiSim	6	36	18	2	7	31	0.4	0.005
GA (κ) (AO)	19	37	17	0	3	24	1.3	0
GA (κ) (SO)	19	48	18	5	5	22	1.2	0.051
AS (κ) (OA)	68	32	0	0	0	0	0	0
AS (κ) (AA)	18	37	14	2	2	27	1.1	0.00047
Priority fatty acids								
Total set of samples	144	426	117	63	26	257	1	1
ED	7	16	6	2	19	50	1.1	0.3
MD	12	35	7	1	9	36	1.1	0
k-means	15	40	12	7	1	25	0.7	0.2
OptiSim	7	17	6	0	18	52	0.8	0.2
GA (κ) (AO)	15	31	22	0	9	22	2.4	0
GA (κ) (SO)	1	4	2	0	3	2	3.4	0
AS (κ) (OA)	71	29	0	0	0	0	0	0
AS (κ) (AA)	16	33	16	1	10	24	1.7	0

7.2.3. Individual experiments

In this section, the multi-experiment aspect of the data is incorporated in the subset selection procedure. For the conventional techniques this is only possible by fixing the number of samples in advance. As it is difficult to determine the best number of selected samples per experiment, we decided to perform the conventional techniques twice, with a different number of selected samples per experiment. In the first case, the number of selected samples per experiment is chosen to be more or less equal; in the second case, the number of selected samples per experiment is chosen proportional to the size of the experiment. As mentioned in Section 6.2.1, for the Kennard and Stone algorithm, we can use either the adapted algorithm (Algorithm 2) or apply the standard algorithm independently to the different data sets corresponding to the different experiments. As the results of the Kennard and Stone algorithm with the Mahalanobis distance were similar to the results with the Euclidean distance in Section 7.2.2, we decided to only use the Euclidean distance here. In case of the algorithm based on k-means clustering and the OptiSim algorithm, we restricted to applying these algorithms on the divided data set. For Genetic Algorithms and Ant Colony Systems, we use the same parameters as in Section 7.2.2 and the fitness function κ is adapted and denoted as κ^* . The adapted fitness function was outlined in Section 6.1.

As mentioned before, the fitness function κ^* equals the minimum value displayed on the graph with the loss or gain of variance. Samples will now be selected to maximize this minimal value.

The Kennard and Stone algorithm

The adapted algorithm, Euclidean distance (ED-adapted): Firstly, the number of selected samples per experiment is kept more or less equal (ED-adapted-1). Figure 7.20 illustrates that, even though the different experiments are taken into account, the variances of some fatty acids are smaller for the selected subset than for the total data set. This is also quantified by the value κ^* in Table 7.2. Figure 7.21 shows that for most fatty acids, the deviation between the mean for the selected subset and the total data set is rather small.

Secondly, the predetermined number of samples per experiment is chosen proportional to the size of the experiment (ED-adapted-2). The value of κ^* in Table 7.2 indicates that for some fatty acids there is a strong loss of variance. Important to notice is that the value of κ^* is much lower now than in the previous case with an equal number of selected samples per experiment. This lower value of κ^* was caused by the strong loss of variance of the C18:1 t11 fatty acid. We do not display the corresponding figure, as it looks similar to the graph in Figure 7.20 (with exception of the C18:1 t11 fatty acid). The deviations from the means are similar to those represented in Figure 7.21.

The standard algorithm, Euclidean distance (ED): For an equal number of samples per experiment (ED-1), Figure 7.22 shows that only for a few fatty acids there is a loss of variance. This loss of variance is also illustrated by the value of κ^* in Table 7.2. The deviations from the means (Figure 7.23) are rather small and similar to the graph in Figure 7.21.

When the number of selected samples per experiment is chosen proportional to the size of the experiment (ED-2), the variances of some fatty acids are much lower for the selected subset than for the total data set, indicated by the value of κ^* in Table 7.2 (Figure 7.24). This value is much lower than for the method ED-1, precisely as was the case for method ED-adapted. The deviations from the means are larger than for the method ED-1 (Figure 7.25).

The k-means clustering based algorithm

Figures 7.26 and 7.28 illustrate that both in the case of an equal number of selected samples per experiment (k-means-1) as well as in the case of a number of samples selected proportionally to the number of samples of each experiment (k-means-2), there is a strong loss of variance for the selected subset with respect to the total data set (Table 7.2, κ^*). In both cases, the deviations from the means are very small (Figures 7.27 and 7.29).

The OptiSim algorithm

When the predetermined number of samples per experiment is proportional to the size of the experiment (Figure 7.32), there is a stronger loss of variance than when the number of samples per experiment is equal (Figure 7.30). This can also be noticed in Table 7.2 (κ^*). The deviations from the means are small and similar in both cases (Figures 7.31 and 7.33).

Genetic Algorithms

Fitness function κ^* (adapted operators): In the function κ^*, the ratios are restricted to the data per experiment separately. The minimum is now computed over both the n variables and the six different experiments. The results (Figures 7.34 and 7.35, Table 7.2) indicate that samples from all experiments are selected, roughly proportional to the size of the experiments. It is important that this can be supervised by means of a good fitness function. The variance of each fatty acid is for the selected subset greater than or more or less equal to the variance of the corresponding fatty acid for the total data set. Since this was exactly our objective, these results show that a collective maximization of these values is possible. This was not the case with the conventional techniques. With the standard Kennard

and Stone algorithm with an equal number of samples per experiment (ED-1) (Figure 7.22), however, almost as good results were obtained for most of the fatty acids. Nevertheless, the requirement for fixing the number of selected samples per experiment in advance, is a major drawback of the Kennard and Stone algorithm. The number of samples that have to be selected from each experiment, in order to obtain a maximal variance, depends on many factors and is therefore not straightforward to determine. Figure 7.35 illustrates that the deviations from the means are rather small, which is a second confirmation that Genetic Algorithms are capable of selecting an optimal subset of samples.

Fitness function κ^* (standard operators): Table 7.2 indicates that 275 samples are selected by the Genetic Algorithm to optimize the fitness function κ^*. However, the difference in the value of κ^* obtained with the adapted operators is rather small, which illustrates that the extra 175 samples do not contribute significantly to the total variability. These results are also illustrated in Figures 7.36 and 7.37.

Ant Colony Optimization

Fitness function κ^* (original AS algorithm): Figures 7.38 and 7.39 illustrate that only samples selected from the first two experiments are selected. The deviations from the means for this subset is rather small but there is a strong loss of variance when the selected subset is used. This indicates that the AS algorithm without the adapted probabilistic rule and the adapted pheromone update rule is not capable of selecting a subset out of a multi-experiment data set.

Fitness function κ^* (adapted AS algorithm): Table 7.2 shows that the number of samples selected from each experiment is more or less proportional to the size of the experiment. Figure 7.41 illustrates that only a small loss of variance is present when the selected subset of samples is used, which is also indicated by the value of κ^* in Table 7.2. The deviations from the means are rather small but larger than when Genetic Algorithms are used to select the optimal subset of samples (Figure 7.41). This indicates that the adapted probabilistic decision rule and the adapted pheromone update rule result in a better performance of the AS algorithm for this particular subset selection problem.

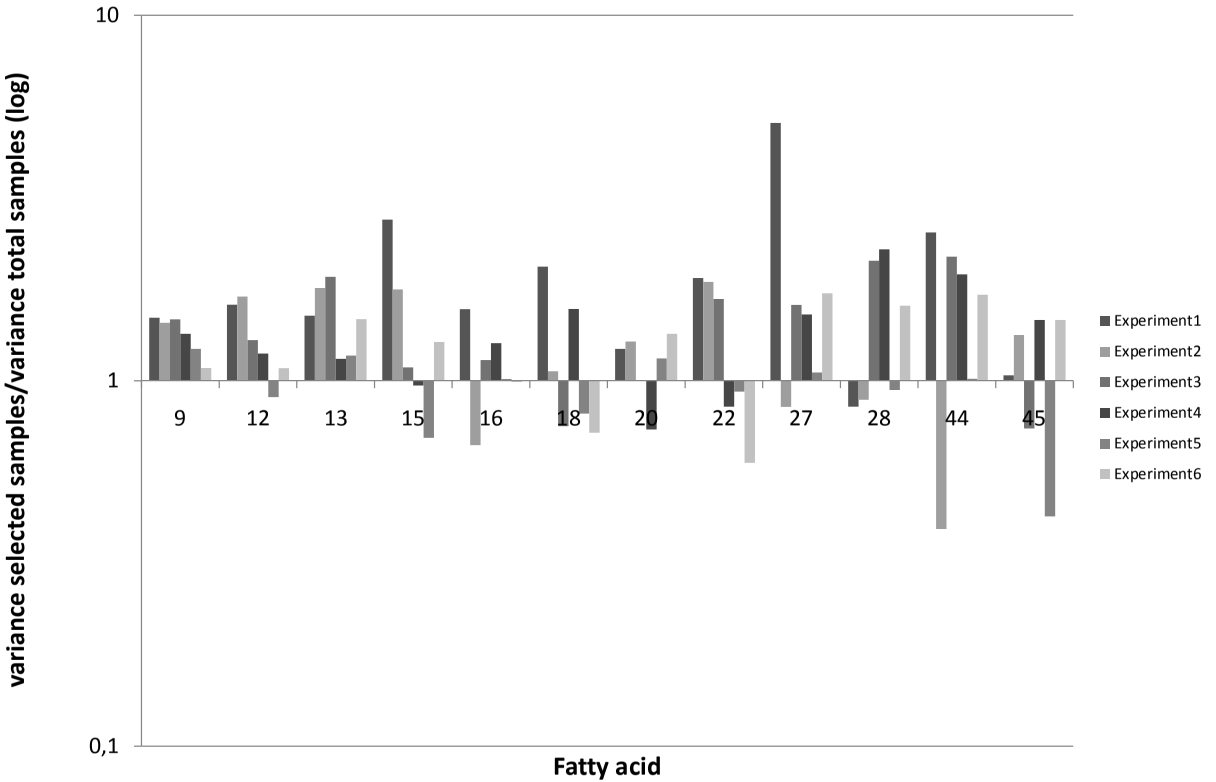


Figure 7.20: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (Adapted Kennard and Stone algorithm, Euclidean distance, equal number of samples per experiment (ED-adapted-1)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

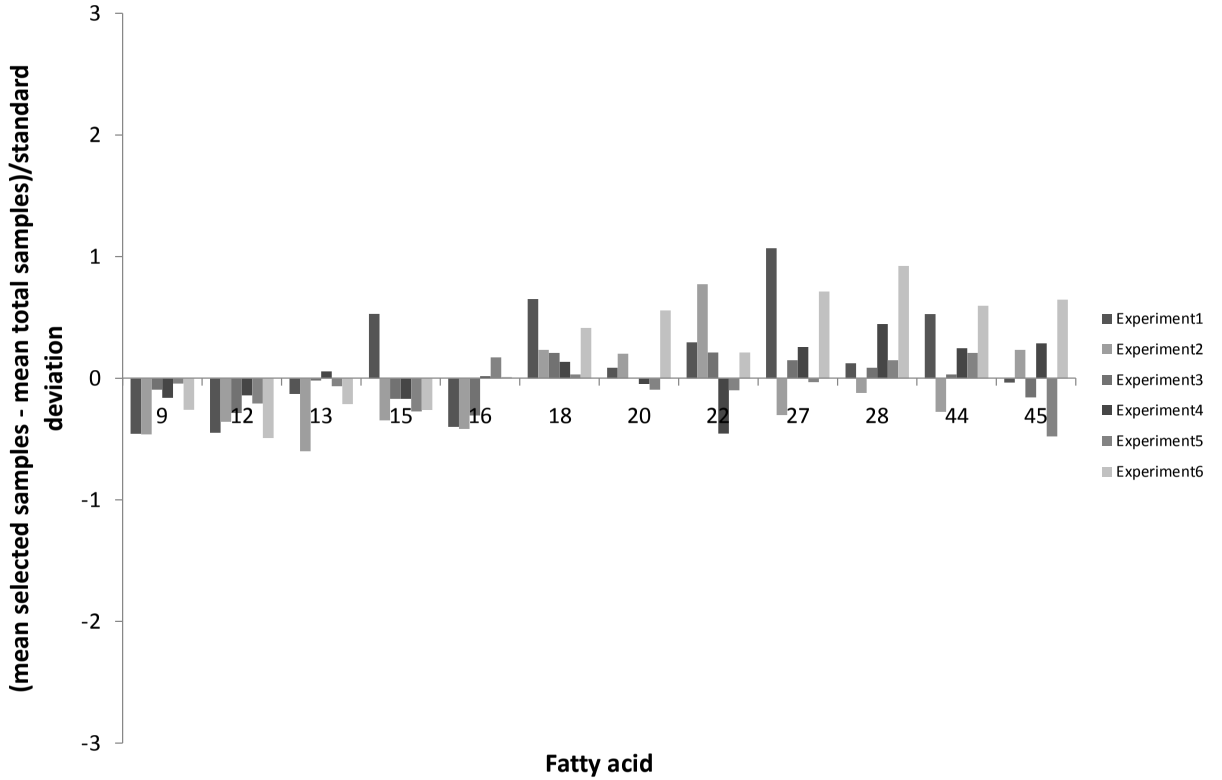


Figure 7.21: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (Adapted Kennard and Stone algorithm, Euclidean distance, equal number of samples per experiment (ED-adapted-1)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

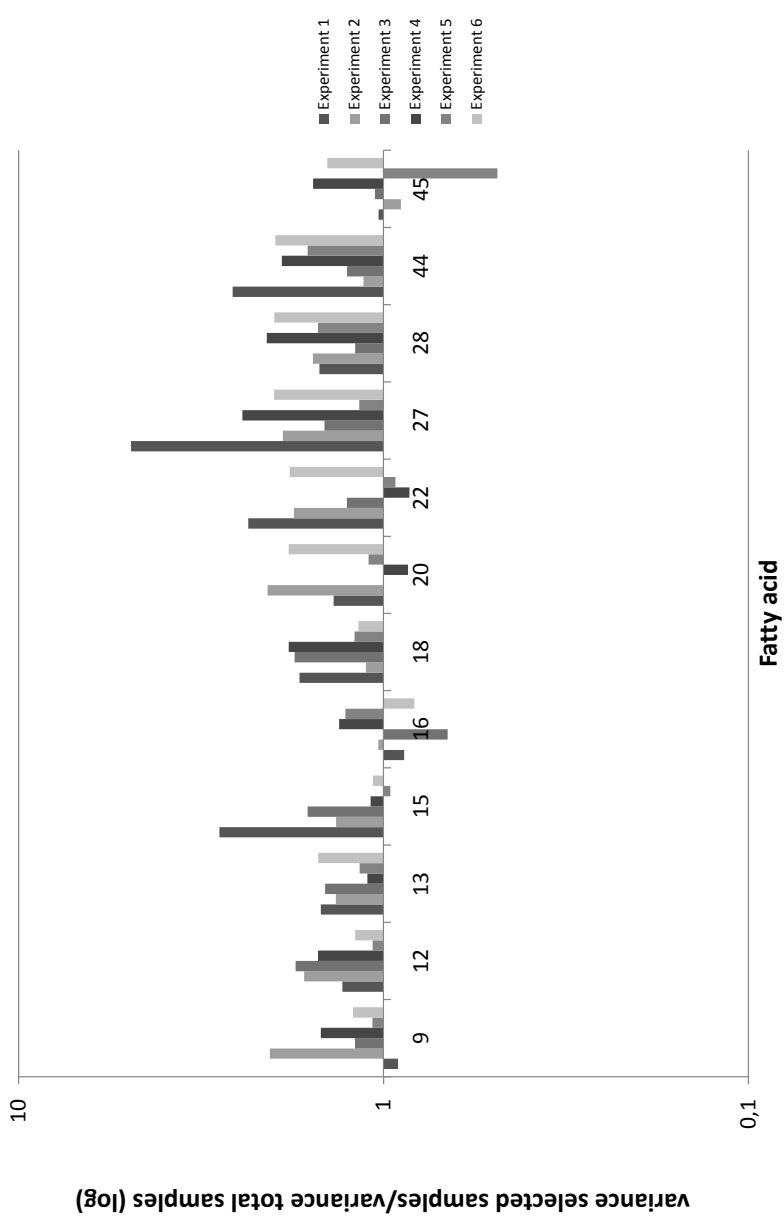


Figure 7.22: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (Standard Kennard and Stone algorithm, Euclidean distance, equal number of samples per experiment (ED-1)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

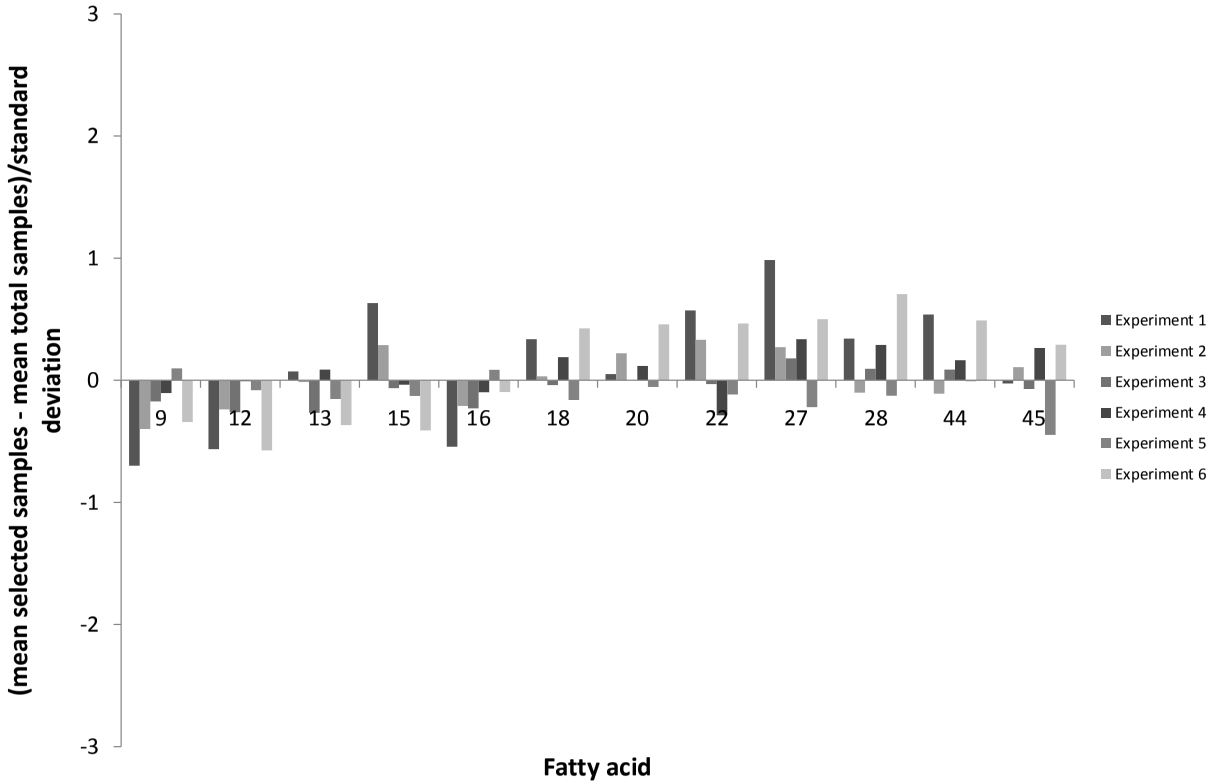


Figure 7.23: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (Standard Kennard and Stone algorithm, Euclidean distance, equal number of samples per experiment (ED-1)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

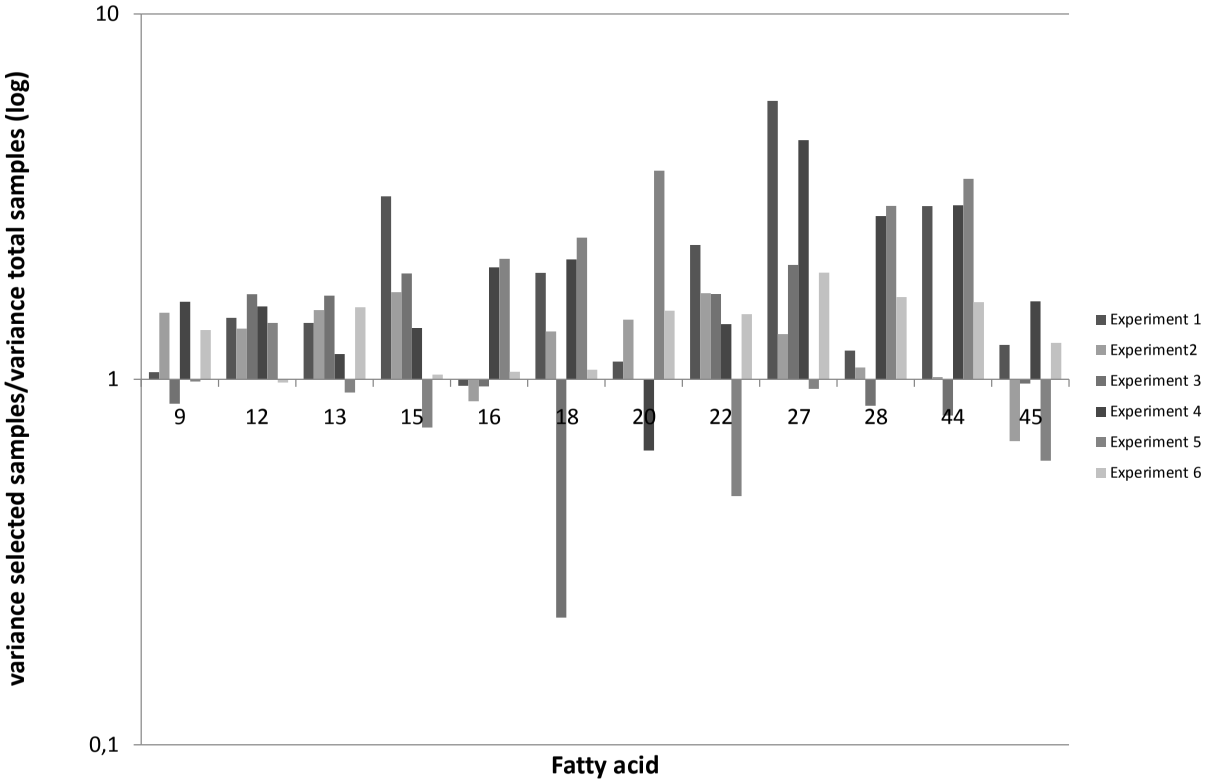


Figure 7.24: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (Standard Kennard and Stone algorithm, Euclidean distance, number of samples per experiment proportional with the size of the experiment (ED-2)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

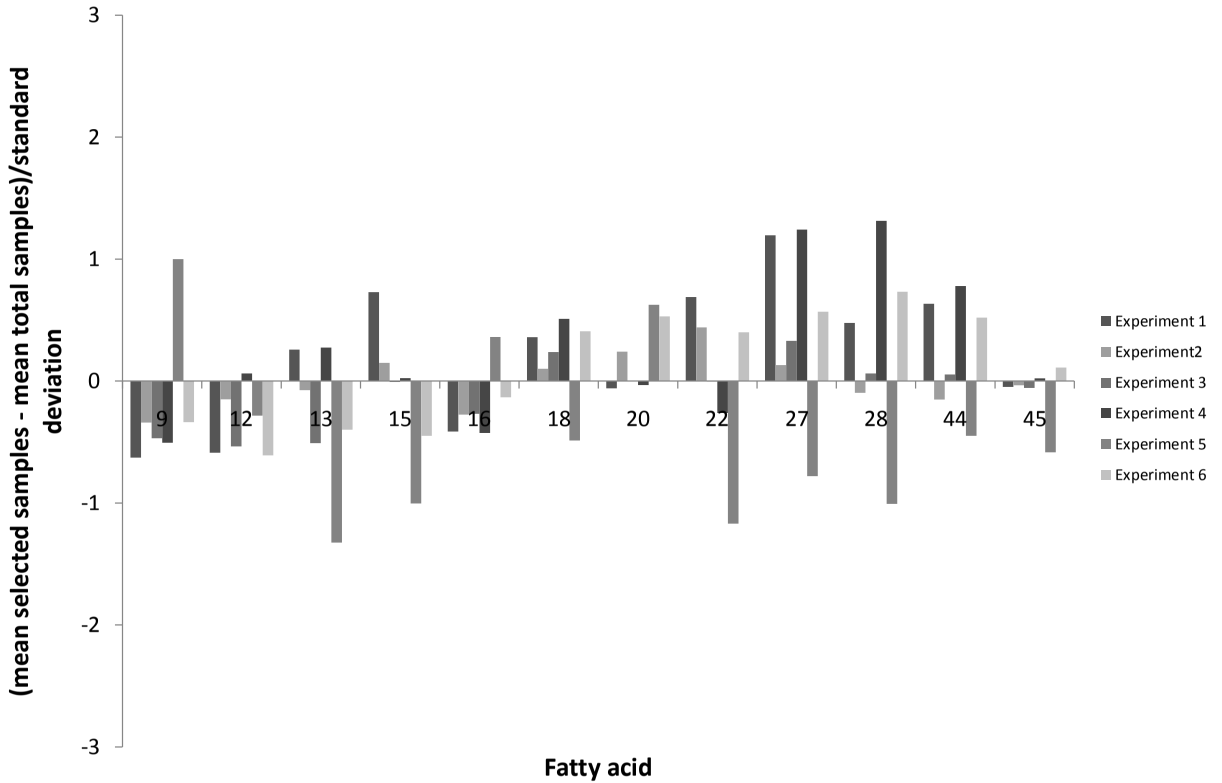


Figure 7.25: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (Standard Kennard and Stone algorithm, Euclidean distance, number of samples per experiment proportional with the size of the experiment (ED-2)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

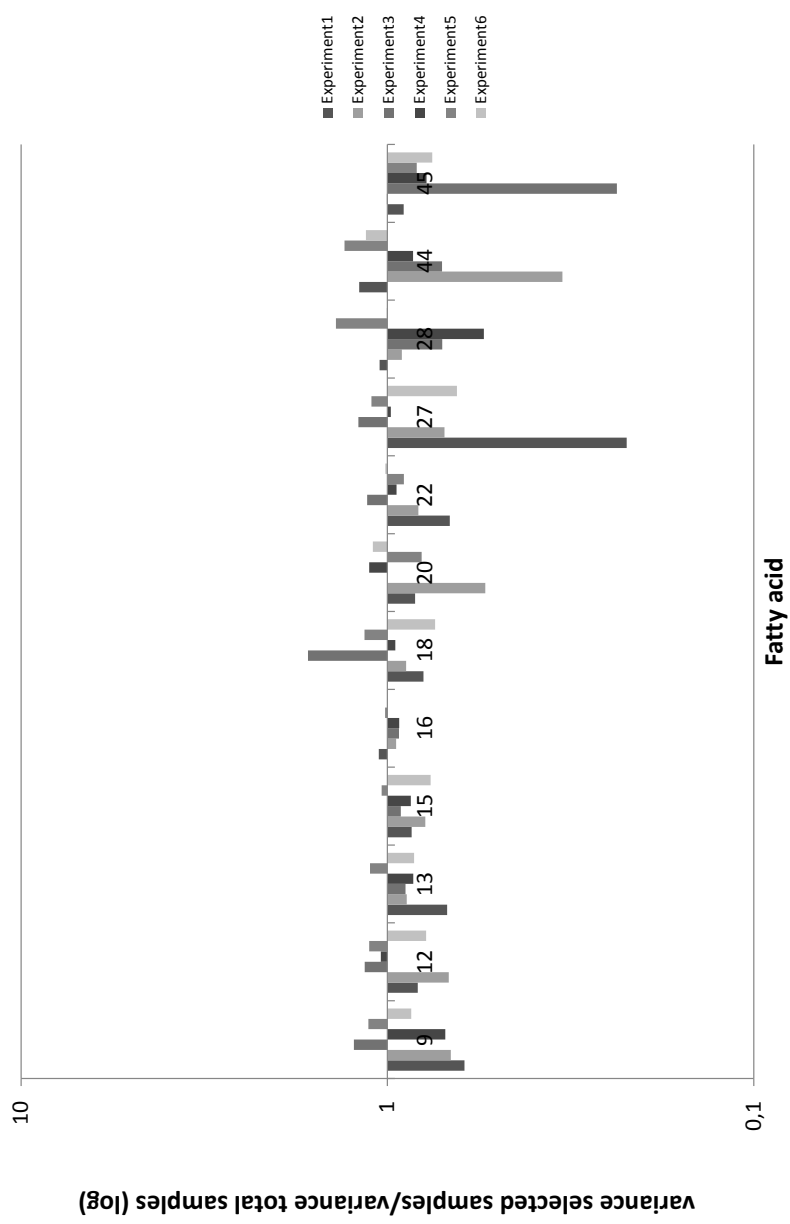


Figure 7.26: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (k-means-1, equal number of samples per experiment). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

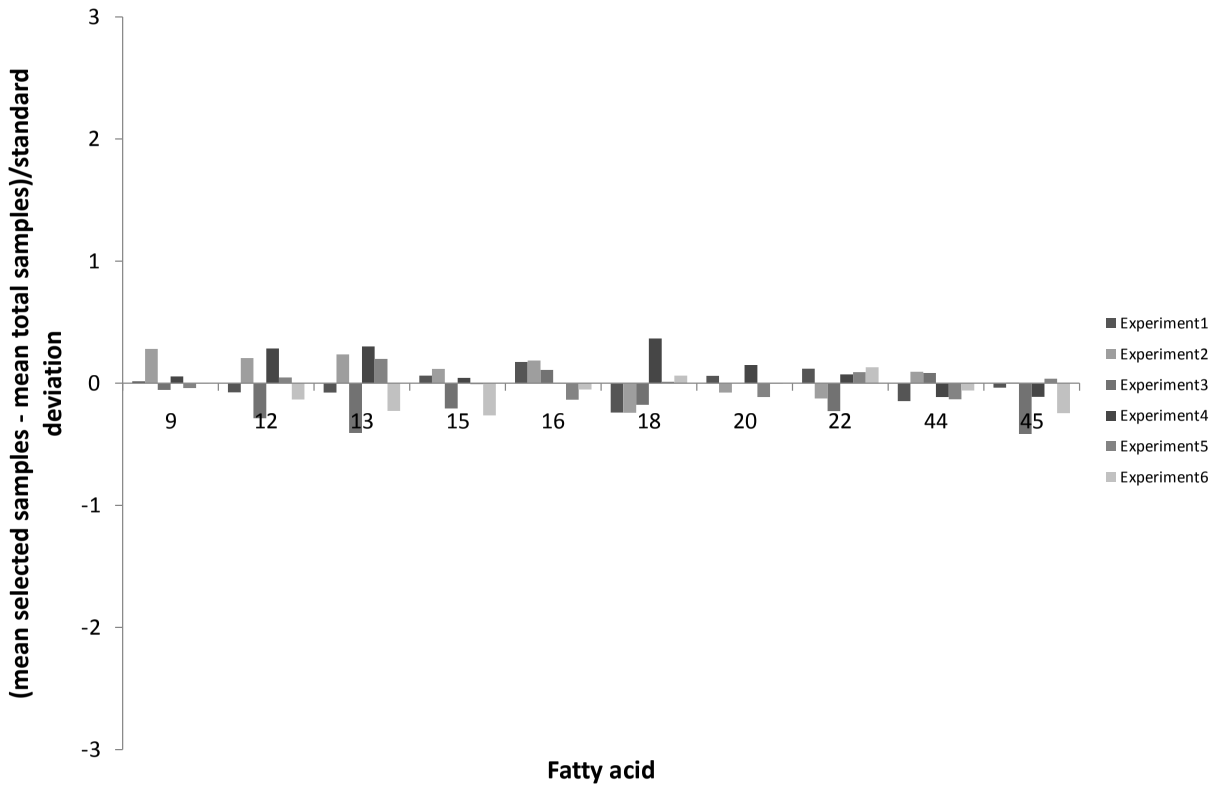


Figure 7.27: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment k-means-1, equal number of samples per experiment). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

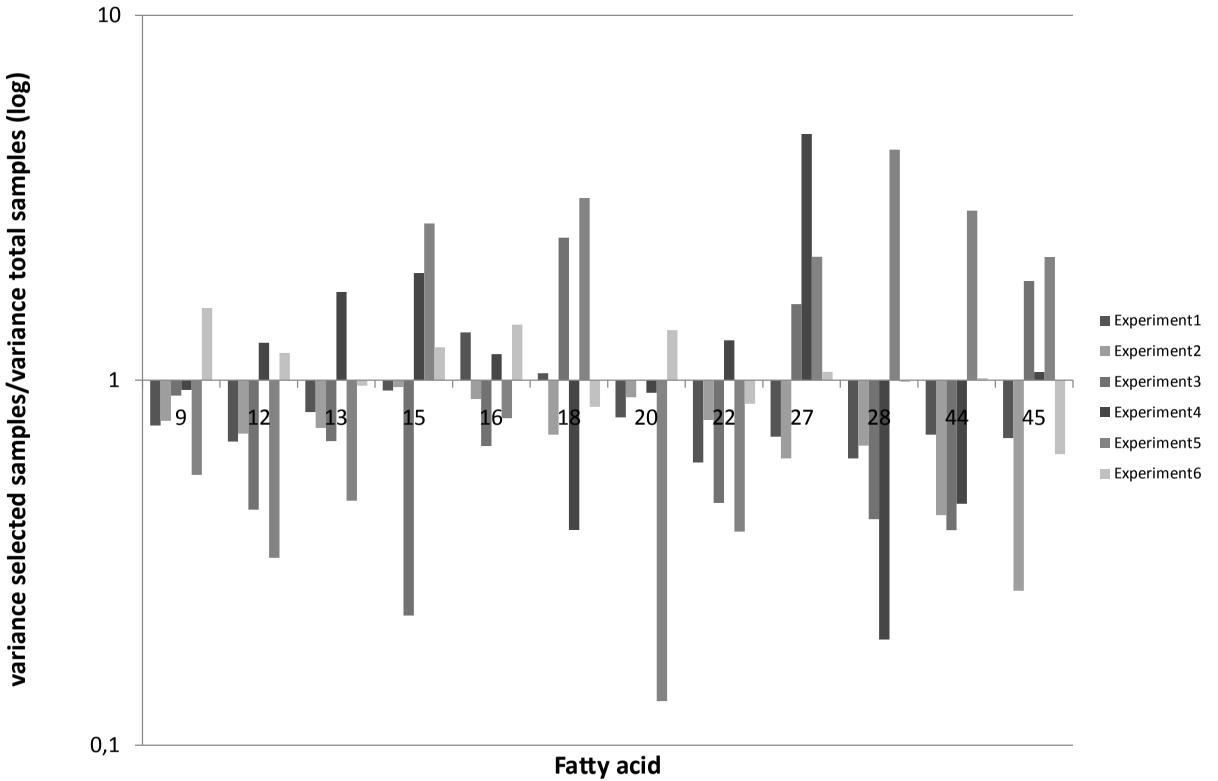


Figure 7.28: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (k-means-2, number of samples per experiment proportional with the size of the experiment). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

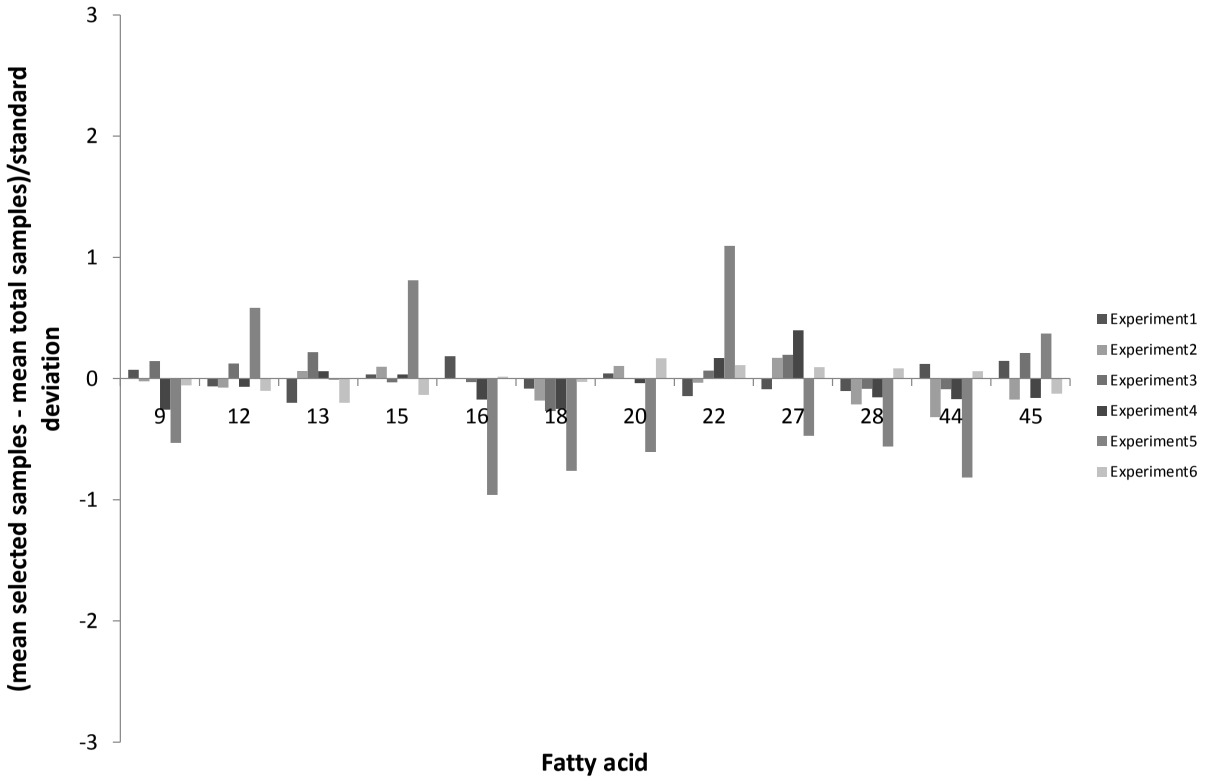


Figure 7.29: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (k-means-2, number of samples per experiment proportional with the size of the experiment). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

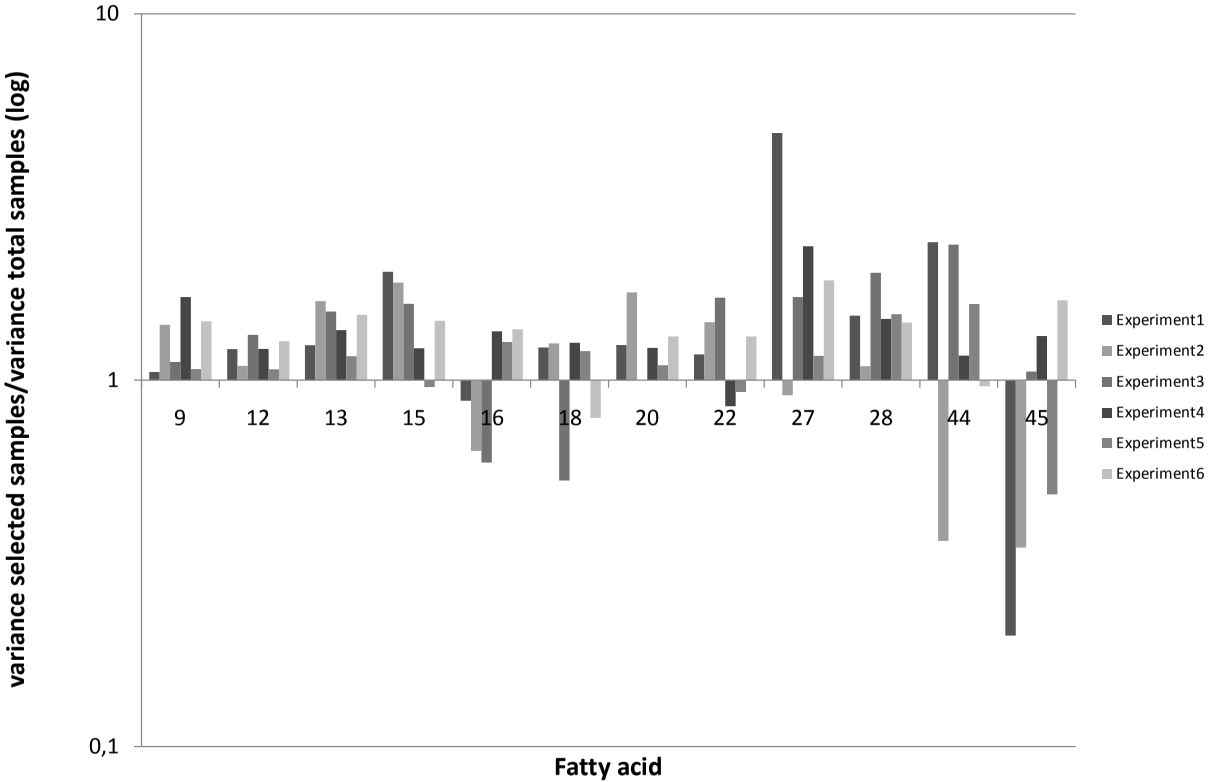


Figure 7.30: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (OptiSim-1, equal number of samples per experiment). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

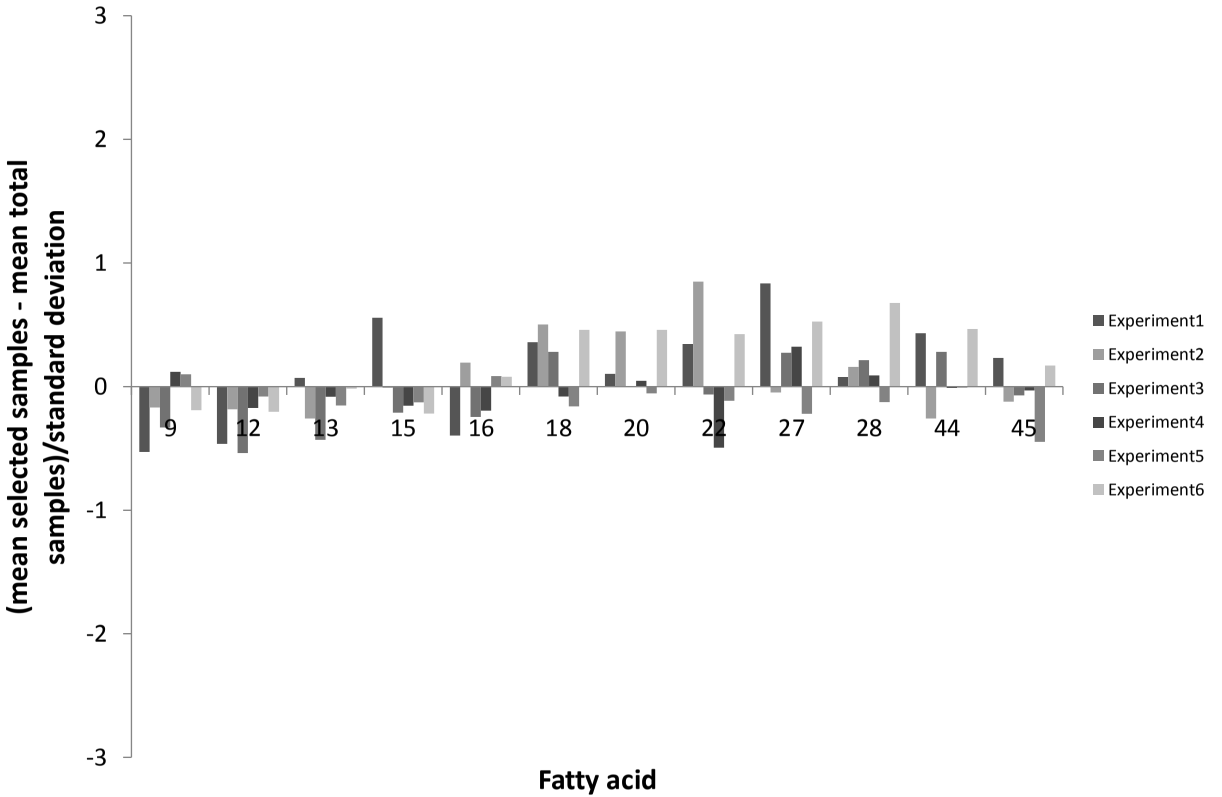


Figure 7.31: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (OptiSim-1, equal number of samples per experiment). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

Figure 7.32: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (OptiSim-2, number of samples per experiment proportional with the size of the experiment). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

Figure 7.33: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (OptiSim-2, number of samples per experiment proportional with the size of the experiment). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

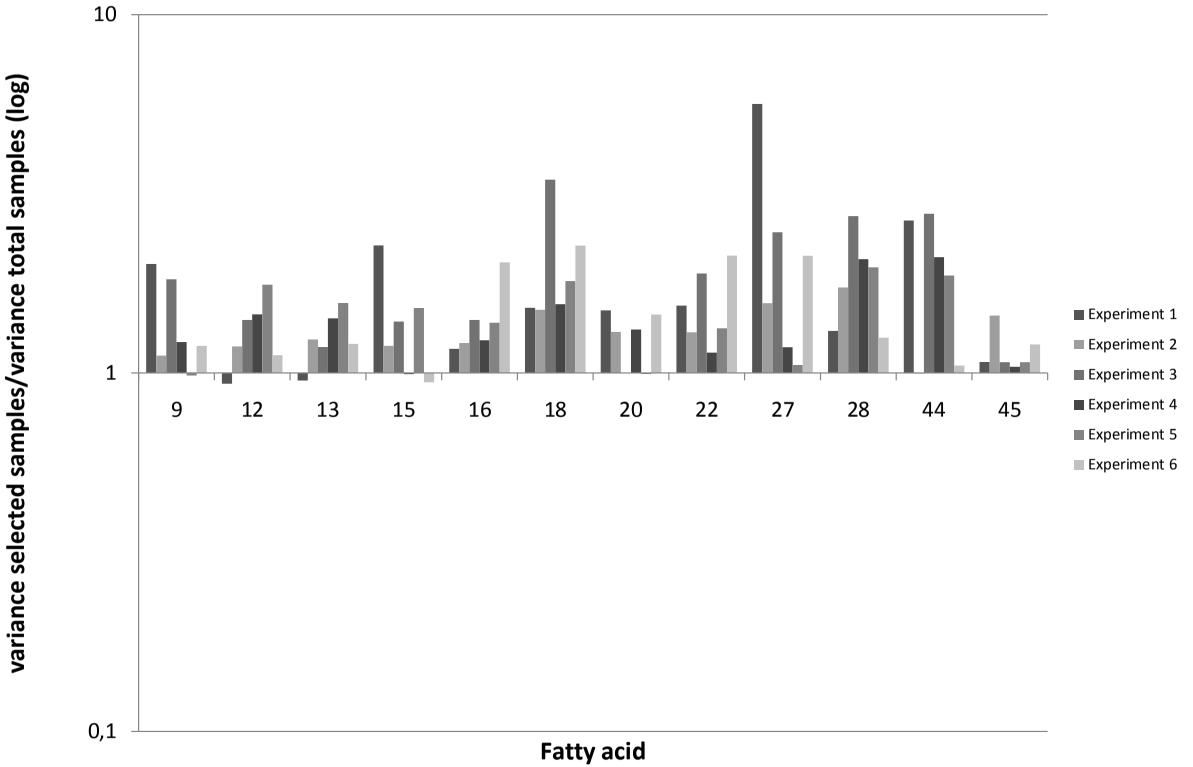


Figure 7.34: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (Genetic Algorithms, method 1 (individual experiments) (AO)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

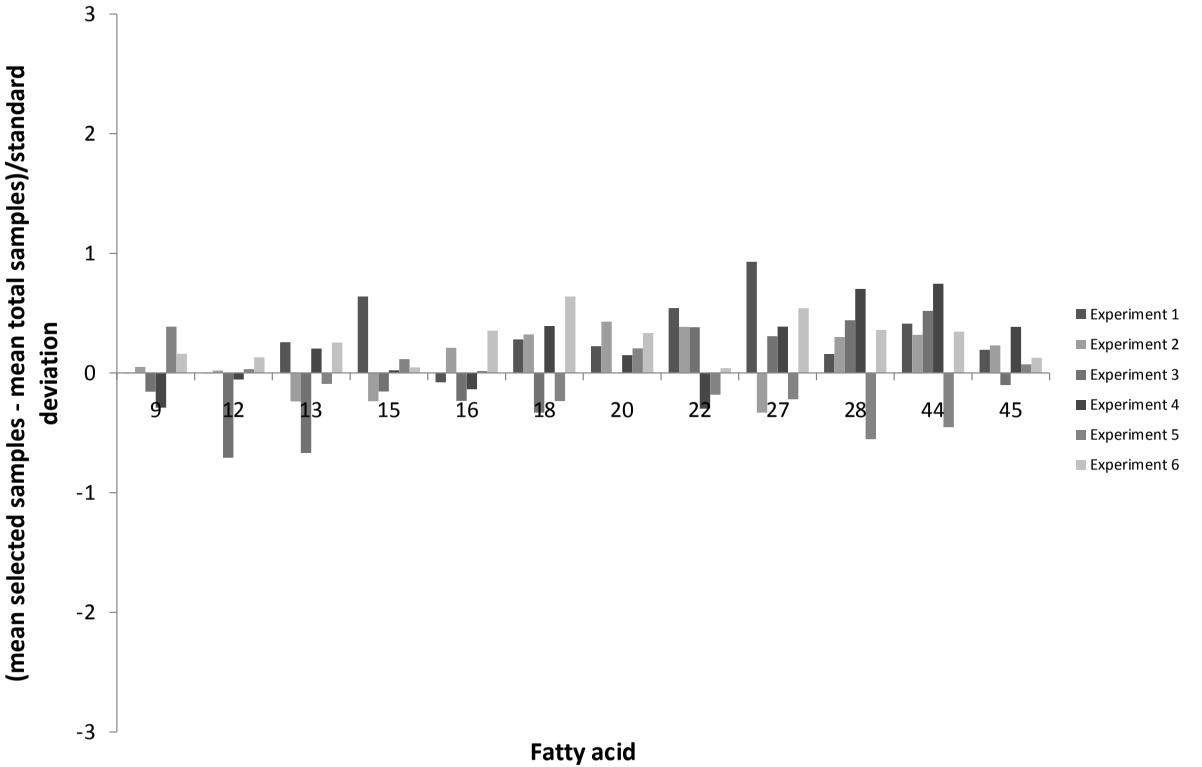


Figure 7.35: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (Genetic Algorithms, method 1 (individual experiments) (AO)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

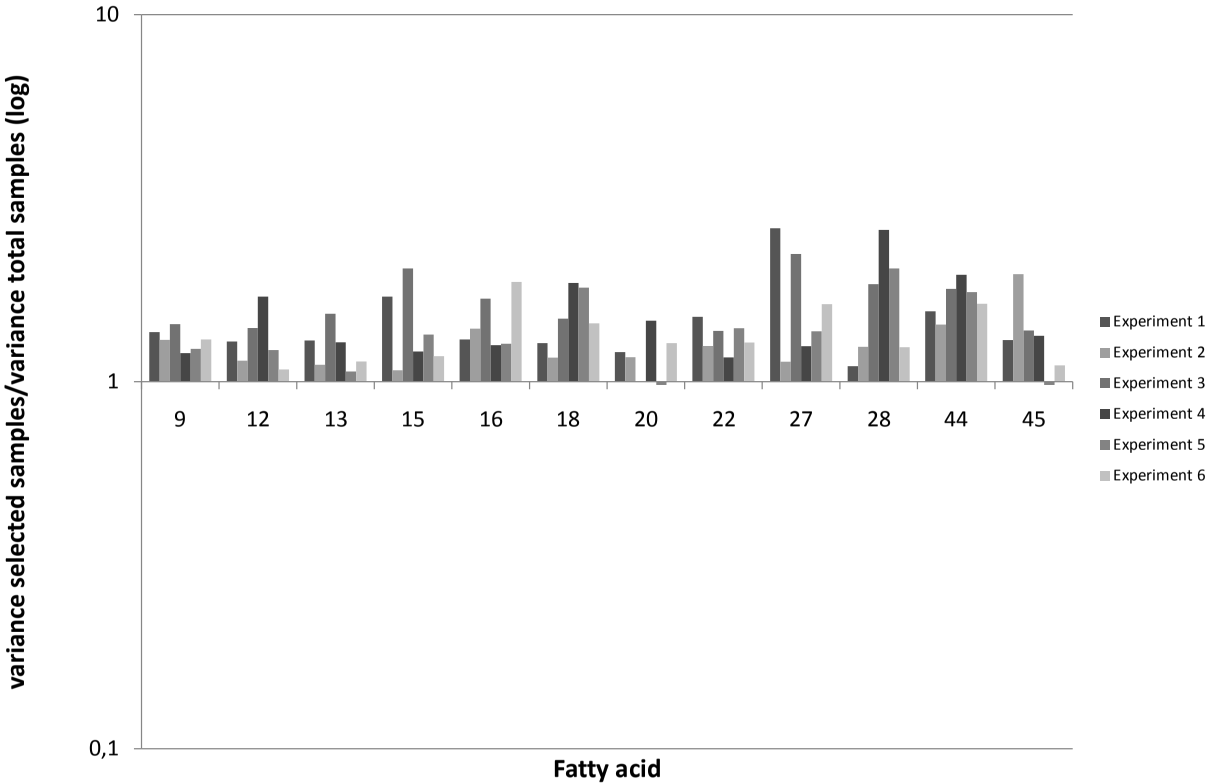


Figure 7.36: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (Genetic Algorithms, method 1 (individual experiments) (SO)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

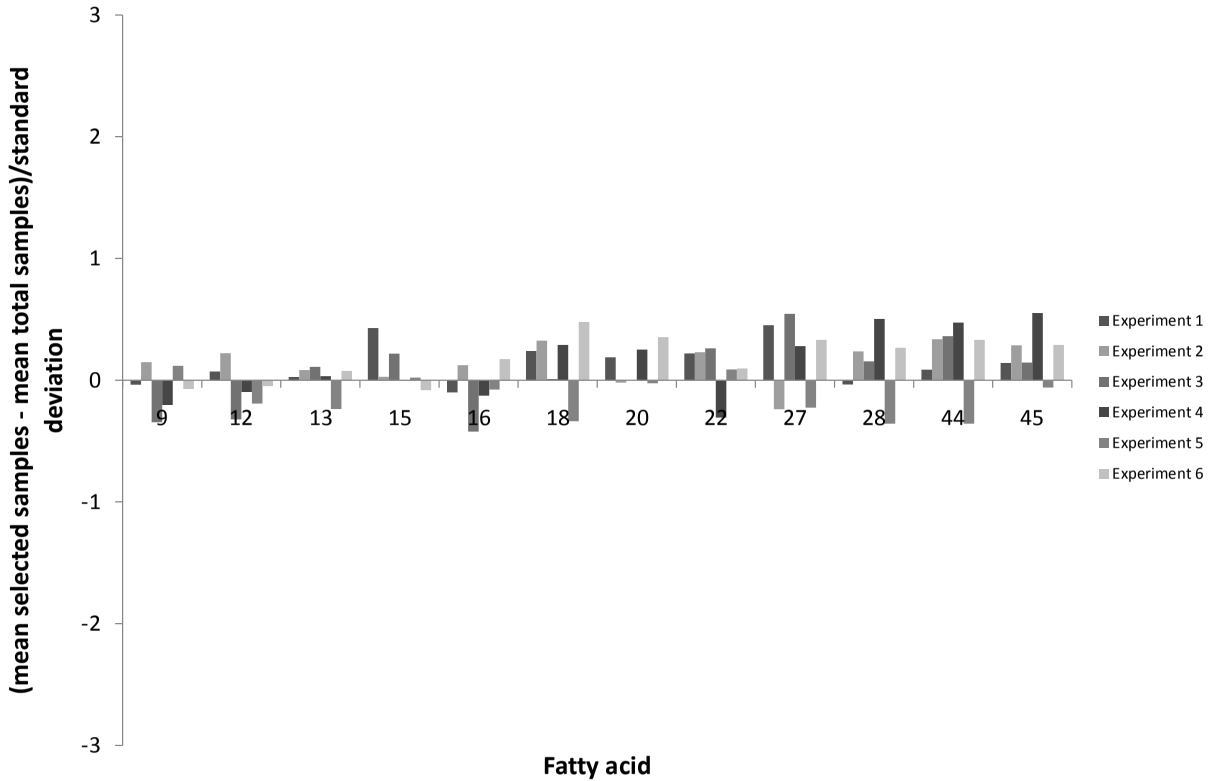


Figure 7.37: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (Genetic Algorithms, method 1 (individual experiments) (SO)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

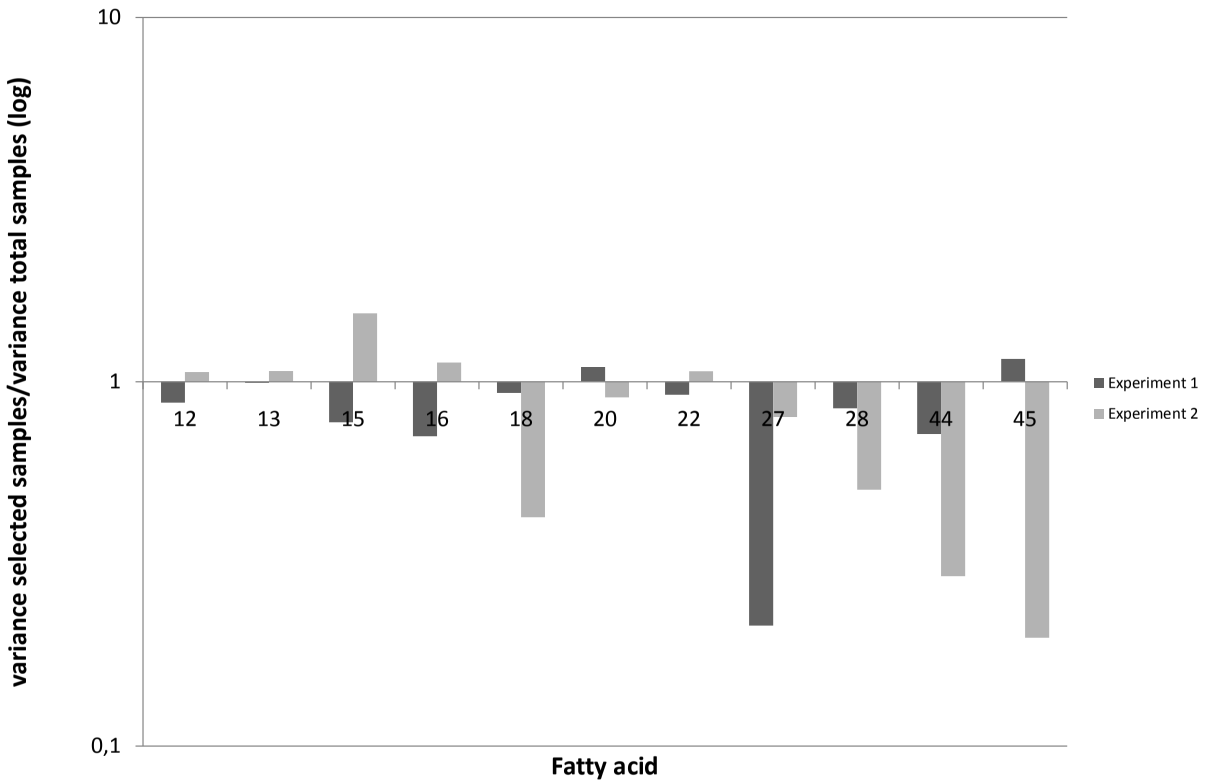


Figure 7.38: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (original AS algorithm (OA) (individual experiments)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

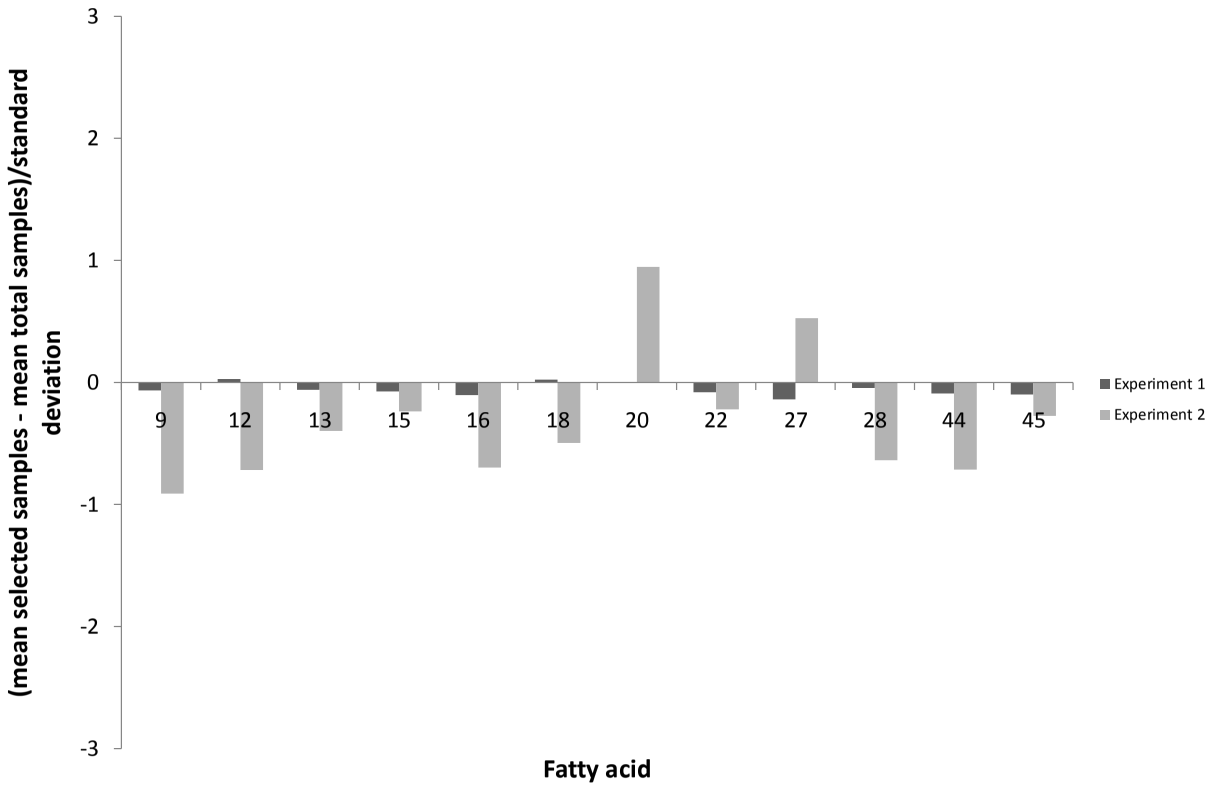


Figure 7.39: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (original AS algorithm (OA) (individual experiments)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

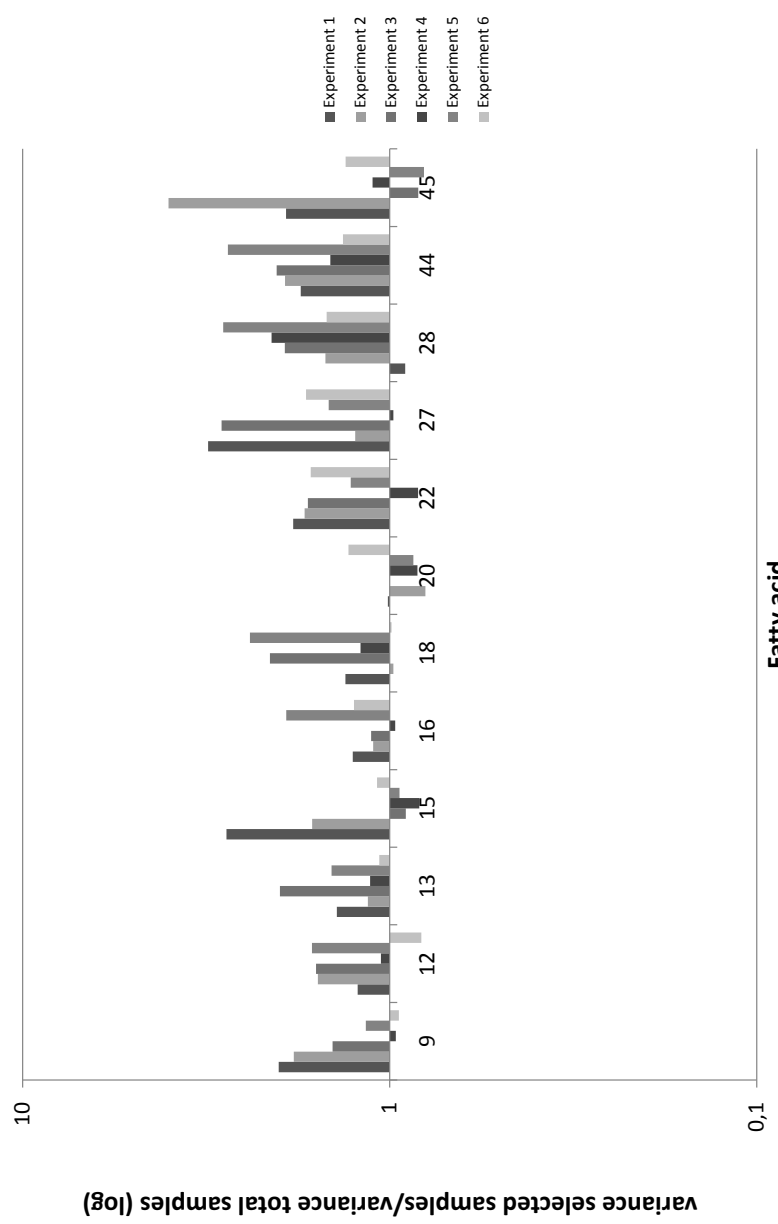


Figure 7.40: Variance of each fatty acid for the selected subset per experiment divided by the variance of the corresponding fatty acid for the total data set per experiment (adapted AS algorithm (AA) (individual experiments)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

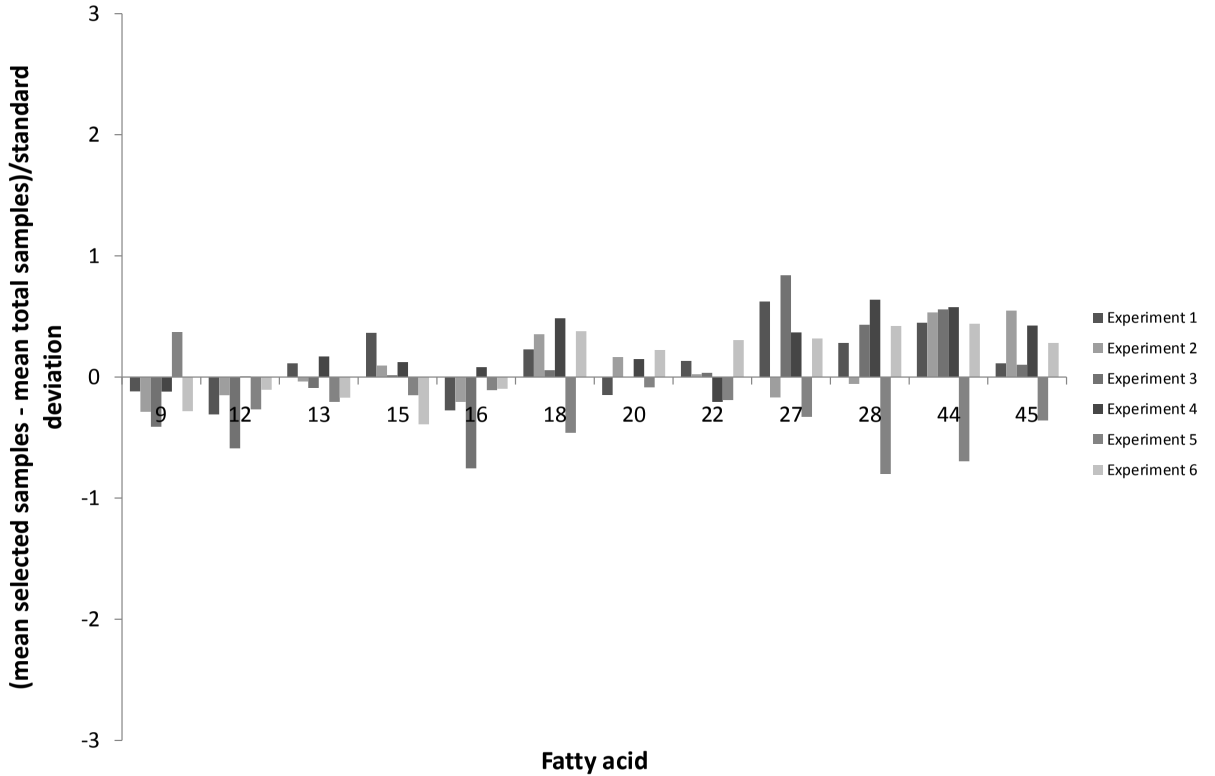


Figure 7.41: Mean of each fatty acid for the selected subset per experiment minus the mean of the corresponding fatty acid for the total data set per experiment, divided by the standard deviation per experiment (adapted AS algorithm (AA) (individual experiments)). The presented fatty acids are iso C14:0 (9), iso C15:0 (12), anteiso C15:0 (13), C15:0 (15), iso C16:0 (16), iso C17:0 (18), C16:1 cis-9 + anteiso C17:0 (20), C17:0 (22), C18:1 trans-10 (27), C18:1 trans-11 (28), C18:2 cis-9,trans-11 (44) and C18:2 trans-10,cis-12 (45).

7.2.4. Reanalysis with the priority fatty acids

In our case study the most important fatty acids were known. This enables us to repeat the described algorithms using only these priority fatty acids. This is essentially equivalent to a dimensionality reduction. This reanalysis confirmed the potential of Genetic Algorithms for this problem (Tables 7.1 & 7.2).

7.2.5. Statistical significance of the differences

Tables 7.1 and 7.2 indicate that the results of the different subset selection algorithms are clearly better when the multi-experiment aspect of the data is taken into account. Therefore, we will only discuss the statistical significance of the differences for this part of the results. Table 7.2 illustrates that there is a strong difference in quality between the standard subset selection algorithms (Kennard and Stone, OptiSim and k-means) and the subset selection algorithms presented here (Genetic Algorithms, Ant Colony Optimization). For that reason we argue that it is not necessary to make a statistical comparison between the standard subset selection algorithms and the subset selection algorithms presented here. However, there is only a small difference in quality between Genetic Algorithms with the adapted operators and the adapted AS algorithm and a statistical comparison would be appropriate. Since we have 50 repetitions, we can rely on the central limit theorem for normality. Therefore, we can apply a two sample T-test with a Satterthwaite correction for unequal variances to compare these two methods [78]. As the p-value is smaller than 0.05, this test indicates that Genetic Algorithms are significantly better than Ant Colony Optimization to select a subset of samples out of this data set.

7.3. Distribution of the optimal subset

In this section, the distribution of the total data set is compared to the distribution of the set of selected samples for some priority fatty acids. We restrict our attention to the case of Genetic Algorithms with the fitness function κ^*, because this algorithm produced the best results. Figures 7.42 and 7.43 illustrate that the distribution of the selected samples is flattened, and it is also important to note that the extreme regions are relatively more represented in the subset. This is a confirmation that this method is capable of selecting a subset of samples that is as informative as the total data set.

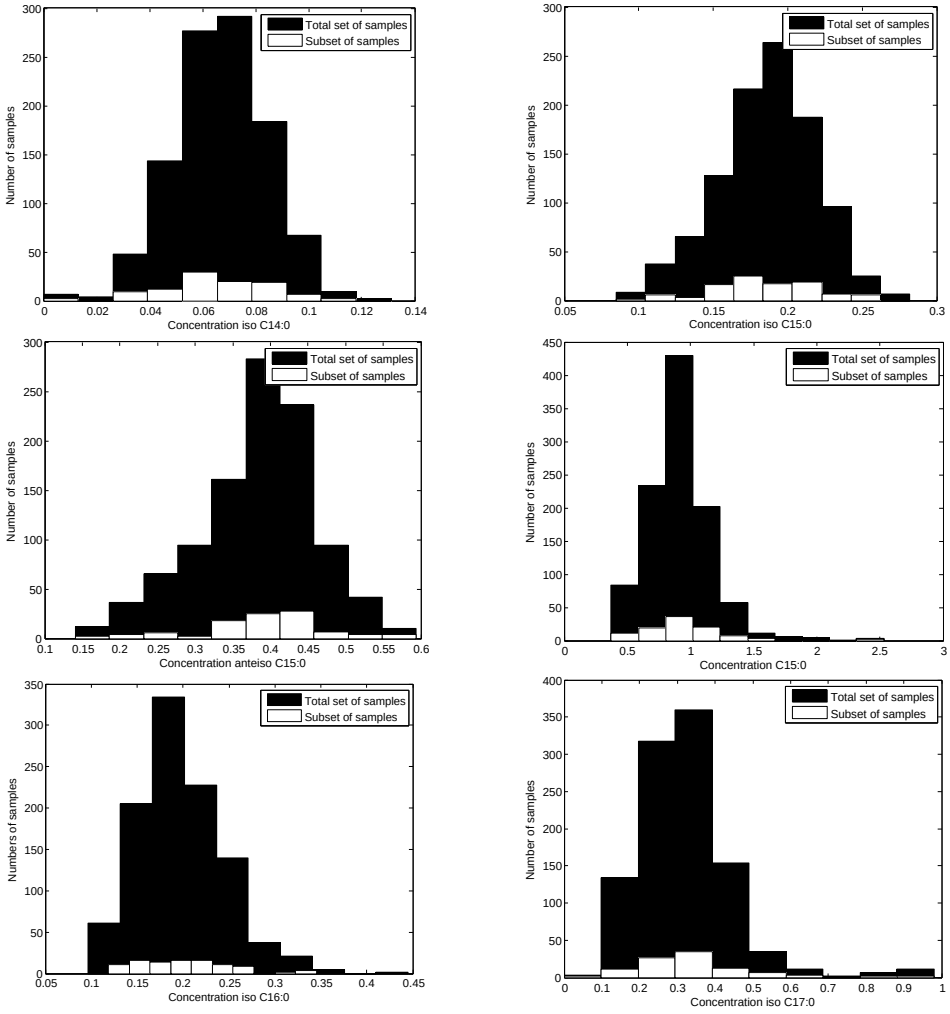


Figure 7.42: Distribution of the concentration of some fatty acids ($g/100\ g$) for the total data set and for the set of selected samples.

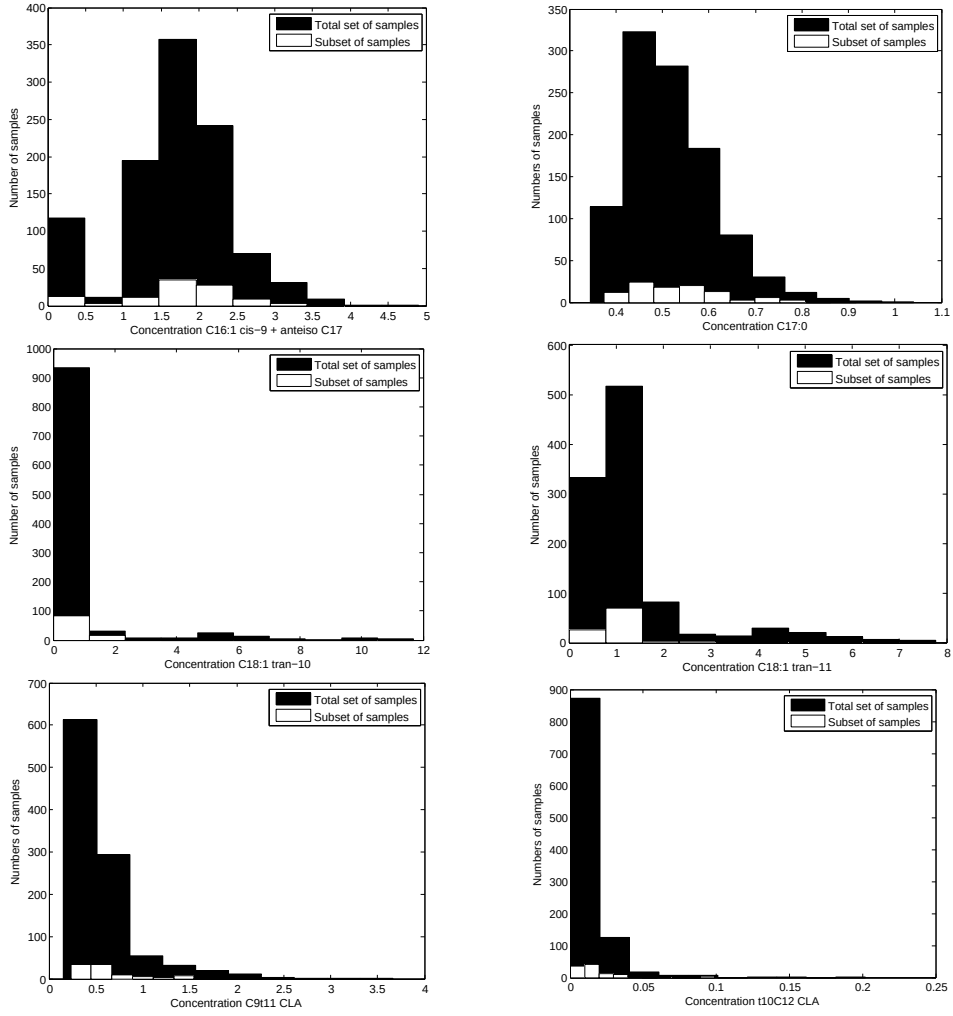


Figure 7.43: Distribution of the concentration of some fatty acids ($g/100\ g$) for the total data set and for the set of selected samples (continuation).

Table 7.2: Summary of the results of selecting a subset of samples out of 1033 samples for all fatty acids when the fitness function is based on separate calculations per experiment. For each method the number of samples selected from each experiment is presented. The results of the adapted and standard Kennard and Stone algorithm are presented with the Euclidean Distance (ED-adapted and ED). For Genetic Algorithms (GA) the results of both adapted operators (AO) and standard operators (SO) are listed.

	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6	κ^*
All fatty acids							
Total set of samples	144	426	117	63	26	257	1
ED-adapted-1	17	17	17	16	16	17	0.39
ED-adapted-2	14	41	11	6	3	25	0.01
ED-1	17	17	17	16	16	17	0.4
ED-2	14	41	11	6	3	25	0.023
k-means-1	17	17	17	16	16	17	0.089
k-means-2	14	41	11	6	3	25	0.023
OptiSim-1	17	17	17	16	16	17	0.1
OptiSim-2	14	41	11	6	3	25	0.089
GA (κ^*) (AO)	16	28	10	13	11	22	0.87
GA (κ^*) (SO)	44	115	19	19	11	67	0.9
AS (κ^*) (OA)	76	24					0
AS (κ^*) (AA)	18	24	17	13	6	22	0.8
Priority fatty acids							
Total set of samples	144	426	117	63	26	257	1
ED-adapted-1	17	17	17	16	16	17	0.32
ED-adapted-2	14	41	11	6	3	25	0.092
ED-1	17	17	17	16	16	17	0.49
ED-2	14	41	11	6	3	25	0.19
k-means-1	17	17	17	16	16	17	0.32
k-means-2	14	41	11	6	3	25	0.047
OptiSim-1	17	17	17	16	16	17	0.72
OptiSim-2	14	41	11	6	3	25	0.2
GA (κ^*) (AO)	18	28	8	9	8	29	1
GA (κ^*) (SO)	33	91	24	16	9	54	1
AS (κ^*) (AA)	17	24	14	11	10	24	1

8

Conclusion

This part of this dissertation deals with subset selection problems, more specifically with the problem of selecting a subset of samples out of a data set consisting of a large number of samples that originate from different experimental settings. The objective is to select a subset of samples in which each class of the experimental settings is sufficiently represented. After a literature study, we decided to apply some well-known and often used methods, namely the Kennard and Stone algorithm, the Optimizable k-Dissimilarity Selection method (OptiSim) and an algorithm based on clustering methods such as k-means clustering. The subset selection problem can also be transformed to an optimization problem and we thus can also make use of combinatorial optimization algorithms. Therefore, algorithms such as Genetic Algorithm and Ant Colony Systems can also be applied to the subset selection problem. Chapter 6 describes these different subset selection algorithms. The performance of these algorithms was then evaluated through a case study consisting of a data set that contains the concentration of 45 fatty acids in 1033 milk samples. These milk samples belong to six different experiments. The objective of this case study was to select a subset of 100 samples representing all the variability present in the total data set. As a consequence, the different experiments have to be sufficiently represented.

The results presented in Chapter 7 lead to the conclusion that Genetic Algorithms and Ant Colony Optimization are very good candidates to obtain a representative subset of samples from a multi-experiment data set. We introduced some modifications and proposed a possible objective function to accomplish this. We believe that the maximization of the fitness function κ^*, defined in Eq. (6.3) (Chapter 6), when calculated for all fatty acids and for each experiment separately, constitutes a good criterion to select a representative subset for a multi-experiment data set. Next to less good results, the conventional methods have a major disadvantage, namely the requirement to fix the number of selected samples per experiment in advance. As mentioned in Section 7.2.3 (Chapter 7), there is no straightforward way to do this. As the simulations have shown, the results are strongly dependent on the number of selected samples per experiment.

Important to note is that the results obtained with the adapted AS algorithm are notably better than the results obtained with the original AS algorithm. The adapted AS algorithm with the maximization of the fitness function κ^* approximates

the results of Genetic Algorithms. This confirms that, for this subset selection problem, the adaptation of the probabilistic rule and the pheromone update rule strongly improves the Ant Colony Optimization algorithm. However, the Ant Colony Optimization algorithm is more time-consuming, even after parallelization of the algorithm, than Genetic Algorithms. In addition, the results obtained with Genetic Algorithms are still significantly better than the results obtained with the Ant Colony Optimization algorithm.

Part III

Calibration of a water and energy balance model

9

Introduction

This part of the dissertation deals with the calibration of hydrologic models. The estimation of the values of various model parameters is a major problem in the application of hydrologic models. Ideally, estimates of these values should be obtained through direct in-situ observations. However, this is usually not possible because of (1) a difference in the spatial scale between the measurement of model parameters and the application of the model, (2) the non-physical meaning of a number of parameters which hence cannot be measured, (3) an inconsistency between the model physics and in-situ observed parameter values due to simplifications of reality by the model and (4) the huge number of model parameters to be estimated in for instance spatially distributed models. Although advances have been made in the use of remote sensing to determine model parameters, such as soil hydraulic parameters [79, 80, 81, 82, 83] and Leaf Area Indices [84, 85, 86], a large number of model parameters are still very difficult to measure in-situ and even impossible to observe in a spatially distributed or catchment-averaged context. In order to overcome this problem, model parameters are usually estimated by tuning them to the value for which the outputs of the model correspond best to observations [87, 88, 89, 90]. A number of problems arise when the parameters for physically-based hydrologic models need to be estimated. These models generally use a full set of meteorologic forcing data, combined with numerous topographic, land cover, and soil parameters, and may result in a large number of output variables. However, these models are usually calibrated using only one or a limited number of variables. Nevertheless, different model outputs are sensitive to different parameter values. The use of observations of one variable for model calibration can hence lead to parameter values that result in a good model performance for some model outputs, but not for all. The use of observations of multiple variables in the estimation of model parameters can be a solution to this problem. Examples of such multi-variable calibration studies are the use of combinations of soil moisture contents, soil temperatures, and sensible and latent heat flux observations [91], the use of observations of latent heat fluxes, soil heat fluxes, and soil moisture values [92], the use of remotely sensed surface skin temperatures and catchment discharge [93], and the use of observations of groundwater levels and runoff rates [94].

A problem typically encountered in the use of observations of different variables for model calibration is that these observations can be of different orders of magnitude.

Without transformation of the data, it is thus difficult to define a single objective function to be minimized, since one or a number of variables will dominate this objective function. For this reason, the above-mentioned studies calculated separate objective functions for each variable. Gupta *et. al.* [91], Houser *et. al.* [92], and Madsen [94] then searched for the parameter values that determine the location of the Pareto front. A different approach was adopted by Crow and Wood [93], in which two different objective functions were normalized by the standard deviations of the observations of the multiple variables. The resulting objective function was then minimized.

The above-mentioned studies led to the conclusion that, for the calibration of physically-based hydrologic models, as many non-redundant variables as possible should be used. Using the approaches of Gupta *et. al.* [91], Houser *et. al.* [92], and Madsen [94], this will lead to a large number of objective functions and, consequently, a high-dimensional Pareto front. One solution to this problem is the use of the Multistart Weight-Adaptive Recursive Parameter Estimation (MWARPE) method [95], in which all variables are taken into account explicitly in the parameter updating. This method iteratively uses the extended Kalman filter equations in a Monte-Carlo framework. No RMSE values of the output variables are optimized throughout the parameter estimation procedure and no weighing of objective functions or rescaling of variables needs to be performed. Another solution is to rescale, per variable, all observations (and the corresponding simulations) by substracting the mean and dividing by the standard deviation, calculated over the calibration period. This is similar to the approach in [93]. If, over all variables, the objective functions are then added, a single objective function is obtained, which can then be minimized. It is important to notice that this normalization of the data is only required to ensure commensurability in a single-objective framework. For this last calibration approach, we restrict ourselves to the Particle Swarm Optimization (PSO) algorithm [28]. We choose this algorithm because we want to investigate the capabilities of PSO to solve complex optimization problems. The application of this optimization algorithm is a recurring theme in further parts of this dissertation. The objective of this part of this dissertation is to thoroughly compare both approaches, using observations of the energy balance and the soil moisture profile. PSO and MWARPE have already been applied to the estimation of parameters of hydrologic models [96, 97], but using only discharge data and not in a multi-variate context.

Chapter 10 discusses MWARPE and PSO in full detail. In Chapter 11, MWARPE and PSO are applied to the calibration of a relatively simple process-based water and energy balance model, applied at the point scale. Chapter 12 presents the conclusions of this part of this dissertation.

10

Methodology

In this chapter, the algorithms used to calibrate a hydrologic model, discussed in the case study, are described. This chapter is set out as follows. Section 10.1 introduces some general definitions regarding the input and the output of the models we would like to calibrate. Section 10.2 describes the MWARPE algorithm in the application of calibration of hydrologic models. While Particle Swarm Optimization is described in full detail in Section 3.2 of Chapter 3, Section 10.3 of this chapter discusses the application of Particle Swarm Optimization to the calibration of a hydrologic model. In Section 10.4, the advantages and disadvantages of both methods are outlined.

10.1. Definition of the model calibration problem

We assume to be given a model that depends on r real-valued inputs whose values are denoted using a vector $\boldsymbol{o} \in \mathbb{R}^r$. The output of the model consists of the value of variables y_j with $j = 1, \ldots, J$ that we can compare to available data $y_{j,i}$ at different observation times $i = 1, \ldots, m_j$. It is also useful to collect all output observations in one large vector $\boldsymbol{y} \in \mathbb{R}^m$, where

$$m = \sum_{j=1}^{J} m_j. \tag{10.1}$$

In order to generate the outputs, the model also depends on some real-valued parameters whose values are denoted using a vector $\boldsymbol{x} \in \mathbb{R}^n$ and which are to be estimated. The general relation between the input $\boldsymbol{o}$, the parameters $\boldsymbol{x}$ and the model output $\boldsymbol{y}$ is thus described by the model and denoted as

$$\boldsymbol{y} = c(\boldsymbol{o}, \boldsymbol{x}). \tag{10.2}$$

Since typically $m \gg n$, it is impossible to invert the relation between $\boldsymbol{y}$ and $\boldsymbol{x}$ for given $\boldsymbol{o}$ and to find an exact solution $\boldsymbol{x}$ for a given set of observed output values $\boldsymbol{y}$. We thus need other methods to calibrate the model, *i.e.* to find an optimal set of parameters $\boldsymbol{x}^*$ such that the estimated output $\hat{\boldsymbol{y}} = c(\boldsymbol{o}, \boldsymbol{x}^*)$ closely resembles the observed output $\boldsymbol{y}$.

10.2. Multistart Weight-Adaptive Recursive Parameter Estimation

Multistart Weight-Adaptive Recursive Parameter Estimation (MWARPE) is a method for parameter estimation, and thus model calibration, that is based on the equations of the Extended Kalman filter for data assimilation. These equations are used recursively in a Monte-Carlo framework, based on which the algorithm can be referred to as Multistart Weight-Adaptive Recursive Parameter Estimation (MWARPE) [95].

We first summarize the equations of the linear and extended Kalman filter, based on the introduction in [98]. The discrete Kalman filter was developed by Kalman [99] to estimate the state of a system that evolves in discrete time based on the previous estimate and a current measurement of some properties of the system that depend on its state. For the linear Kalman filter, all dependencies are assumed to be linear. The state of the system at time k can be described using n real numbers and is denoted as $\boldsymbol{x}(k) \in \mathbb{R}^n$. It evolves according to the linear stochastic difference equation

$$\boldsymbol{x}(k) = A\boldsymbol{x}(k-1) + B\boldsymbol{u}(k-1) + \boldsymbol{w}(k-1)\,. \tag{10.3}$$

The vector $\boldsymbol{u}(k) \in \mathbb{R}^l$ describes an optional input that drives or controls the system. The stochastic component of the evolution is given by $\boldsymbol{w}(k-1) \in \mathbb{R}^n$ and is assumed to be a Gaussian white noise term with zero mean and a covariance $Q(k)$ that possibly depends on the time k. The $n \times n$ matrix A and the $n \times l$ matrix B describe the deterministic dependency of the state $\boldsymbol{x}(k)$ on the previous state $\boldsymbol{x}(k-1)$ and the input $\boldsymbol{u}(k-1)$ respectively. In addition, m different real-valued properties of the system are measured at every time k, resulting in a vector output $\boldsymbol{y}(k) \in \mathbb{R}^m$ that linearly depends on the state of the system according to

$$\boldsymbol{y}(k) = H\boldsymbol{x}(k) + \boldsymbol{v}(k)\,. \tag{10.4}$$

The $m \times n$ matrix H describes the dependency and the vector $\boldsymbol{v}(k)$ the measurement noise, which is also assumed to be white and distributed according to Gaussian distribution with zero mean and a possibly time-dependent covariance $R(k)$. The Kalman filter provides a method to use measurement data of $\boldsymbol{y}(k)$ to improve the estimation of the state of the system. Let $\hat{\boldsymbol{x}}(k-1)$ denote the last estimate of the state of the system. We also define a corresponding estimate error covariance $P(k-1)$, in such a way that

$$E[\boldsymbol{x}(k-1)] = \hat{\boldsymbol{x}}(k-1)\,, \tag{10.5}$$

$$E[(\boldsymbol{x}(k-1) - \hat{\boldsymbol{x}}(k-1))(\boldsymbol{x}(k-1) - \hat{\boldsymbol{x}}(k-1))^{\mathrm{T}}] = P(k-1)\,. \tag{10.6}$$

The next estimate $\hat{\boldsymbol{x}}(k)$ is obtained in a two-step cycle, using a time update or

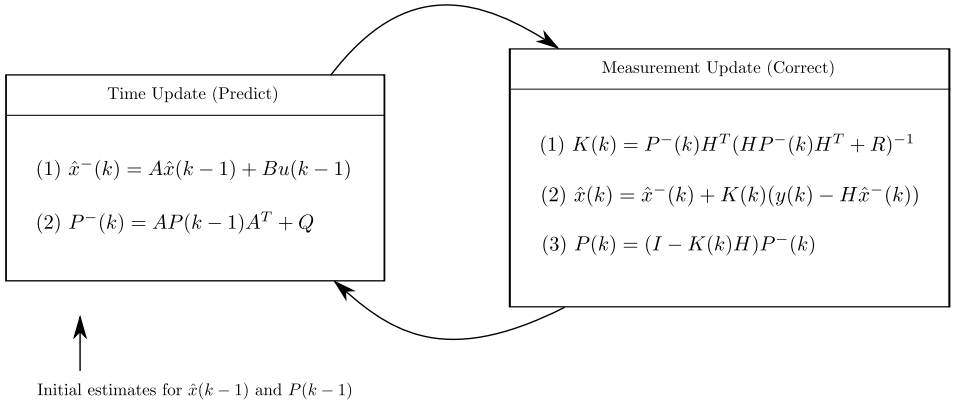


Figure 10.1: Equations of the linear Kalman filter

prediction step and a measurement update or correction step. The corresponding equations are summarized in Figure 10.1. Without the measurement data, an *a priori* estimate $\hat{\boldsymbol{x}}^-(k)$ for the state of the system is predicted using the linear difference equation (10.3). In the correction phase, we then compute the mismatch

$$\boldsymbol{y}(k) - H\hat{\boldsymbol{x}}^-(k) \tag{10.7}$$

between the observed output $\boldsymbol{y}(k)$ and the estimated output based on the *a priori* estimate $\hat{\boldsymbol{x}}^-(k)$, and use this to compute a corrected parameter estimate $\hat{\boldsymbol{x}}(k)$. Of central importance is the computation of an $n \times m$ matrix that is called the Kalman gain and is computed such that the new estimate $\hat{\boldsymbol{x}}(k)$ minimizes the trace of the corresponding estimate error covariance $P(k)$.

Since many processes in nature are non-linear [98], the state of a system will more generally evolve according to a nonlinear stochastic difference equation given by

$$\boldsymbol{x}(k) = \boldsymbol{f}(\boldsymbol{x}(k-1), \boldsymbol{u}(k-1), \boldsymbol{w}(k-1))\,, \tag{10.8}$$

and the corresponding measurement will be related to the state by

$$\boldsymbol{y}(k) = \boldsymbol{h}(\boldsymbol{x}(k), \boldsymbol{v}(k))\,. \tag{10.9}$$

In that case, one should use the extended Kalman filter [98]. The *a priori* update of the state estimate is given by

$$\hat{\boldsymbol{x}}^-(k) = \boldsymbol{f}(\hat{\boldsymbol{x}}(k-1), \boldsymbol{u}(k-1), 0). \tag{10.10}$$

To compute the *a priori* estimate error covariance, the extended Kalman filter uses a first order Taylor expansion to linearize Eq. (10.8) to

$$\boldsymbol{x}(k) = \hat{\boldsymbol{x}}^-(k) + A(k)\big(\boldsymbol{x}(k-1) - \hat{\boldsymbol{x}}(k-1)\big) + W(k)\boldsymbol{w}(k-1)\,, \tag{10.11}$$

where the entries of the $n \times n$ matrix $A(k)$ and the $n \times n$ matrix $W(k)$ are given by

$$A_{i,j}(k) = \frac{\partial f_i}{\partial x_j}(\hat{\boldsymbol{x}}(k-1), \boldsymbol{u}(k-1), \mathbf{0}), \tag{10.12}$$

$$W_{i,j}(k) = \frac{\partial f_i}{\partial w_j}(\hat{\boldsymbol{x}}(k-1), \boldsymbol{u}(k-1), \mathbf{0}). \tag{10.13}$$

This results in the update equation

$$P^-(k) = A(k)P(k-1)A(k)^{\mathrm{T}} + W(k)Q(k-1)W(k)^{\mathrm{T}}. \tag{10.14}$$

To correct this *a priori* prediction, the output relation Eq. (10.9) is linearized at $\hat{\boldsymbol{x}}^-(k)$, resulting in

$$\boldsymbol{y}(k) = \boldsymbol{h}(\hat{\boldsymbol{x}}^-(k), 0) + H\big(\boldsymbol{x}(k) - \hat{\boldsymbol{x}}^-(k)\big) + V\boldsymbol{v}(k). \tag{10.15}$$

where the entries of the $m \times n$ matrix H and the $m \times m$ matrix V are given by

$$H_{i,j}(k) = \frac{\partial h_i}{\partial x_j}(\hat{\boldsymbol{x}}^-(k), \mathbf{0}), \tag{10.16}$$

$$V_{i,j}(k) = \frac{\partial h_i}{\partial v_j}(\hat{\boldsymbol{x}}^-(k), \mathbf{0}). \tag{10.17}$$

Having a measurement $\boldsymbol{y}(k)$, we can update the state estimate as

$$\hat{\boldsymbol{x}}(k) = \hat{\boldsymbol{x}}^-(k) + K(k)\big(\boldsymbol{y}(k) - \boldsymbol{h}(\hat{\boldsymbol{x}}^-(k), 0)\big) \tag{10.18}$$

where the Kalman gain $K(k)$ is now given by

$$K(k) = P^-(k)H(k)^{\mathrm{T}}(H(k)P^-(k)H(k)^{\mathrm{T}} + V(k)R(k)V(k)^{\mathrm{T}})^{-1} \tag{10.19}$$

and the estimate error covariance $P(k)$ is updated as before.

In the calibration algorithm MWARPE [95], the set of model parameters is interpreted as the state of the system and thus denoted by $\boldsymbol{x}$. For each iteration level k, the entire calibration period is considered. The parameter vector is thus expected not to change, except due to stochastic effects, so that the evolution equation is given by

$$\boldsymbol{x}(k) = \boldsymbol{x}(k-1) + \boldsymbol{w}(k-1). \tag{10.20}$$

We thus have to use the $n \times n$ unit matrix for A and W in the equations of the extended Kalman filter. The parameters $\boldsymbol{x}$ are 'observed' through the model output $\boldsymbol{y}$ and compared to the measured data over the whole calibration period. The

measurement of the system state is thus described by

$$\boldsymbol{y}(k) = c(\boldsymbol{x}(k)) + \boldsymbol{v}(k). \tag{10.21}$$

where c is the model that needs to be calibrated. This results in the $m \times m$ unit matrix V and

$$H_{i,j}(k) = \frac{\partial c_i}{\partial x_j}(x(k)) \tag{10.22}$$

for the entries $H_{i,j}$ of the $m \times n$ matrix H. The extended Kalman filter is then recursively applied, always using the same set of calibration data for the observations $\boldsymbol{y}(k)$, until convergence in the parameter values $\boldsymbol{x}(k)$ is achieved or until a predefined number of iterations has been reached. Following [95], this algorithm is applied to a predefined number of starting points that are distributed uniformly in the parameter space. For the different solutions resulting from these starting points, the solution that produces the closest fit between the observations and the model output is assumed to contain the best parameter values. The closeness of the fit is evaluated using the Root Mean Square Error, which will be defined in the next section [95].

10.3. Particle Swarm Optimization

As mentioned in Chapter 9, physically-based hydrologic models may result in a large number of output variables. For the calibration of such models, all these multiple outputs should be taken into account, as is done by the MWARPE method. Alternatively, we can try to construct a single objective function that imposes that all model outputs try to approximate the observed values. This objective function has to be optimized and the calibration problem is cast to an optimization problem. As objective function, we have chosen to work with the overall 'Root Mean Square Error' (RMSE). The RMSE for a single output variable j can be calculated by using the observed data $y_{j,i}$ at the different observation times $i = 1, \ldots, m_j$ in:

$$\text{RMSE}_j = \sqrt{\frac{\sum_{i=1}^{m_j} (y_{j,i} - \hat{y}_{j,i})^2}{m_j}} \tag{10.23}$$

with $\hat{y}_{j,i}$ the simulated data of output variable j at observation time i. The overall RMSE is then defined as the sum of the different standardized RMSE values of the different output variables. Hence, for every output variable, both the observed data and the model output are standardized by subtracting the mean of the observed data and dividing by the standard deviation of the observed data. In this way, equal weights are given to the different output components of the hydrologic model in the objective function.

Because the hydrologic model connecting the input variables to the output variables cannot expected to be linear, we may assume that local minima would be present for the overall RMSE. It is thus not possible to use a local optimization algorithm based on the concepts discussed in Chapter 2, Section 2.2. For that reason, we choose to work with Particle Swarm Optimization (PSO), which is a population-based global optimization algorithm. For a detailed description of the PSO algorithm, we refer to Chapter 3 (Section 3.2).

10.4. Advantages and drawbacks of both methods

A number of direct advantages and drawbacks can be identified for both algorithms. Firstly, for MWARPE, no RMSE between the observations and corresponding model simulations is optimized. Instead, Eq. (10.7) shows that the mismatch for every observation is explicitly taken into account in the parameter updating. This has as clear advantage that no weighing of the different RMSE values or rescaling of the observations has to be performed if multiple variables with different orders of magnitude are used in the model calibration. Further, as demonstrated in [95] for a simple rainfall-runoff model, this will lead to parameter combinations that will work well under both high and low flow conditions, as opposed to traditional RMSE minimization methods, which tend to work well under only high or low flow conditions, depending on the transformation of the observations. In other words, this problem is the result of the focus on a single objective. The major drawback of MWARPE is the high dimensionality $m = \sum_{j=1}^{J} m_j$ of the matrix $[H(k)P^{-}(k)H(k)^{\mathrm{T}} + R(k)]$ that needs to be inverted. If, for example, 5 variables are used, and hourly simulations are used for one month (30 days), a 3600×3600 matrix needs to be inverted. The major advantage of Particle Swarm Optimization is that the algorithm is easy to understand and easy to implement. Furthermore, the algorithm has a small tendency of getting trapped in local minima and the balance between the global and local exploration of the search space can be controlled [3, 100]. As a major drawback, a number of parameters inherent to the algorithm have to be determined, for which the values depend on the problem at hand.

11

Case Study

In this chapter, the two calibration algorithms described in Chapter 10 are applied on a case study. For the purpose of this study, a very simple hydrologic model was developed by professor V. Pauwels of the Department of Forest and Water Management of Ghent University (Section 11.1). The data set used for the calibration of the model is described in Section 11.2. Section 11.3 discusses the implementation of the used calibration algorithms. In Section 11.4, the results of the two calibration algorithms are presented and discussed.

11.1. Model description

The model applied in this case study gives a description of the water and energy balance in time for one point on the earth's surface. The model is thus one-dimensional in space, where the coordinate z (m) labels the depth beneath the surface. For the discretization of the differential equations used in this model, the vertical coordinate z is defined positive upwards and the used number of nodes is 21 with a separation distance of 5 cm. The inputs needed by the model are the air temperature T_a (K), dew point temperature (T_d) (K), air pressure P_a (kPa), wind speed $u(z)$ (ms^{-1}), incoming long wave radiation $L_{w,i}$ (Wm^{-2}), incoming solar radiation $R_{s,i}$ (Wm^{-2}) and precipitation. The outputs generated by this model are the soil moisture content at 5 cm depth θ_1 (-), soil moisture content at 9 cm depth θ_2 (-), soil moisture content at 15 cm depth θ_3 (-), soil moisture content at 25 cm depth θ_4 (-), the net radiation Rn (Wm^{-2}), the latent heat flux LE (Wm^{-2}), the sensible heat flux H (Wm^{-2}) and the ground heat flux G (Wm^{-2}).

The movement of soil water in the unsaturated zone is modelled using a numerical solution to the Richards equation [101]:

$$C_{\mathrm{m}}(\psi)\frac{\partial\psi}{\partial t} = \frac{\partial}{\partial z}\left(K(\psi)\frac{\partial\psi}{\partial z}\right) + \frac{\partial K(\psi)}{\partial z}\,, \tag{11.1}$$

with ψ the pressure head (m) (< 0 for unsaturated soils), C_{m} the specific moisture capacity (m^{-1}), t the time (s), z the vertical coordinate (m), and K the hydraulic conductivity (ms^{-1}). The relationship between K and ψ is modelled using the

Brooks-Corey equations [102]:

$$K(\psi) = \begin{cases} K_s(z) \left(\dfrac{\psi}{\psi_c}\right)^{-2-3\lambda} & \text{if } \psi < \psi_c\,, \\ K_s(z) & \text{if } \psi \geq \psi_c\,, \end{cases} \tag{11.2}$$

with K_s the saturated hydraulic conductivity (ms^{-1}), ψ_c the air entry pressure head (m), and λ the pore size distribution index (-). The relationship between the volumetric soil moisture content θ (-) and the pressure head is also modelled using the Brooks-Corey equations [102]:

$$\theta(\psi) = \begin{cases} \theta_r + (\theta_s - \theta_r) \left(\dfrac{\psi}{\psi_c}\right)^{-\lambda} & \text{if } \psi < \psi_c\,, \\ \theta_s & \text{if } \psi \geq \psi_c\,, \end{cases} \tag{11.3}$$

with θ_r and θ_s the saturated and residual soil moisture content (-) respectively. The specific moisture capacity can be written as [102]:

$$C_{\mathrm{m}}(\psi) = \frac{\partial\theta}{\partial\psi} = \begin{cases} -\dfrac{\lambda}{\psi_c}(\theta_s - \theta_r) \left(\dfrac{\psi}{\psi_c}\right)^{-\lambda-1} & \text{if } \psi < \psi_c\,, \\ 0 & \text{if } \psi \geq \psi_c\,. \end{cases} \tag{11.4}$$

The exponential decay of the hydraulic conductivity with the depth is written as [103]:

$$K_s(z) = K_{s0} e^{-fz}\,, \tag{11.5}$$

with K_{s0} the value of K_s at depth 0 (m^{-1}), z the depth below the surface (m), and f the TOPMODEL parameter (m^{-1}). Equation (11.1) is solved through a Crank-Nicholson finite difference discretization and a Picard iteration scheme. The boundary conditions are a Dirichlet condition (constant pressure head) at the bottom of the profile, and a Neumann condition (imposed flux), calculated as the difference between the precipitation and the evapotranspiration, at the top of the profile. The hydraulic conductivity between the nodes is calculated as the arithmetic mean of the hydraulic conductivity of the above and underlying node. The evapotranspiration is calculated through an iteration for the surface skin temperature as follows [104]:

$$R_{s,i}(1-\alpha) + L_{w,i} - \epsilon\sigma T_s^4 = \underbrace{\frac{C_p\rho_a}{\Psi}\frac{e_s(T_s) - e_a}{r_{av} + r_c}}_{\mathrm{LE}} + \underbrace{C_p\rho_a\frac{T_s - T_a}{r_{ah}}}_{\mathrm{H}} + \underbrace{\kappa\frac{T_s - T_1}{\triangle z}}_{\mathrm{G}}\,, \tag{11.6}$$

with α the surface albedo (-), ϵ the emissivity (-), σ the Stefan-Bolzmann constant ($\mathrm{Wm}^{-2}\mathrm{K}^{-4}$), T_s the surface skin temperature (K), C_p the specific heat of moist air ($\mathrm{Jkg}^{-1}\mathrm{K}^{-1}$), ρ_a the density of air (kgm^{-3}), Ψ the psychrometric constant (kgPa^{-1}), e_s the saturated vapor pressure (kPa), e_a the actual vapor pressure

(kPa), r_{av} the aerodynamic resistance to vapor transport (sm^{-1}), r_c the surface resistance (sm^{-1}), r_{ah} the aerodynamic resistance to heat transport (sm^{-1}), κ the soil thermal conductivity ($Wm^{-1}K^{-1}$), T_1 the soil temperature below the first soil layer (K), and $\triangle z$ the depth of the first soil layer (m). The left-hand side of Eq. (11.6) indicates the net radiation, the first term of the right-hand side the latent heat flux (LE), the second term the sensible heat flux (H), and the third term the ground heat flux (G). The aerodynamic resistances for heat or vapor transport are calculated as follows [104]:

$$r_{av} = \frac{1}{u(z)K_v^2}\ln^2\left(\frac{z-d}{z_{0v}}\right), r_{ah} = \frac{1}{u(z)K_v^2}\ln^2\left(\frac{z-d}{z_{0h}}\right), \tag{11.7}$$

with $u(z)$ the wind speed (ms^{-1}), K_v the von Karman constant ($\simeq 0.4$), d the zero plane displacement height (m), z_{0v} the roughness length for vapor transport and z_{0h} the roughness length for heat transport (m). The roughness length for vapor transport z_{0v} is equal to $f_v h$, with h the vegetation height (m). For heat transport $z_{0h} = f_h h$. The zero plane displacement height is equal to $f_d h$. The roughness length for vapor transfer fraction f_v, the roughness length for heat transfer fraction f_h and the zero plane displacement height fraction f_d are three of the eleven parameters that must be calibrated. Since the height of the canopy changed during the growing season, the following expression for h was used:

$$h = \frac{h_{max}}{1 + e^{-p_h(i-\bar{i})}}, \tag{11.8}$$

with h_{max} known as 1 m, p_h a parameter, for which observations of the canopy (in this study winter wheat) height indicate that it can be assumed to be equal to 0.003 h^{-1} [105]. $\bar{i}$ (h) is half the number of time steps during the simulation, and i (h) the time step since the onset of the simulation.

The relationship between the saturated vapor pressure and the surface skin temperature is [104]:

$$e_s(T_s) = 0.6108e^{\frac{17.27 \cdot T_s}{237.3 + T_s}}. \tag{11.9}$$

In this equation T_s is entered in degrees Celsius. The psychrometric constant can be calculated as [104]:

$$\Psi = \frac{P_a C_p}{0.622 l_v}, \tag{11.10}$$

with P_a the air pressure (kPa), and l_v the latent heat of vaporization (Jkg^{-1}), which can be calculated as [104]:

$$l_v = 2501000 - 2361T_a. \tag{11.11}$$

In this equation, T_a is entered in degrees Celcius. The density of air (kgm^{-3}) can

be calculated as [104]:

$$\rho_a = 3.486 \frac{P_a}{275 + T_a} \,. \tag{11.12}$$

In this equation, T_a is entered in degrees Celcius as well. Equation (11.6) is solved through an iteration for the surface skin temperature T_s. A Newton-Raphson iterative scheme is used for this purpose. T_1 can be calculated through a numerical solution to the heat conduction equation [106]:

$$\frac{\partial CT}{\partial t} = \frac{\partial}{\partial z} \left[\kappa \frac{\partial T}{\partial z} \right] \,, \tag{11.13}$$

with C the volumetric heat capacity of the soil ($\mathrm{Jm^{-3}K^{-1}}$), T the soil temperature (K), and z the vertical coordinate (m). C and κ are assumed to be constant and homogeneous throughout the soil profile. Equation (11.13) is solved through a Crank-Nicholson finite difference discretization. The boundary conditions for this equation are a constant temperature at both the top and bottom of the profile. At the top of the profile, this temperature is equal to T_s, and at the bottom it is equal to a predefined temperature, *i.e.* a linearly increasing temperature of 10°C at the beginning of the study period to 15°C at the end of the study period. These values are based on measurements of the temperature during the AgriSAR 2006 campaign. The details of this campaign will be discussed in Section 11.2.

As a summary, eleven parameters need to be estimated: λ, ψ_c, K_s, f, α, κ, C, r_c, f_d, f_h, and f_v. We acknowledge the fact that the model represents a very strong simplification of the physical reality. A state-of-the-art land surface model could have been used as well. However, this would have implied the application of a sensitivity analysis, in order to select the calibrated model parameters. This would have led to a similar amount of calibrated parameters as with this simple model. Further, the focus of this case study is on the potential of two different calibration methods to estimate parameter values for a model that generates the required model output, not on the representation of all physical processes involved. For this reason, the model is deemed sufficiently realistic.

11.2. Site and data description

The data used in this study have been acquired in the framework of the AgriSAR 2006 campaign (AGRIcultural bio/geophysical retrieval from frequent repeat pass SAR and optical imaging), for which the test site was located in Mecklenburg-Vorpommern in North-East Germany, approximately 150 km North of Berlin. More specifically, Time Domain Reflectometry (TDR)-based soil moisture observations and Bowen Ratio Energy Balance (BREB)-based observations of the energy balance components in a large winter wheat field were available from April 20 through July 5, 2006. The soil moisture was measured at a depth of 5, 9, 15, and 25 cm.

Figure 11.1: Station set-up in the winter-wheat field. The soil moisture observations are taken at the same location as the (BREB)-based observations [105].

Meteorologic data from the weather station at Görmin were used as model forcing. All observations were converted to an hourly time step. Figure 11.1 shows a map with the location of the Görmin weather station (point 1 on the map) and the station for the (TDR)-based soil moisture and the (BREB)-based observations (point 2 on the map). Points 3 and 4 on the map correspond to the location of other measurement equipment. The observed data of this equipment was not used in this case study. A detailed description of this data set is given in [105].

This data was processed by professor V. Pauwels of the Department of Forest and Water Management of Ghent University. In order to remove outliers in the latent and sensible heat fluxes, observations for which the Bowen ratio was between -0.7 and -1.3 were removed from the data set [107]. Section 11.1 shows that the model is forced to close the energy balance. It is thus advisable that the data used to estimate the model parameters close the energy balance as well, a requirement that is met by the Bowen ratio method. Since the model calculates the net radiation and the ground heat flux, these variables should be used in the parameter estimation. The Bowen ratio method calculates the latent heat flux as a rest-term, while Section 11.1 shows that in the model a different approach is used. For this reason, both the latent and sensible heat fluxes were used in the model parameter estimation.

This data set is used to determine the model parameters. The data set is divided into two periods henceforth referred to as the first and second period, such that, when taking into account the missing data points, both periods contain approximately 50 % of the available data. The first period contains data measured from April 20 - June 21, whereas the second period contains data measured from June 22 - July 5. Both periods are used in the search for optimal hydrologic parameters; either the first period is used as training data set whereas the second period is then used as validation data set, or vice versa. In short, the first validation period corresponds to the second calibration period and the second validation period corresponds to

the first calibration period.

As already described (see Section 10.2), a major drawback of the MWARPE method is the high dimensionality, equal to the number of observations, of the matrix to be inverted. Therefore, using all hourly observations would lead to an excessively large matrix to be inverted, and hence the data set used for calibration was reduced as follows. Since the terms of the energy balance show a very strong diurnal cycle, only the observations at 1 PM and 1 AM were taken into account. Further, since the soil moisture data are relatively constant on a daily basis, only soil moisture observations at midday were used. This data set for the remainder of this chapter is referred to as the reduced data set, whereas the data set consisting of all hourly observations is further referred to as the hourly data set.

In contrast to the MWARPE method, PSO can easily handle all hourly observations since it does not rely on matrix inversion. However, in order to have a fair comparison between MWARPE and PSO, the training of PSO was firstly restricted to the reduced data set (see Section 11.3.1). To evaluate the impact of this restriction, the PSO-algorithm was also trained using the hourly data set (see Section 11.4.2).

11.3. Implementation of the calibration methods

In this section, the determination of the parameters inherent to the calibration algorithms is discussed.

11.3.1. Multistart Weight-Adaptive Recursive Parameter Estimation

The diagonal values in the parameter model error covariance matrix $Q(k)$ and the initial parameter error covariance matrix $P(0)$ were set equal to the square of a fraction of the corresponding parameter value in the parameter vectors $\mathbf{x}^-(k)$ and $\mathbf{x}(0)$, respectively [95]. $\mathbf{x}(0)$ is the vector with the initial guesses of the parameter values. Trial and error revealed that when this fraction was equal to 0.05, convergence in the parameter values was obtained relatively quickly (10-20 iterations), while oscillations in the parameter values occurred rarely. The noise in the soil moisture observations was assumed to be equal to one percent. For the net radiation and the latent and sensible heat fluxes the observation noise was assumed to be equal to 25 Wm^{-2}, and for the ground heat flux this was assumed to be equal to 10 Wm^{-2}. These values were also obtained by trial and error. Table 11.1 shows the acceptable range for the different model parameters, also referred to as the parameter space. When a starting point is located outside this parameter space, the parameters have been given the values of the nearest boundary. The

Table 11.1: Parameter space of the different hydrologic model parameters.

parameter	unit	Description	minimum	maximum
λ	-	Pore size distribution index	0.1	5
ψ_c	m	Bubbling pressure	-5	-0.1
K_s	ms^{-1}	Saturated hydraulic conductivity	2.78E-08	5.56E-05
f	m^{-1}	TOPMODEL exponential decay parameter	0.01	20
α	-	Albedo	0.01	0.9
κ	$Wm^{-1}K^{-1}$	Soil thermal conductivity	0.01	5
C	$Jm^{-3}K^{-1}$	Soil heat capacity	10E+04	15E+05
r_c	sm^{-1}	Surface resistance	0.01	250
f_d	-	Zero plane displacement height fraction	0.4	0.9
f_h	-	Roughness length for heat transfer fraction	0.01	0.5
f_v	-	Roughness length for vapor transfer fraction	0.01	0.5

algorithm is stopped when the parameter values do not change any more (difference less than 1 %). During the iteration process, the differences in the magnitude of the entries in the matrix to be inverted become very large. This can lead to numerical instabilities in the matrix inversion and consequently to a strong loss in numerical precision. When the error in the matrix inversion became larger than 0.0001, the algorithm is aborted. To compare the two calibration methods, the number of starting points of MWARPE is set equal to the population size of the Particle Swarm Optimization algorithm. However, in contrast to PSO, the different starting points are treated independently by the MWARPE method, meaning that the evolution of each particle is not influenced by the other particles.

11.3.2. Particle Swarm Optimization

As mentioned in Section 10.3, as objective function we choose the overall 'Root Mean Square Error' (RMSE), which is defined as the sum of the different standardized RMSE values of the different soil moisture observations and energy balance measurements. Table 11.2 displays the values used to standardize the data, and this for the different calibration and validation periods. The values used to standardize the data for the first (second) calibration period are the same as the values used for the first (second) validation period.

As mentioned in Section 11.3.1, the model parameters have to be positioned in a particular parameter space (see Table 11.1). When a population member is trying to move outside the parameter space during the PSO algorithm, the boundaries act as perfect reflectors. Phrased differently, the direction of displacement of that particle is inverted in order to keep it inside the parameter space.

The results of PSO also depend on the choice of several parameters: the population size N, the cognitive parameter c_1, the social parameter c_2 and the inertia weight w. According to Engelbrecht [3], a good value for the population size is given by $N = 30$. In order to determine good values for the parameters c_1, c_2 and w, an exhaustive search was performed for which the number of iterations is set to

Table 11.2: Values used to standardize the data

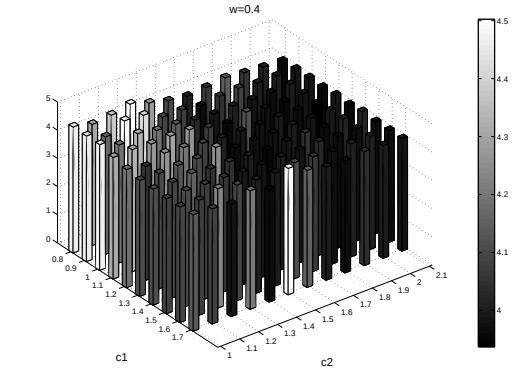
		First period		Second period	
	units	mean	standard deviation	mean	standard deviation
θ_1	-	0.170	0.055	0.120	0.045
θ_2	-	0.183	0.53	0.135	0.035
θ_3	-	0.181	0.049	0.124	0.027
θ_4	-	0.185	0.050	0.106	0.028
G	Wm^{-2}	3.799	18.063	4.904	14.601
H	Wm^{-2}	31.934	63.457	43.775	87.580
LE	Wm^{-2}	30.239	112.638	74.107	136.790
Rn	Wm^{-2}	65.972	155.618	122.786	204.969

40. A suitable approach to determine the maximal number of iterations is further described. The search procedure is carried out on the entire hourly data set.

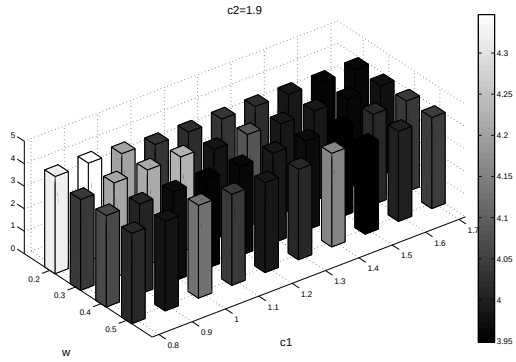
The parameter c_1 is varied between 0.8 and 1.7, c_2 between 1 and 2.1, and w between 0.2 and 0.5, with steps of 0.1. The parameter values are chosen in the convergence domain of Particle Swarm Optimization, *i.e.* the region for which the population will converge [108].

Figure 11.2 shows the dependence of the mean RMSE values, obtained after 40 iterations, on the different combinations of c_1, c_2 and w. Figure 11.2 (a) presents this evolution for different combinations of c_1 and c_2 given a fixed inertia weight ($w = 0.4$). This figure indicates that lowest RMSE values are obtained for combinations of $c_2 = 1.7, ..., 2.1$ with $c_1 = 1.4, ..., 1.7$. Figure 11.2 (b) shows the evolution of mean RMSE values for combinations of w and c_1, given a fixed parameter c_2 ($c_2 = 1.9$), for which it can be seen that the lowest mean RMSE value is obtained with $w = 0.4$ and $c_1 = 1.5$. Figure 11.2 (c) shows the evolution of mean RMSE values for different values of w and c_2 when $c_1 = 1.5$. From this figure, the same conclusion can be drawn, *i.e.* the lowest mean RMSE values are obtained for $c_2 = 1.9$ and $w = 0.4$ or $w = 0.5$. Therefore, the final parameter values are chosen to be $c_2 = 1.9$, $c_1 = 1.5$ and $w = 0.4$.

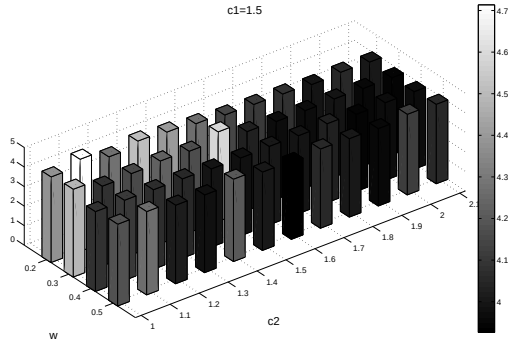
Concerning the maximum number of iterations to be used in the search for the optimal hydrologic parameters, it is not neccessary that the entire population converges to the same position, as this would lead to an excessive number of iterations in which the position of the best particle is not changed any further. A possible condition is that one population member finds the optimal solution. This could be accomplished by requiring that the RMSE value of the best population member does not change during a certain number of iterations. However, in order to avoid situations in which only one member satisfies the above criterion, it is better to require a significant part of the population to converge and hence to increase the probability that the global optimum is found by that part of the population. Accordingly, the implementation used in this case study requires a



(a)



(b)



(c)

Figure 11.2: Dependence of the RMSE on different values of the PSO parameters.

convergence of half of the population as stopping criterion.

11.4. Results and discussion

In this section, the results are presented in detail and our observations are discussed. Firstly, the model parameters are estimated using the reduced data set (Section 11.4.1). Secondly, the model parameters are estimated with the Particle Swarm Optimization algorithm using the hourly data set in Section 11.4.2 and compared with the results in Section 11.4.1.

11.4.1. Model parameter estimation using the reduced data set

For both methods, the hydrologic model parameters are estimated based on the reduced calibration data set, whereas the validation was performed on the hourly validation data set. The results of both calibration algorithms are compared. In the first part of this section, results are presented for which the number of model evaluations in the calibration process was not restricted. the second part of this section then describes the results for which the number of model evaluations was set to a predetermined value. As mentioned above, MWARPE was only used on the basis of the reduced data set. PSO does not have this limitation, and was firstly employed on the reduced data set, and secondly on the hourly data set. The comparison of both calibration methods is based on the overall RMSE values obtained by application of the hydrologic model on the validation data with the hydrologic parameters, acquired by both calibration methods for the calibration data. The non-standardized RMSE values of the individual variables (the different energy balance terms and soil moisture values of the different layers) are presented as well.

Unrestricted number of model evaluations

As the obtained RMSE values, for 20 repetitions, are not normally distributed (p-values < 0.05 are obtained with the Lilliefors test [109]), a non-parametric test was used to search for significant differences between the RMSE values obtained on the basis of both methods. Therefore, the the Wilcoxon rank sum test seems to be appropriate [109].

The RMSE values obtained with the two methods after calibration on the reduced data set, for both calibration periods and validation periods, are given in Table 11.3. Comparison of the mean RMSE values for the calibration period leads to the conclusion that the MWARPE method gives better results than the PSO method (Table 11.3). This is reflected in the result of the Wilcoxon rank sum test: p-values

Table 11.3: Mean RMSE values and their standard deviations obtained for the two calibration and the corresponding validation periods for MWARPE and PSO applied on the reduced data set. The calibration RMSE values are calculated using the reduced data set, while for the validation RMSE values the hourly data set is used.

		First calibration period		Second calibration period	
Variable	Units	MWARPE	PSO	MWARPE	PSO
Overall	-	3.356 ± 0.051	3.420 ± 0.046	4.401 ± 0.005	4.606 ± 0.116
θ_1	-	0.022 ± 0.001	0.024 ± 0.003	0.027 ± 0.000	0.030 ± 0.002
θ_2	-	0.019 ± 0.001	0.020 ± 0.002	0.023 ± 0.000	0.026 ± 0.002
θ_3	-	0.019 ± 0.001	0.019 ± 0.002	0.012 ± 0.000	0.012 ± 0.001
θ_4	-	0.024 ± 0.003	0.022 ± 0.003	0.033 ± 0.001	0.035 ± 0.002
Rn	Wm^{-2}	43.928 ± 0.160	44.277 ± 0.564	39.906 ± 1.560	40.656 ± 4.384
LE	Wm^{-2}	62.466 ± 0.385	64.221 ± 1.000	62.904 ± 0.704	66.620 ± 5.083
H	Wm^{-2}	51.422 ± 0.415	53.005 ± 0.980	54.324 ± 0.872	54.592 ± 1.239
G	Wm^{-2}	9.731 ± 0.116	9.765 ± 0.313	8.627 ± 0.162	9.085 ± 0.294
		Second validation period		First validation period	
Variable	Units	MWARPE	PSO	MWARPE	PSO
Overall	-	6.637 ± 0.124	6.605 ± 0.120	4.750 ± 0.130	5.014 ± 1.467
θ_1	-	0.031 ± 0.002	0.033 ± 0.005	0.030 ± 0.001	0.033 ± 0.014
θ_2	-	0.021 ± 0.001	0.026 ± 0.006	0.032 ± 0.002	0.034 ± 0.016
θ_3	-	0.016 ± 0.001	0.014 ± 0.003	0.029 ± 0.002	0.032 ± 0.019
θ_4	-	0.054 ± 0.003	0.049 ± 0.005	0.032 ± 0.002	0.036 ± 0.025
Rn	Wm^{-2}	55.488 ± 0.616	54.772 ± 1.034	48.768 ± 0.135	48.779 ± 0.561
LE	Wm^{-2}	74.553 ± 2.171	74.248 ± 3.479	60.127 ± 0.351	62.461 ± 2.135
H	Wm^{-2}	58.754 ± 1.910	58.809 ± 2.777	51.385 ± 0.499	53.421 ± 1.677
G	Wm^{-2}	19.780 ± 0.972	19.97 ± 1.282	12.989 ± 0.145	12.826 ± 0.225

< 0.05 were obtained for the first and second calibration period, respectively. Concerning the RMSE values obtained for the respective validation periods, no significant differences were found (p-values > 0.05 were obtained for the first and second validation period, respectively). It should also be noted that the standard deviations are of the same relative order of magnitude for all observed variables. This can be attributed to the standardization of these variables (Table 11.3). These standard deviations are an indication of the variability and uncertainty in the output variables. Table 11.3 shows that the standard deviations are rather small for both calibration methods. They are, however, larger for PSO than for MWARPE.

Table 11.4: Mean values of the observed and simulated data (obtained after application of the hydrologic model with the hydrologic parameter values resulting in the best RMSE values on the validation data, corresponding to the first calibration period) for the different energy balance terms and soil moisture layers, the intercept, slope and r^2 of the linear fit for these output variables and the RMSE and Nash-Sutcliffe model efficiency coefficient for each output variable.

	mean observed data	mean simulated data	Intercept	Slope	r^2	RMSE	Nash-Sutcliffe model efficiency coefficient
θ_1 (MWARPE)	0.120	0.111	-0.002	0.944	0.927	0.026	0.798
θ_1 (PSO)	0.120	0.107	-0.005	0.936	0.906	0.030	0.716
θ_2 (MWARPE)	0.131	0.121	0.000	0.922	0.970	0.020	0.859
θ_2 (PSO)	0.131	0.117	-0.004	0.922	0.959	0.024	0.789
θ_3 (MWARPE)	0.125	0.128	0.009	0.956	0.965	0.017	0.888
θ_3 (PSO)	0.125	0.126	0.005	0.964	0.972	0.014	0.916
θ_4 (MWARPE)	0.119	0.142	0.031	0.933	0.894	0.039	0.539
θ_4 (PSO)	0.119	0.141	0.027	0.955	0.917	0.036	0.611
G (MWARPE)	2.326	6.574	4.618	0.841	0.472	16.235	0.033
G (PSO)	2.326	6.604	4.648	0.841	0.472	16.238	0.033
H (MWARPE)	20.201	10.884	-1.174	0.597	0.597	52.503	0.524
H (PSO)	20.201	9.130	-3.005	0.601	0.586	53.842	0.499
LE (MWARPE)	27.486	37.841	16.052	0.793	0.766	65.099	0.736
LE (PSO)	27.486	39.030	17.032	0.800	0.752	67.613	0.715
Rn (MWARPE)	50.013	55.269	7.413	0.957	0.929	52.303	0.918
Rn (PSO)	50.013	54.731	6.883	0.957	0.930	52.040	0.919

Figure 11.3 presents the scatterplots of the observed versus the simulated data for the different energy balance terms and soil moisture layers, obtained after application of the hydrologic model with the hydrologic parameter values resulting in the best RMSE values on the validation data, corresponding to the first calibration period. Table 11.4 presents the mean observed and simulated value for the different energy balance terms and soil moisture layers, the intercept, slope and r^2 of the linear fit for these output variables and the RMSE and Nash-Sutcliffe model efficiency coefficient for each output variable. The Nash-Sutcliffe model efficiency coefficient E_i is calculated as follows

$$E_j = 1 - \frac{\sum_{i=1}^{m_j}(y_{j,i} - \hat{y}_{j,i})^2}{\sum_{i=1}^{m_j}(y_{j,i} - \overline{y}_j)^2} \tag{11.14}$$

with m_j the number of data points of variable j, $\boldsymbol{y}_j \in \mathbb{R}^{m_j}$ the observed data of variable j, $\hat{\boldsymbol{y}}_j \in \mathbb{R}^{m_j}$ the simulated data of variable j and $\overline{y}_j$ the mean of the observed data of variable j. This coefficient is used to determine the predictive power of a hydrologic model. This coefficient lies in the interval $[-\infty, 1]$. The closer this coefficient is to 1, the more accurate the model predictions are, when the coefficient is equal to 0 the model predictions are as accurate as using the mean to model the observed data. As stated above, the model performance is similar to the performance of more complicated models that have been applied on the data set [105]. Since for both calibration algorithms the statistics are similar, the decision was made to focus on the RMSE for the comparison of both methods. Figure 11.3 shows that the difference between the simulated and observed data obtained for both calibration methods is similar for the energy balance terms as well as for the soil moisture layers. The distribution of the total energy balance (Rn) is very well modelled, however, the distribution of the different components in this energy balance is less well recovered by the simulations. For the energy balance term H and the soil moisture layers θ_1 and θ_2 there is an underestimation and for the energy balance terms G and LE and the soil moisture layers θ_3 and θ_4 a small overestimation can be noticed. The simulated data of the energy balance term Rn and the soil moisture layer at a depth of 9 cm (θ_2) are now discussed in more detail.

Figures 11.4 and 11.5 show parts of the observed and simulated data for the energy balance term Rn and the soil moisture layer at a depth of 9 cm, respectively, after application of the hydrologic model with the hydrologic parameters values resulting in the best RMSE values on the validation data, corresponding to the first calibration period. The results of the other energy balance terms and soil moisture layers are similar and are not presented. Similar results were obtained when the second period was used as calibration, therefore these results are not shown either. Concerning the results for the energy balance term Rn (Figure 11.4), differences between the results from both methods are completely negligible in comparison to the deviations between the simulated and observed data. The deviations between

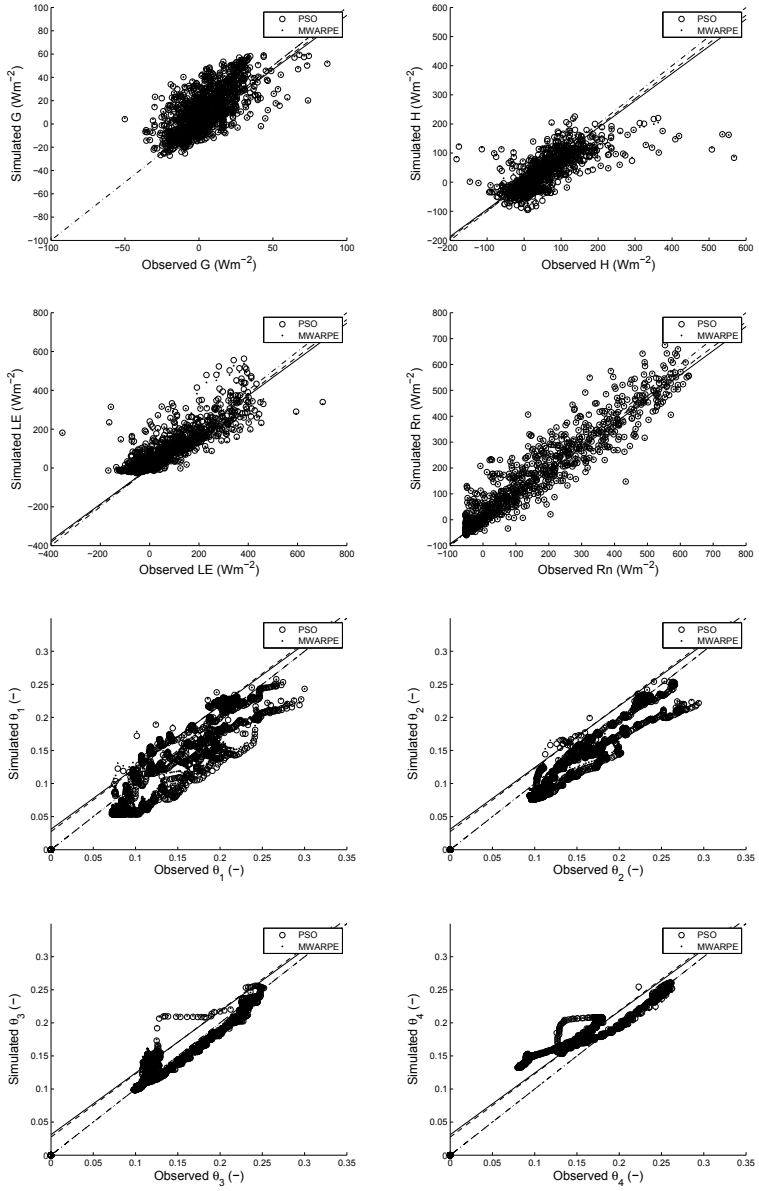


Figure 11.3: Scatterplot of the observed and simulated data of the energy balance terms and the soil moisture layers. The simulated data are obtained after application of the hydrologic model with the hydrologic parameter values resulting in the best RMSE values on the validation data, corresponding to the first calibration period. The solid lines represent the linear fit of the data obtained with MWARPE, the dashed lines represent the linear fit of the data obtained with PSO and the solid-dashed lines represent the perfect fit.

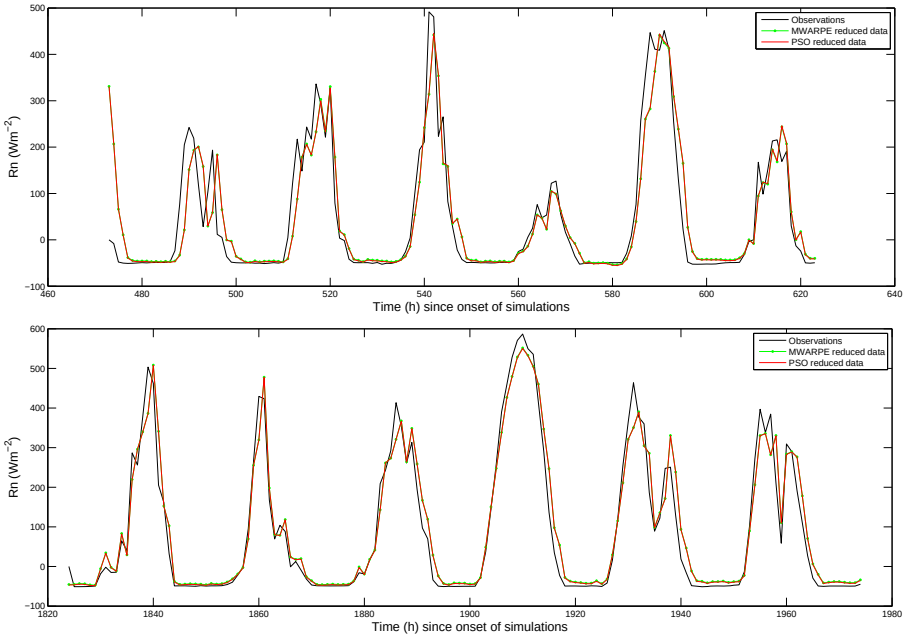


Figure 11.4: Time series of the energy balance term Rn for a part of the first calibration period (4 AM April 20 - 4 PM April 27, for the energy balance terms no data were available before this period) (top panel) and a part of the corresponding validation period (8 PM June 15 - 12 PM June 22) (bottom panel), this for the simulation data corresponding to the lowest RMSE values on the validation data.

the simulated and observed data are rather small, and the order of magnitude remains constant over time. These conclusions are valid for both the calibration period and validation period and lead to the conclusion that the hydrologic model offers a very good description of the energy balance term Rn, both on a qualitative as a quantitative level.

Contrarily, concerning the soil moisture data (Figure 11.5), a clear difference between the model results based on the hydrologic model parameters obtained with MWARPE and PSO is noticed. The order of magnitude of this difference is larger for the validation period as compared to the calibration period. The results obtained with MWARPE are closer to the observed data, as is reflected in the separate and total RMSE values (Tables 11.3). Both methods yield hydrologic parameters that result in a similar behaviour of the hydrologic model with respect to θ_2. However it should be stated that the model only succeeds in mimicking the global behaviour of the measured values. A noticeable discrepancy is the fact that the simulated data are much smoother than the observed data and do not show hourly fluctuations. This can either be due to the use of only one soil moisture observation per day for calibration, or a deficiency in the hydrologic model if it does not allow for rapid oscillations in the soil moisture content due to for example

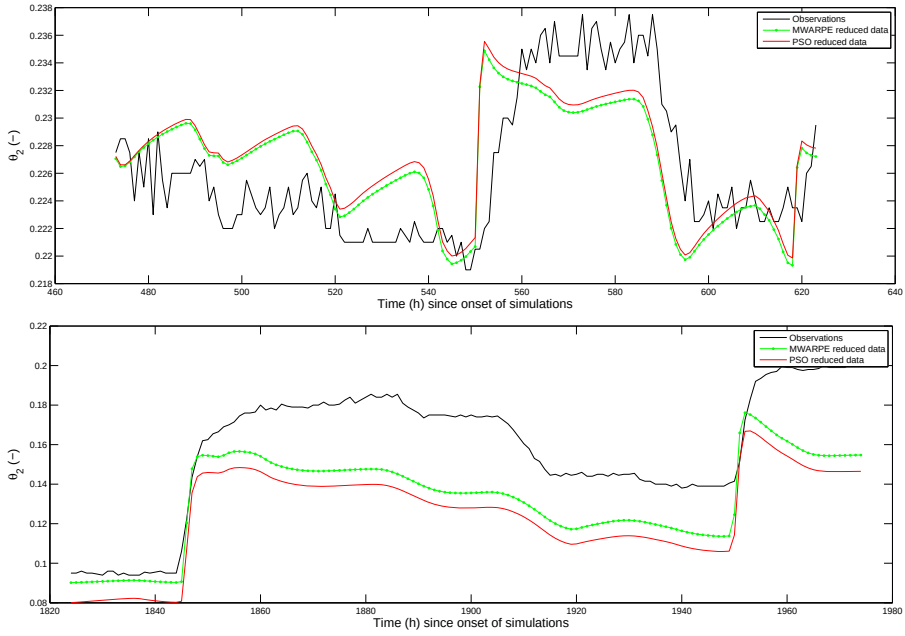


Figure 11.5: Time series of the soil moisture layer values at a depth of 9 cm for a part of the first calibration period (4 AM April 20 - 4 PM April 27) (top panel) and a part of the corresponding validation period (8 PM June 15 - 12 PM June 22) (bottom panel), this for the simulation data corresponding to the lowest RMSE values on the validation data.

the use of uniform model parameters for the entire soil profile, the assumption of the absence of roots in the soil profile, and uncertainty in the constitutive equations relating the pressure head to the hydraulic conductivity and the soil moisture content. The same conclusions are valid for the soil moisture data at the other layers (data not shown).

Figure 11.6 presents the boxplots of the different hydrologic parameters obtained with both methods for the first (top panel) and second (bottom panel) calibration period. These boxplots reflect the uncertainty in and the variability of the hydrologic parameters. A boxplot is a representation of five numbers: the minimum value, the first quartile, the median or second quartile, the third quartile and the maximum value. The crosses that lie outside the box are considered as outliers, and do not contribute to the minimum and maximum value. It is possible that the five values of the boxplot are indistinguishable on the figure, and that only a single line and the outliners are visible (for example for parameter K_s in Figure 11.6 (bottom panel)). Concerning the first calibration period, a large difference between both methods for the values for ψ_c, κ, C, r_c and f_v is noticed. The boxplots of the other hydrologic parameters overlap to a large extent (Figure 11.6 (top panel)). Concerning the second calibration period, little or no overlap is noticed for the parameters λ, ψ_c, K_s, α, κ, C and f_h (Figure 11.6 (top panel)). The parameter f_d differs only slightly in both cases, although this can be a consequence of the large spread for this variable. The fact that in most cases the resulting RMSE values of both methods do not differ strongly (as reflected by the standard deviations in Table 11.3), although the corresponding parameter estimates show little overlap, indicates that the model can result in similar output values for different combinations of input parameters. This problem is referred to as equifinality [110].

Important to notice is that the range of both the estimated parameters and the resulting RMSE values is, on average, larger for PSO, without this resulting in a reduced average model performance. It can be argued that a calibration algorithm that leads to several possible parameter configurations, all resulting in similar model performance, can be considered to be more informative than an algorithm that tends to converge to rather the same parameter configuration. This is supported by the fact that in some cases PSO outperforms MWARPE with respect to the validation data.

For the first (second) calibration period the number of model evaluations for MWARPE is 4572 (5334) and for PSO 1140 (1260). This means that MWARPE needs ca. 4 times as many model evaluations as PSO does. Figure 11.7 represents the behaviour of the RMSE of the energy balance component Rn, obtained with MWARPE, in function of the number of iterations. This figure shows that after 4 iterations the RMSE for Rn is rather stable. For the other energy balance terms and soil moisture layers similar results were obtained. The distinction of different numbers of model evaluations will be eliminated in the next section by restricting

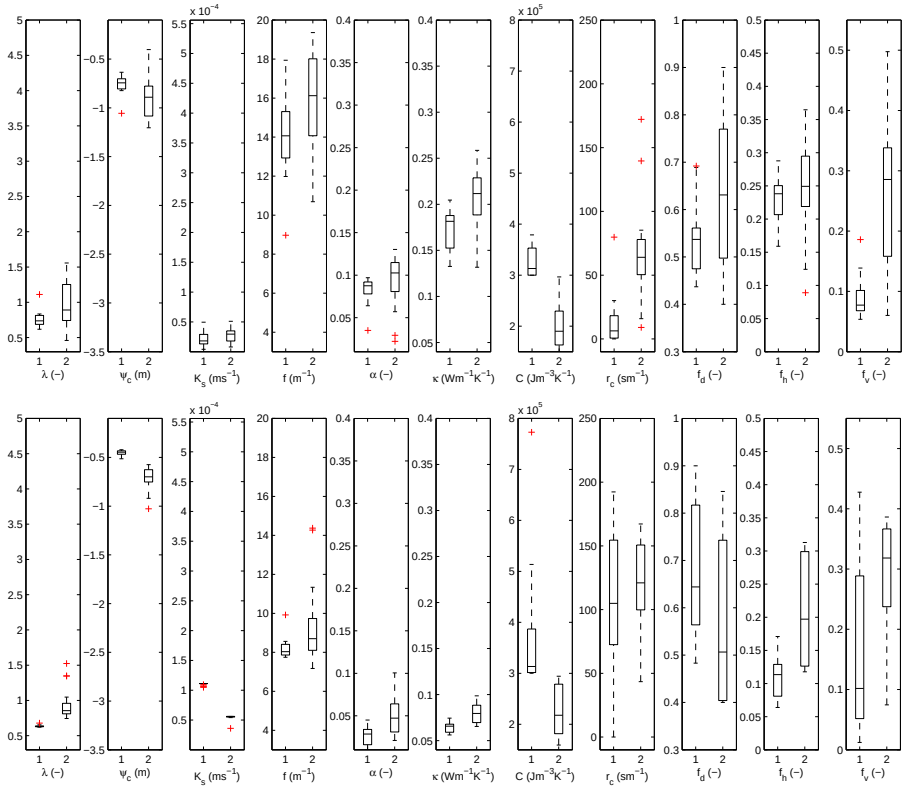


Figure 11.6: Boxplot of the hydrologic model parameters obtained with MWARPE (1) and with PSO (2) for the first calibration period (top panel) and for the second calibration period (bottom panel).

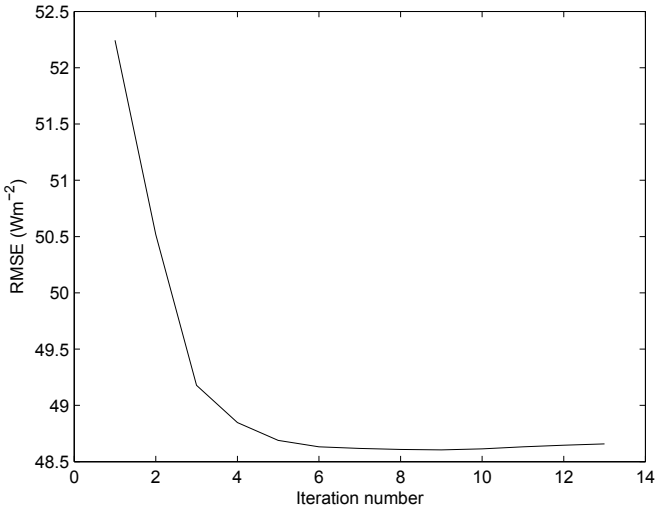


Figure 11.7: Evolution of the RMSE for the energy balance term *Rn* obtained with MWARPE.

the number of model evaluations for both methods.

Restricted number of model evaluations

The number of model evaluations was restricted according to the following scheme. The median of the number of model evaluations of the 20 repetitions for the PSO method was 1140 and 1260 for the first and second calibration period, respectively. MWARPE needs 12 model runs, per iteration step, to calculate the Jacobian, more precisely 1 undisturbed run and 11 disturbed runs for the different hydrologic model parameters. As the purpose is to restrict the number of model evaluations as close as possible to these medians, and every starting point of MWARPE needs 12 model evaluations for every iteration step, the total number of model evaluations has to be a multiple of 360 ($= 12 \cdot 30$). The number of model evaluations was hence rounded to the nearest multiple of 360 to both medians, which is 1080. From this number, a maximum of 3 iterations for MWARPE, and 36 iterations for PSO can be concluded.

Table 11.5 lists the mean RMSE values obtained after application of the hydrologic model with the hydrologic parameters acquired by both methods. The standard deviations of these mean RMSE values are also presented. The variability on the output variables is higher with PSO used as calibration method than with MWARPE used as calibration method. This table shows that the mean RMSE values for the parameters obtained with MWARPE for the first calibration period are lower compared to the mean RMSE values for the parameters obtained with PSO. This has been confirmed by the Wilcoxon rank sum test (p-value < 0.05).

Table 11.5: Mean RMSE values and their standard deviations obtained for the two calibration and the corresponding validation periods for MWARPE and PSO applied on the reduced data set with a restricted number of model evaluations. The calibration RMSE values are calculated using the reduced data set, while for the validation RMSE values the hourly data set is used.

		First calibration period		Second calibration period	
Variable	Units	MWARPE	PSO	MWARPE	PSO
Overall	-	3.400 ± 0.022	3.471 ± 0.089	4.545 ± 0.074	4.625 ± 0.168
θ_1	-	0.022 ± 0.002	0.025 ± 0.004	0.027 ± 0.003	0.030 ± 0.003
θ_2	-	0.019 ± 0.001	0.021 ± 0.003	0.023 ± 0.003	0.025 ± 0.002
θ_3	-	0.019 ± 0.001	0.019 ± 0.002	0.013 ± 0.001	0.013 ± 0.004
θ_4	-	0.024 ± 0.002	0.022 ± 0.003	0.035 ± 0.003	0.036 ± 0.003
Rn	Wm^{-2}	44.559 ± 0.724	45.147 ± 1.019	42.611 ± 2.721	39.604 ± 1.930
LE	Wm^{-2}	63.823 ± 1.392	65.038 ± 1.546	63.482 ± 1.090	64.140 ± 1.490
H	Wm^{-2}	52.814 ± 0.680	52.511 ± 0.993	55.097 ± 2.253	54.016 ± 0.706
G	Wm^{-2}	9.950 ± 0.348	10.181 ± 0.954	9.257 ± 0.484	8.931 ± 0.319
		Second validation period		First validation period	
Variable	Units	MWARPE	PSO	MWARPE	PSO
Overall	-	6.603 ± 0.286	6.910 ± 0.294	4.983 ± 0.441	5.090 ± 1.157
θ_1	-	0.030 ± 0.004	0.032 ± 0.005	0.032 ± 0.004	0.035 ± 0.014
θ_2	-	0.021 ± 0.003	0.025 ± 0.007	0.034 ± 0.007	0.036 ± 0.015
θ_3	-	0.015 ± 0.002	0.015 ± 0.004	0.032 ± 0.007	0.033 ± 0.015
θ_4	-	0.053 ± 0.004	0.051 ± 0.006	0.035 ± 0.006	0.036 ± 0.016
Rn	Wm^{-2}	54.931 ± 1.397	55.045 ± 1.853	48.758 ± 0.377	48.797 ± 0.232
LE	Wm^{-2}	74.77 ± 4.212	70.381 ± 3.886	61.063 ± 1.090	61.224 ± 1.077
H	Wm^{-2}	59.908 ± 4.935	59.836 ± 3.197	51.382 ± 0.979	52.287 ± 0.728
G	Wm^{-2}	20.195 ± 2.366	23.741 ± 2.556	13.017 ± 0.451	12.931 ± 0.142

Concerning the RMSE values obtained on the validation data, the same conclusions can be drawn (p-value < 0.05). Concerning the second period and corresponding validation period, no significant differences were observed (p-value > 0.05 for calibration and p-value > 0.05 for validation).

Next, the influence of limiting the number of model evaluations can also be tested for significance. It is to be expected that the RMSE values should be lower when the number of model evaluations is not restricted. This is true for almost all cases. Test results confirmed this expectation for PSO for the first calibration period and corresponding second validation period (p-values < 0.05) and for MWARPE for both calibration periods (p-values < 0.05 for the first and second period). However, in the case of MWARPE for the second validation period (Tables 11.3 and 11.5), a lower RMSE value has been obtained for the restricted number of model evaluations (p-value < 0.05), in contrast to the fact that the calibration RMSE is significantly higher (p-value < 0.05). It is an overall observation that MWARPE performs well in reducing the RMSE of the calibration data, but that the corresponding effect on the validation RMSE is negligible (in comparison with PSO).

For the energy balance term Rn, the simulated data obtained with both calibration methods are very similar and approximate the observed data of Rn. This is the case for both calibration and validation periods (see first paragraph of Section 11.4.1). Similar results were found between the simulation results for the other energy

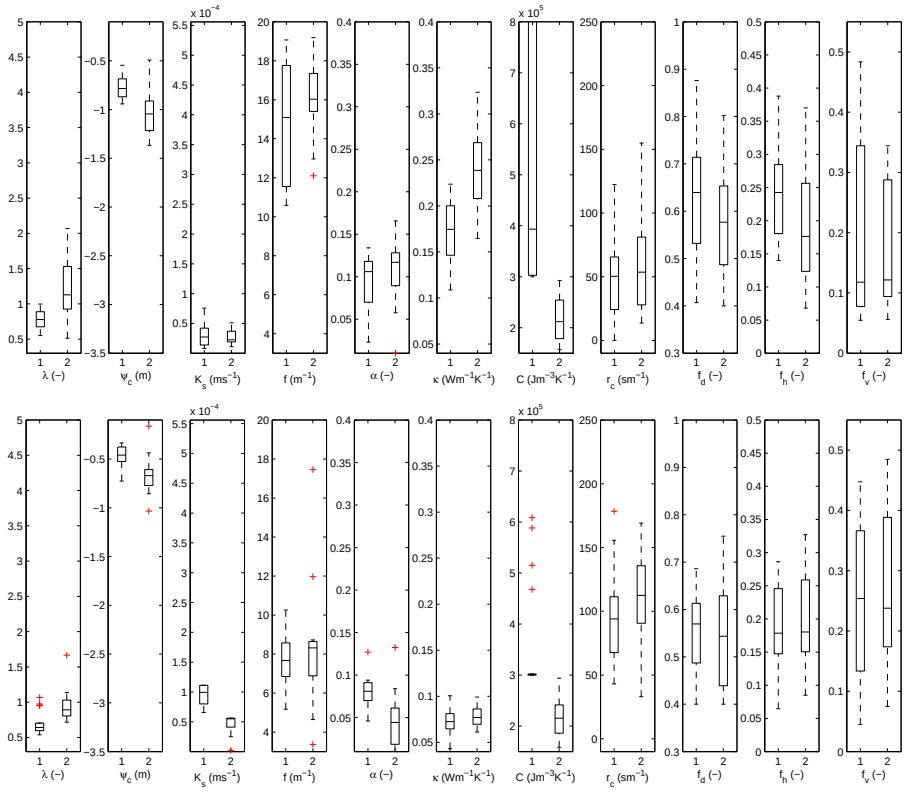


Figure 11.8: Boxplot of the hydrologic model parameters obtained with MWARPE (1) and with PSO (2) with a restricted number of model evaluations for the first calibration period (top panel) and for the second calibration period (bottom panel).

balance terms, therefore these results were not shown.

The simulated data of the soil moisture layer at a depth of 9 cm, obtained with the MWARPE method, are once again better than when the PSO method is used as calibration method. All remarks of the previous section (where the number of evaluations was not restricted), still apply (see first paragraph of Section 11.4.1).

Figure 11.8 presents the boxplots of the hydrologic parameters obtained with both methods for the first (top panel) and second (bottom panel) calibration period. For the first calibration period, the parameters K_s, f, α, r_c, f_d, f_h and f_v have a strong overlap. For the second calibration period, little or no difference is noticed for the parameters f, κ, r_c, f_d, f_h and f_v. An obvious consequence of restricting the number of model evaluations is that the spread on the parameter estimates has grown for MWARPE, and is now of the same order as the spread of the PSO estimates.

11.4.2. Model parameter estimation using the hourly data set

As already mentioned, PSO is not restricted to the use of the reduced data set. In this section a comparison of PSO on the reduced and hourly data set and the application of MWARPE on the reduced data set is made. This comparison is made by the use of the Kruskall-Wallis test [109]. This test indicates whether significant differences are present between PSO on the reduced data set, PSO on the hourly data set and MWARPE on the reduced data set. In order to determine which results are significantly different, a Bonferroni correction has to be applied. The Bonferroni correction states that, if n different hypotheses are to be tested with a collective significance level α, the individual hypotheses are to be compared pairwise with a significance level of α/n. As the above-described applications differ in the use of calibration data, only RMSE values obtained on the validation data sets will be compared. In this section, the results of the energy balance term Rn and the soil moisture layer at a depth of 9 cm are presented in the graphs. The results of the other energy balance terms and soil moisture data were similar and are not presented.

Unrestricted number of model evaluations

Table 11.6 shows the mean validation RMSE values for the different calibration methods. Although MWARPE results in lower mean RMSE values for the first validation period, no significant differences were found. For the second validation period, the lowest mean validation RMSE value is obtained for the application of PSO on the hourly data set. However, only a significant difference was found between this application and the application of MWARPE (p-value < 0.05).

In Figures 11.9 and 11.10 the simulated data obtained with both methods are plotted against the observed data for the energy balance term Rn and for the soil moisture at a depth of 9 cm, respectively. Figure 11.9 illustrates that the the difference between the methods is almost negligible. This is true for the calibration period as well as for the validation period.

Concerning the soil moisture data θ_2, the difference between the results obtained with PSO using the hourly data set and the results obtained with PSO using the reduced data set is very small (Figure 11.10). In this case MWARPE outperforms PSO for the validation period, as can be seen in Table 11.6 as well. The simulated data obtained using PSO with the hourly data set are also much smoother than the observed data and do not show hourly fluctuations either. Since in this case all hourly observations are used, the mismatch between the observed and simulated soil moisture values must be attributed to the hydrologic model (see Section 11.4.1).

Figure 11.11 presents the estimated values of the hydrologic parameters obtained

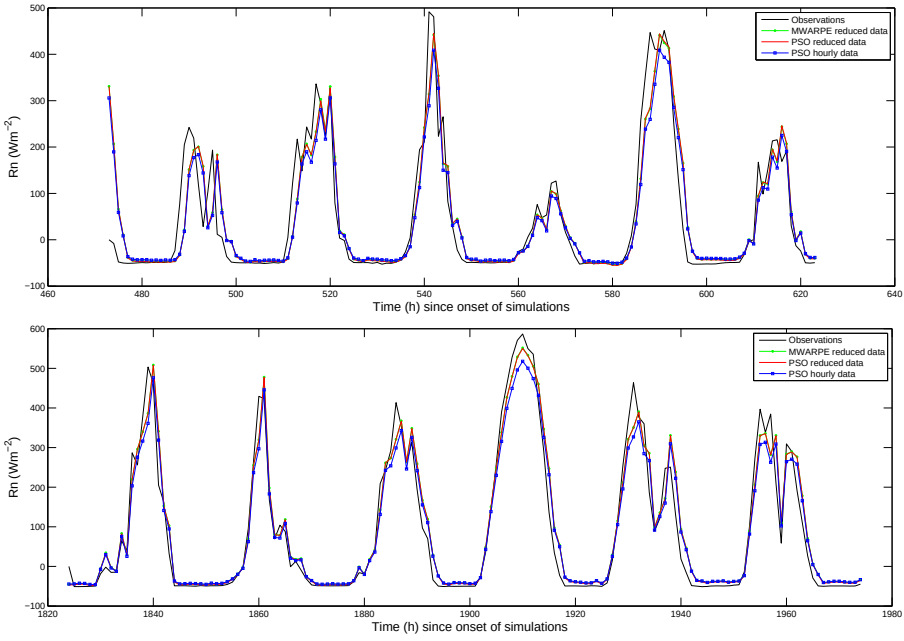


Figure 11.9: Time series of the energy balance term Rn for a part of the first calibration period (4 AM April 20 - 4 PM April 27, for the energy balance terms no data were available before this period) (top panel) and a part of the corresponding validation period (8 PM June 15 - 12 PM June 22) (bottom panel) obtained with PSO applied on the hourly data set and the reduced data set and MWARPE applied on the reduced data set, this for the simulation data corresponding to the lowest RMSE values on the validation data.

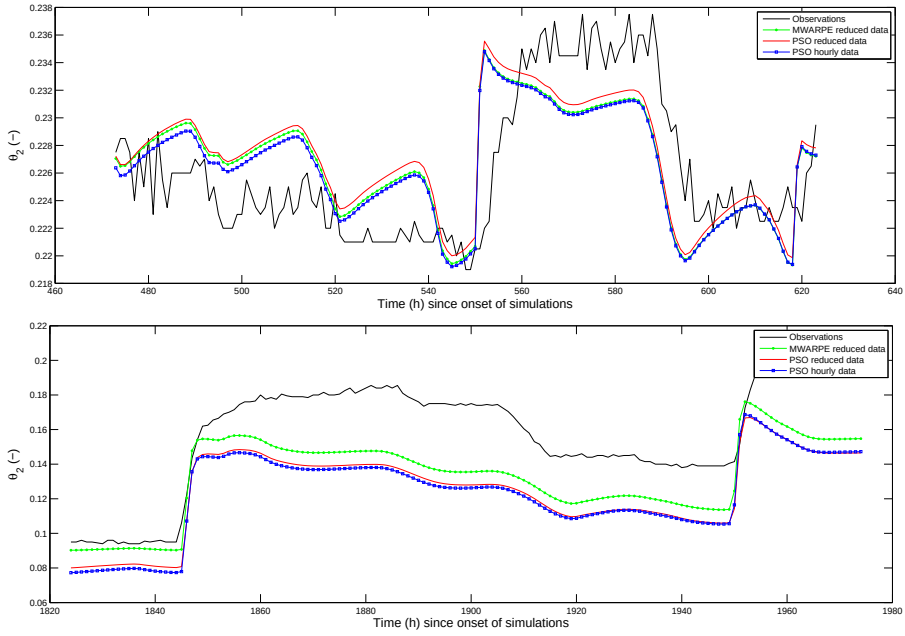


Figure 11.10: Time series of the soil moisture layer values at a depth of 9 cm for a part of the first calibration period (4 AM April 20 - 4 PM April 27) (top panel) and a part of the corresponding validation period (8 PM June 15 - 12 PM June 22) (bottom panel) obtained with PSO applied on the hourly data set and the reduced data set and MWARPE applied on the reduced data set, this for the simulation data corresponding to the lowest RMSE values on the validation data.

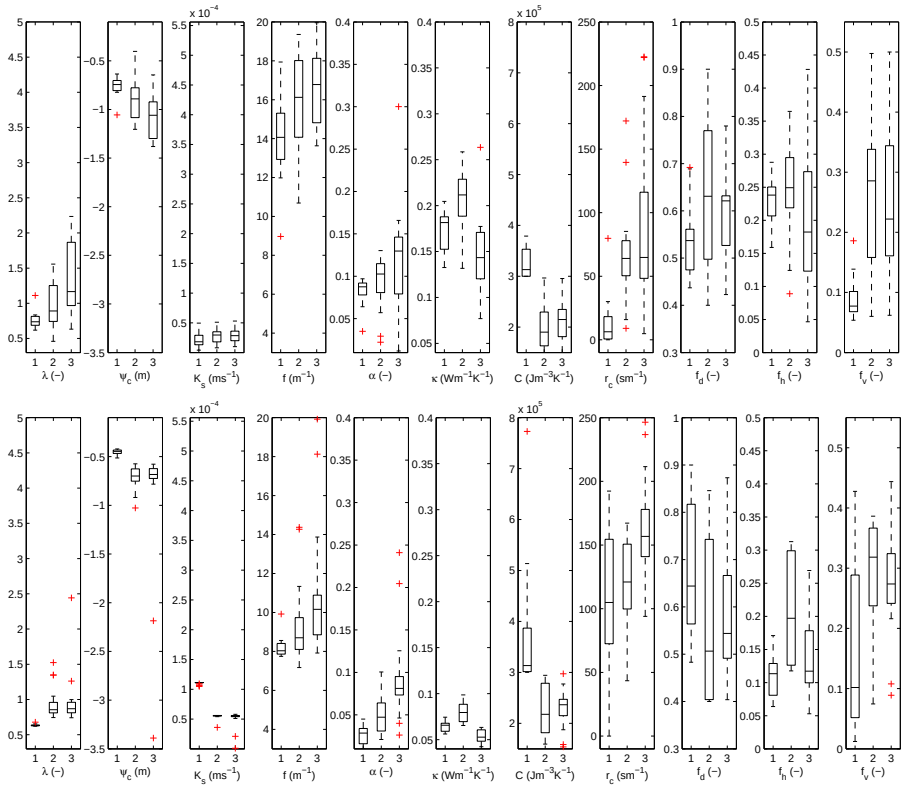


Figure 11.11: Boxplot of the hydrologic model parameters obtained with MWARPE (1) applied on the reduced data set, with PSO applied on the reduced data set (2) and with PSO applied on the hourly data set (3) for the first calibration period (top panel) and for the second calibration period (bottom panel).

with both methods for the first (top panel) and second (bottom panel) calibration period. For both calibration periods, similar conclusions can be drawn. The estimated parameters of the two applications of PSO show a stronger overlap with each other than with the estimates of MWARPE. Furthermore, the spread of the parameters obtained with PSO applied on the hourly data set is larger than that of the parameters resulting from its application on the reduced data set. Again we can conclude that the model can result in rather the same output for strongly different combinations of input parameters.

Table 11.6: Mean RMSE values and their standard deviations obtained for the two validation periods for MWARPE applied on the reduced data set, PSO applied on the reduced data set and PSO applied on the hourly data set. These validation RMSE values are calculated using the hourly data set.

Variable	Units	MWARPE applied on the reduced data set	PSO applied on the reduced data set	PSO applied on the hourly data set	MWARPE applied on the reduced data set	PSO applied on the reduced data set	PSO applied on the hourly data set
Overall	-	4.750 ± 0.130	5.014 ± 1.467	5.423 ± 1.991	6.637 ± 0.124	6.605 ± 0.120	6.461 ± 0.419
θ_1	-	0.030 ± 0.001	0.033 ± 0.014	0.036 ± 0.020	0.031 ± 0.002	0.033 ± 0.005	0.036 ± 0.004
θ_2	-	0.032 ± 0.002	0.034 ± 0.016	0.038 ± 0.022	0.021 ± 0.001	0.026 ± 0.006	0.030 ± 0.006
θ_3	-	0.029 ± 0.002	0.032 ± 0.019	0.037 ± 0.025	0.016 ± 0.001	0.014 ± 0.003	0.013 ± 0.002
θ_4	-	0.032 ± 0.002	0.036 ± 0.025	0.042 ± 0.032	0.054 ± 0.003	0.049 ± 0.005	0.049 ± 0.004
Rn	Wm^{-2}	48.768 ± 0.135	48.779 ± 0.561	50.289 ± 2.947	55.488 ± 0.616	54.772 ± 1.034	55.542 ± 3.691
LE	Wm^{-2}	60.127 ± 0.351	62.461 ± 2.135	62.499 ± 3.401	74.553 ± 2.171	74.248 ± 3.479	72.807 ± 6.263
H	Wm^{-2}	51.385 ± 0.499	53.421 ± 1.677	51.763 ± 1.209	58.754 ± 1.910	58.809 ± 2.777	62.505 ± 10.959
G	Wm^{-2}	12.989 ± 0.145	12.826 ± 0.225	13.805 ± 0.260	19.780 ± 0.972	19.970 ± 1.282	15.393 ± 1.638

Restricted number of model evaluations

As was determined in Section 11.4.1, a maximum number of 1080 model evaluations is set. Table 11.7 presents the RMSE values obtained by application of the hydrologic model, with the hydrologic parameters acquired by the two applications of PSO and one application of MWARPE on the validation data set.

For the first validation period, no significant differences in RMSE values were found. For the second validation period, RMSE values obtained with PSO applied on the hourly data set are significantly lower than those obtained with PSO applied on the reduced data set and those obtained with MWARPE (p-values < 0.05, respectively).

For the results of Rn, similar conclusions can be drawn as in Section 11.4.2. For the calibration period, the simulated soil moisture data θ_2 (Figure 11.12) are during time steps 480-550 on average closer to the observed data with PSO applied on the hourly data set. However, during time steps 550-620 the situation is reversed. The different applications of the simulated data show the same evolution and only differ by a constant shift. For the validation period, the results obtained with PSO applied on the hourly data set are closer to the observed data for the full range of the time series shown in the graph.

The results concerning the estimates of the hydrologic parameters obtained with both methods are similar to those of Section 11.4.2. Therefore, these are not presented.

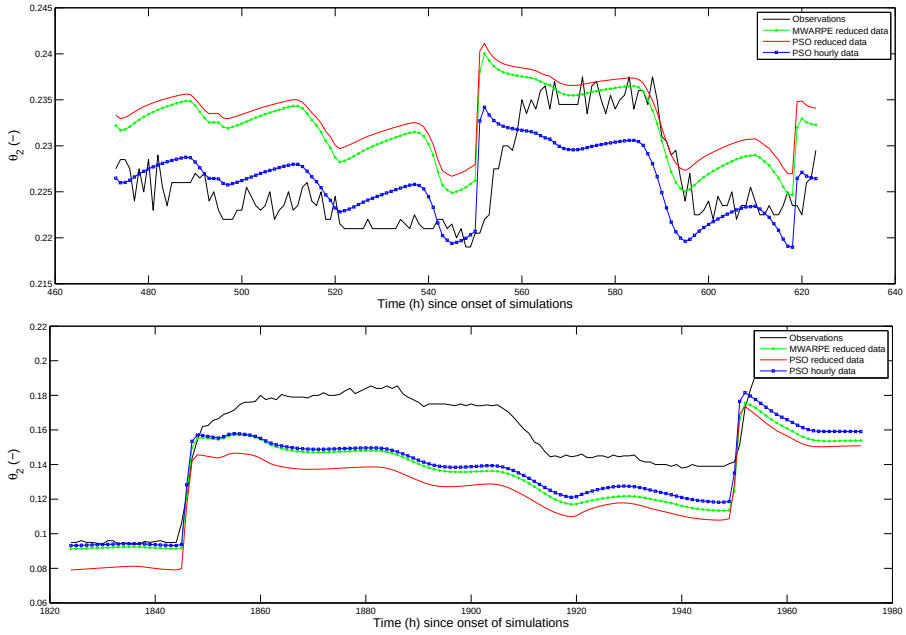


Figure 11.12: Time series of the energy balance term *Rn* for a part of the first calibration period (4 AM April 20 - 4 PM April 27, for the energy balance terms no data were available before this period) (top panel) and a part of the corresponding validation period (8 PM June 15 - 12 PM June 22) (bottom panel) obtained with PSO applied on the hourly data set and the reduced data set and MWARPE applied on the reduced data set with a restricted number of model evaluations, this for the simulation data corresponding to the lowest RMSE values on the validation data.

Table 11.7: Mean RMSE values and their standard deviations obtained for the two validation periods for MWARPE applied on the reduced data set, PSO applied on the reduced data set and PSO applied on the hourly data set, all applied with a restricted number of model evaluations. These validation RMSE values are calculated using the hourly data set.

Variable	Units	First validation period			Second validation period		
		MWARPE applied on the reduced data set	PSO applied on the reduced data set	PSO applied on the hourly data set	MWARPE applied on the reduced data set	PSO applied on the reduced data set	PSO applied on the hourly data set
Overall	-	4.983 ± 0.441	5.090 ± 1.157	5.448 ± 1.752	6.603 ± 0.286	6.910 ± 0.294	6.400 ± 0.221
θ_1	-	0.032 ± 0.004	0.035 ± 0.014	0.037 ± 0.018	0.030 ± 0.004	0.032 ± 0.005	0.031 ± 0.004
θ_2	-	0.034 ± 0.007	0.036 ± 0.015	0.039 ± 0.021	0.021 ± 0.003	0.025 ± 0.007	0.024 ± 0.006
θ_3	-	0.032 ± 0.007	0.033 ± 0.015	0.037 ± 0.022	0.015 ± 0.002	0.015 ± 0.004	0.014 ± 0.003
θ_4	-	0.035 ± 0.006	0.036 ± 0.016	0.041 ± 0.026	0.053 ± 0.004	0.051 ± 0.006	0.050 ± 0.006
Rn	Wm^{-2}	48.758 ± 0.377	48.797 ± 0.232	50.687 ± 4.482	54.931 ± 1.397	55.045 ± 1.853	55.942 ± 1.774
LE	Wm^{-2}	61.063 ± 1.090	61.224 ± 1.077	63.239 ± 4.354	74.770 ± 4.212	70.381 ± 3.886	70.309 ± 3.476
H	Wm^{-2}	51.382 ± 0.979	52.287 ± 0.728	52.155 ± 1.101	59.908 ± 4.935	59.836 ± 3.197	63.190 ± 6.259
G	Wm^{-2}	13.017 ± 0.451	12.931 ± 0.142	13.797 ± 0.254	20.195 ± 2.366	23.741 ± 2.556	17.166 ± 1.745

12

Conclusion

In this part of this dissertation, two calibration methods, MWARPE and PSO, are compared for the parameter estimation of a simple hydrologic water and energy balance model. These algorithms are thoroughly discussed in Chapter 10. MWARPE does not depend on the definition of an objective function and does not require an additional parameter identification task. However, MWARPE relies on the inversion of possibly large matrices, which has two unfavourable consequences. Firstly, as this operation is time consuming, MWARPE is the slower of the two candidates. Secondly, completing this operation in acceptable time and with an acceptable amount of resources requires the use of a reduced data set, *i.e.* not all the available data can be used for training the model. On the other hand, PSO has the advantage that it is easy to understand and implement. However, PSO requires the determination of an optimal value for the different parameters involved in the algorithm, which can be a time-consuming task. Furthermore, PSO depends on the definition of an objective function. Particularly, if multiple objectives are involved, no unique or ideal way exists to define the overall objective function.

In Chapter 11, these algorithms are applied on a case study, namely the calibration of a relatively simple process-based water and energy balance model. This case study tested both methods in three cases. In the first case the two methods were applied on a reduced hydrologic data set. In the second case the number of model evaluations was restricted for both methods. In the third case, the added value of taking into account all hourly observations for PSO is investigated.

In most cases, the mean calibration RMSE value is significantly lower when MWARPE is used instead of PSO. However, this significant difference does, in most cases, not transfer to the RMSE values on the corresponding validation data sets. From this, it can be concluded that both methods are equally capable of calibrating the hydrologic model, but that in some cases the use of MWARPE may lead to an overfit. This is illustrated by a slightly poorer performance on validation data, using hydrologic parameters that result in a very good model fit on the calibration data. For the second case, it was found that the effect of restricting the number of model evaluations was negligible. As for the third case, lower RMSE values were obtained when PSO was applied on the hourly data set and compared to the results obtained by both methods applied on the reduced data set. However, this difference is not always significant and is rather small.

Overall, it can be concluded that, for this hydrologic case study, both calibration methods yield more or less similar results with a more practical applicability for PSO. Therefore, one might prefer to use PSO for parameter estimation. However, one should furthermore bear in mind that the parameters inherent to PSO need to be re-estimated on different case studies.

It can also be stated that the simple hydrologic model used in this study leads to a similar performance as more complicated land surface models [105]. If an efficient parameter estimation algorithm needs to be chosen for land surface models, one could argue that this search could be performed using simplified versions, in order to reduce computing time.

Part IV

Uncertainty propagation

13

Introduction

In 1992, Zimmerman defined uncertainty as 'the lack of necessary information to quantitatively and qualitatively ... describe, prescribe or predict deterministically and numerically a system, its behaviour or other characteristics' [111]. Introducing uncertainty into mathematical models is a subject of great interest to many engineering applications (see among others [112, 113, 114, 115, 116]). In engineering modelling, uncertainty can be divided into two groups, namely aleatoric and epistemic uncertainties [117, 118, 119]. Aleatoric or stochastic uncertainty is the natural randomness in a process; it is the inherent variability of some phenomena and cannot be reduced by enhancing the available knowledge. This kind of uncertainty can be described by probability theory. Epistemic or systematic uncertainty stems from incomplete knowledge. This type of uncertainty is related to the state of knowledge and can be reduced by improving knowledge about the system. In this dissertation we will restrict ourselves to epistemic uncertainty. Fuzzy set theory is developed to incorporate epistemic uncertainty into mathematical models. Even the most complex model of a real system necessarily involves a series of assumptions and approximations, which are required to compensate for our incomplete understanding of the real world. These assumptions and approximations should be handled as uncertain variables and this uncertainty should be propagated through the model. This can be done by representing the assumptions and approximations of the uncertain variables by fuzzy quantities. Uncertainty can be introduced in the simulation process of systems by Zadeh's extension principle [120] that extends functions of real numbers to functions with fuzzy quantities as arguments.

However, the application of the extension principle to compute with fuzzy quantities is a complex matter. Luckily, a more practical approach operating directly on α-cuts was established by Nguyen [121] and is applicable to continuous functions and upper semi-continuous fuzzy quantities with compact support describing non-interactive variables as inputs. It effectively turns computing with fuzzy intervals into interval analysis [122] on α-cuts. Fullér and Keresztfalvi (1991) [123] generalized Nguyen's theorem for interactive variables, described by triangular norms, as input. In general, however, the result is approximative merely, since only a finite number of α-cuts can be considered. For the common arithmetic operations, such as addition and multiplication, this approach leads to a kind of layered interval arithmetic.

In rare cases, exact formula can be established avoiding to resort to α-cuts (see e.g. [124, 125]).

Several practical implementations of the extension principle based on α-cuts are available for (locally) monotone continuous functions of non-interactive variables described by fuzzy intervals. The vertex method was developed for computing with monotone continuous functions of non-interactive variables described by fuzzy intervals, and can be extended for non-monotone continuous functions by performing an extreme value analysis [126, 127, 128]. However, this is often not possible, for instance when dealing with more complex functions. As an alternative to the vertex method, the transformation method was also developed for computing with monotone functions [129]. Yet another approach for (locally) monotone continuous functions is to make use of gradual numbers [130, 131]. Essential to all these approaches is the fact that a monotone function defined on a hyperrectangle reaches its extremal values in some of the vertices of that hyperrectangle, requiring the evaluation of a finite number of points only. However, this is not necessarily the case for non-monotone functions and an optimization algorithm is hence needed to search for these extremal values [132, 133].

Alternative methods that do not rely on a decomposition into α-cuts are available in literature. In these approaches, the fuzzy input intervals are decomposed into smaller fuzzy intervals, based on a partitioning of the universe on which these fuzzy input intervals are defined. Next, within each partition, the function is replaced by an approximation, for example a monotone function, such that its extension to fuzzy intervals can easily be evaluated. While this approach has been successfully applied for univariate functions [134, 135], its major drawback is that the number of partitions or samples rises exponentially with the number of dimensions. Even with advanced approximation techniques [136, 137], this approach quickly becomes unfeasible when the number of dimensions grows or when the function has strong local features or shows an oscillatory behavior.

The objective of this part of the dissertation is to develop a computationally efficient Fuzzy Calculator based on Nguyen's approach. Four optimization algorithms are compared to determine the minimum and maximum of the function for different α-cuts: (1) Gradient Descent based on Sequential Quadratic Programming (GD) [138, 13], which is a local optimization algorithm, (2) the Simplex-Simulated Annealing approach (SIMPSA) [34], a global heuristic optimization algorithm, (3) Particle Swarm Optimization (PSO) [28], a global heuristic population-based optimization algorithm and (4) a combination of Particle Swarm Optimization and Gradient Descent based on Sequential Programming (PSO_GD). These optimization algorithms were discussed in detail in Chapter 3. In particular, the capability of PSO to solve a complex optimization problem was already tested in Part III. Furthermore, two approaches are considered to determine the number of α-cuts on which these algorithms need to be applied. In the first approach, a fixed number of

α-cuts is used, while in the second one, three α-cuts are chosen initially, and this number is extended by determining the required numbers of α-cuts self-consistently based on a linearity criterion. In addition, the Fuzzy Calculator can be applied in a non-parallel or a parallel fashion. The latter is only important when PSO is employed, in which case several swarms simultaneously search for the optima of different α-cuts and moreover communicate with each other in order to locate these optima more accurately and/or more rapidly.

Chapter 14 discusses the methodology used to develop the different Fuzzy Calculators. In Chapter 15, in order to evaluate and compare the different Fuzzy Calculators, they are applied to a number of test functions. The Fuzzy Calculator leading to the best results in Chapter 15, is then applied to a case study in Chapter 16. This case study deals with the propagation of uncertainty through a physically-based surface scattering model. Finally, Chapter 17 summarizes our conclusions.

14

Methodology

This chapter recapitulates the background of fuzzy calculus and goes on to describe the construction of our Fuzzy Calculator. Section 14.1 introduces the concept of fuzzy sets and provides some definitions for fuzzy quantities, which are used in further sections. Section 14.2 briefly describes the definition and properties of triangular norms, which are used to model interaction between variables. Section 14.3 recalls Zadeh's extension principle used to define the output of a continuous function of variables described by fuzzy intervals. In Sections 14.4 and 14.5 the functioning of the Fuzzy Calculator is outlined in full detail. Section 14.6 describes the different optimization algorithms used by the Fuzzy Calculator.

14.1. Fuzzy set theory

Fuzzy set theory is introduced to deal with incomplete or imprecise information. This theory was developed by Lotfi A. Zadeh [139] in 1965 as an extension of classical set theory. In contrast to classical set theory, in fuzzy set theory elements can partially belong to the set. This partial membership can be described by a membership degree defined by a membership function in the real unit interval $[0, 1]$ [140]. In this section, some definitions are introduced that are used in further sections.

A membership function represents the membership degree $A(x)$ of elements x in some universe U to a fuzzy set A. We identify the fuzzy set A with its membership function $A : U \mapsto [0, 1] : x \mapsto A(x)$. If an element is not included in the set A, then the membership degree $A(x)$ of that element is 0. An element is fully included in the set A if the membership degree $A(x)$ of that element is 1. Crisp or classical sets are obtained when all elements $x \in U$ have either $A(x) = 0$ or $A(x) = 1$. For fuzzy sets, general membership degrees $0 < A(x) < 1$ are also possible. We will mainly be interested in fuzzy sets defined for elements x on the real line, with thus $U = \mathbb{R}$. In this case, a fuzzy set A is more specifically called a fuzzy quantity A. The support of a fuzzy quantity A is the crisp set of all elements $x \in \mathbb{R}$ that have a nonzero membership degree:

$$\text{supp}(A) = \{x \in \mathbb{R} \mid A(x) > 0\}. \tag{14.1}$$

The core of a fuzzy quantity A is the crisp set of all elements $x \in \mathbb{R}$ that have a membership degree equal to one:

$$\text{core}(A) = \{x \in \mathbb{R} \mid A(x) = 1\}. \tag{14.2}$$

A fuzzy quantity A is called normal if there exists an $x \in \mathbb{R}$ such that $A(x) = 1$. The α-cut A_α of a fuzzy quantity A is the crisp set of elements $x \in \mathbb{R}$ that belong to A at least the degree $\alpha \in]0, 1]$

$$(A)_\alpha = \{x \in \mathbb{R} \mid A(x) \geq \alpha\}. \tag{14.3}$$

A fuzzy quantity A for which all α-cuts are closed and that has a compact support, *i.e.* $\overline{\text{supp}(A)} = \overline{\{x \in \mathbb{R} \mid A(x) > 0\}}$ is bounded, is called an upper semi-continuous fuzzy quantity A. Fuzzy quantities A are called convex if the α-cuts of A are connected, *i.e.* do not consist of several disconnected closed intervals:

$$(\forall (x_1, x_2) \in \mathbb{R}^2)(\forall \eta \in]0, 1])(A(\eta x_1 + (1 - \eta)x_2) \geq \min(A(x_1), A(x_2))). \tag{14.4}$$

A fuzzy number is a fuzzy quantity that is upper semi-continuous, convex and normal and has the membership degree $A(x) = 1$ at precisely one element. When this last condition is not fulfilled, the fuzzy quantity is commonly called a fuzzy interval.

At last, a distinction is made between non-interactive and interactive fuzzy quantities A_i. In a non-fuzzy setting where the quantities A_i represent intervals, non-interactivity means that none of the combinations $(x_1, \ldots, x_n) \in A_1 \times \cdots \times A_n$ is deemed impossible. The joint membership function of the n fuzzy quantities can then be represented by $\min(A_1(x_1), \ldots, A_n(x_n))$. Of course, this is not always the case, e.g. when some A_i and A_j are related to the same physical observable. One way to model interactivity is to replace the minimum operator in the joint membership function by another operator T that transforms the individual membership degrees $A_1(x_1)$ to $A_n(x_n)$ into a valid joint membership degree $T(A_1(x_1), \ldots, A_n(x_n))$. Such operators T are called triangular norms and have to satisfy a number of properties. Triangular norms are formally introduced in the next section. Note that there are other possibilities for constructing the joint membership function, e.g. by applying a possibilistic clustering algorithm to a joint measurement of the relevant variables, based on physical or artificially generated data [141]. In this chapter and in Chapter 15, we restrict ourselves to a construction based on triangular norms. In Chapter 16, the Fuzzy Calculator is applied to propagate uncertainty through a physically-based surface scattering model, for which the interactivity between the fuzzy input intervals can be described by a possibilistic clustering algorithm.

14.2. Triangular norms

The main idea of triangular norms or t-norms is to generalize the intersection of classical sets to fuzzy sets or the Boolean conjunction of classical logic to fuzzy logic. Using this prescription, a t-norm T allows to define the joint membership degree of fuzzy quantity A_1 taking the value x_1 *and* fuzzy quantity A_2 taking the value x_2 as $T(A_1(x_1), A_2(x_2))$. Hence, t-norms can be used to describe the interactivity between variables, by replacing the minimum-operator encountered in the previous section with a more general t-norm T. We now summarize the main properties that have to be satisfied by t-norms.

A t-norm is a binary operation T on the unit interval $[0,1]$, *i.e.* a function $T : [0,1]^2 \to [0,1]$ with the following properties for all $\alpha, \beta, \xi \in [0,1]$ [142, 143]:

$$\begin{array}{lll} \text{(T1)} & T(\alpha,\beta) = T(\beta,\alpha) & \text{(commutativity)} \\ \text{(T2)} & T(\alpha, T(\beta,\xi)) = T(T(\alpha,\beta),\xi) & \text{(associativity)} \\ \text{(T3)} & T(\alpha,\beta) \leq T(\alpha,\xi) \text{ whenever } \beta \leq \xi & \text{(monotonicity)} \\ \text{(T4)} & T(\alpha,1) = \alpha & \text{(boundary condition)} \end{array} \tag{14.5}$$

The combination of conditions (T3) and (T4) with the restriction $T(\alpha,\beta) \in [0,1]$ allows to further conclude that for all $\alpha, \beta \in [0,1]$:

$$T(\alpha,\beta) \leq \min(\alpha,\beta), \tag{14.6}$$

$$T(\alpha,0) = 0. \tag{14.7}$$

Since T is a binary operation that maps two elements from the set $[0,1]$ to an output in $[0,1]$ and T is associative, the structure $([0,1],T)$ is a semigroup. More strongly, it is a commutative semigroup with neutral element 1 and zero element 0. In addition, the monotonicity of T allows to conclude that $([0,1],T,\leq)$ is a fully ordered semigroup with respect to the standard ordering relation $\leq$ on the set $[0,1]$. These properties uniquely define a t-norm: any binary operation T on $[0,1]$ such that $([0,1],T,\leq)$ is a fully ordered commutative semigroup with neutral element 1 and zero element 0 is a t-norm.

Furthermore, the associativity allows us to extend each t-norm T to an n-ary operation. For each n-tuple $(\alpha_1,\ldots,\alpha_n) \in [0,1]^n$, the value $T(\alpha_1,\ldots,\alpha_n)$ is defined iteratively through

$$T(\alpha_1,\ldots,\alpha_n) = T_{i=1}^{n}\alpha_i = T(T_{i=1}^{n-1}\alpha_i, \alpha_n) \tag{14.8}$$

with the case $n = 2$ given by the definition of $T(\alpha_1,\alpha_2)$. For any $\alpha \in [0,1]$, we also

define

$$\alpha_T^{(0)} = 1, \qquad \alpha_T^{(1)} = \alpha, \qquad \alpha_T^{(n)} = T(\underbrace{\alpha, \ldots, \alpha}_{n \text{ times}}). \tag{14.9}$$

There exist four basic t-norms $T_{\mathbf{M}}$, $T_{\mathbf{P}}$, $T_{\mathbf{L}}$ and $T_{\mathbf{D}}$ that are given by

$$T_{\mathbf{M}}(\alpha, \beta) = \min(\beta, \alpha), \qquad \text{(minimum)} \tag{14.10}$$

$$T_{\mathbf{P}}(\alpha, \beta) = \alpha \cdot \beta, \qquad \text{(product)} \tag{14.11}$$

$$T_{\mathbf{L}}(\alpha, \beta) = \max(\alpha + \beta - 1, 0), \qquad \text{(Łukasiewicz t-norm)} \tag{14.12}$$

$$T_{\mathbf{D}}(\alpha, \beta) = \begin{cases} 0 & \text{if } (\alpha, \beta) \in [0, 1[^2, \\ \min(\alpha, \beta) & \text{otherwise} \end{cases} \qquad \text{(drastic product)} \tag{14.13}$$

If two t-norms T_A and T_B satisfy $T_A(\alpha, \beta) \leq T_B(\alpha, \beta)$, for all $(\alpha, \beta) \in]0, 1[^2$, then T_A is called *weaker* than T_B and T_B is *stronger* than T_A, which is denoted as $T_A \leq T_B$. Note that not every pair of t-norms can be ordered. It is, however, possible to define the absolute weakest and the absolute strongest t-norms. In particular, the minimum t-norm $T_{\mathbf{M}}$ is the largest or strongest t-norm. It is the only t-norm that satisfies $T(\alpha, \alpha) = \alpha$, for all $\alpha \in [0, 1]$. The drastic product $T_{\mathbf{D}}$ is the smallest or weakest t-norm; it is the only t-norm that has $T(\alpha, \alpha) = 0$, for all $\alpha \in [0, 1[$. Thus, any t-norm T satisfies $T_{\mathbf{D}} \leq T \leq T_{\mathbf{M}}$. The four basic t-norms can be ordered as

$$T_{\mathbf{D}} < T_{\mathbf{L}} < T_{\mathbf{P}} < T_{\mathbf{M}}. \tag{14.14}$$

Condition (T3) imposes monotonicity but does not require the t-norm to be strictly monotone. A t-norm T is said to be strictly monotone if $T(x, y) < T(x, z)$ whenever $x > 0$ and $y < z$. Equivalently, this t-norm satisfies the *cancellation law*:

$$T(\alpha, \beta) = T(\alpha, \xi) \qquad \Rightarrow \qquad \alpha = 0 \vee \beta = \xi \tag{14.15}$$

A t-norm can be continuous (e.g. $T_{\mathbf{M}}$ and $T_{\mathbf{P}}$) but is not required to be so (e.g. $T_{\mathbf{D}}$) . Because of (T1) and (T3), a t-norm T will be continuous if and only if it is continuous in its first component. A t-norm that is continuous and strictly monotone is called a strict t-norm (e.g. $T_{\mathbf{P}}$).

One further defines for a t-norm T:

- an *idempotent element* $\alpha \in [0, 1]$ if $T(\alpha, \alpha) = \alpha$. Every t-norm has the trivial idempotent elements $\alpha = 0$ and $\alpha = 1$.
- a *nilpotent element* $\alpha \in]0, 1[$ if there exists some $n \in \mathbb{N}_0$ such that $\alpha_T^{(n)} = 0$.
- a *zero divisor* $\alpha \in]0, 1[$ if there exists some $\beta \in]0, 1[$ such that $T(\alpha, \beta) = 0$.

If α is an idempotent element, $\alpha_T^{(n)} = \alpha$ and α can never be a nilpotent element.

Every nilpotent element is a zero divisor, but not conversely. $T_{\mathbf{M}}$ is the only t-norm for which every $\alpha \in [0,1]$ is an idempotent element; it has no zero divisors or nilpotent elements. $T_{\mathbf{L}}$ and $T_{\mathbf{D}}$ have the complete set $]0,1[$ as nilpotent elements. T_P has neither nilpotent elements nor non-trivial idempotent elements. For a general t-norm T, the set of nilpotent elements is always of the form $]0,\gamma[$ or $]0,\gamma]$. If T is continuous and has $]0,1[$ as its set of nilpotent elements, it is called a *nilpotent t-norm* (e.g. $T_{\mathbf{L}}$). Finally, one can introduce the concept of *Archimedean* t-norms. A t-norm T is Archimedean if for any $(\alpha,\beta) \in]0,1[^2$, there exists some $n \in \mathbb{N}_0$ such that $\alpha_T^{(n)} < \beta$. A t-norm is Archimedean if and only if it satisfies the *limit property*:

$$\forall \alpha \in]0,1[: \lim_{n\to\infty} \alpha_T^{(n)} = 0. \tag{14.16}$$

For continuous Archimedean t-norms, yet another equivalent definition can be given. A continuous t-norm is Archimedean if and only if for any $\alpha \in]0,1[$, the t-norm satisfies $T(\alpha,\alpha) < \alpha$. An Archimedean t-norm that is left-continuous is automatically continuous. Continuous Archimedean t-norms are either strict or nilpotent.

All relations between the different properties of t-norms are summarized in Figure 14.1.

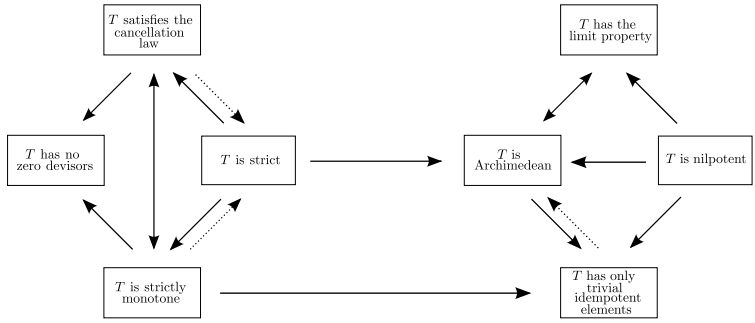


Figure 14.1: Implications between the algebraic properties of t-norms. The dotted arrow indicates that the corresponding implication holds for continuous t-norms [143].

Given the four basic t-norms, one can construct other t-norms. This requires the introduction of the concept of a pseudo-inverse. For a monotone function $f : [a,b] \to [c,d]$, where $[a,b]$ and $[c,d]$ are two closed subintervals of the extended real line $[-\infty,\infty]$, the pseudoinverse $f^{(-1)} : [c,d] \to [a,b]$ is defined as [142, 143]

$$f^{(-1)}(y) = \sup\{x \in [a,b] | (f(x) - y)(f(b) - f(a)) < 0\} \tag{14.17}$$

If f is a bijection from $[a,b]$ to $[c,d]$, then the definition of the pseudoinverse $f^{(-1)}$ corresponds to the inverse function f^{-1}. The definition of the pseudoinverse $f^{(-1)}$ is more general and can also be applied to functions f that are not injective. If

f is strictly increasing ($f(a) < f(b)$), for all $y \in [c,d]$, we get $f^{(-1)}(y) = \sup\{x \in [a,b] | f(x) < y\}$. On the other hand, if f is strictly decreasing ($f(a) > f(b)$), then for all $y \in [c,d]$, we obtain $f^{(-1)}(y) = \sup\{x \in [a,b] | f(x) > y\}$. Finally, if f is a constant function ($f(a) = f(b)$), then for all $y \in [c,d]$ we have $f^{(-1)}(y) = a$.

If the function $f(\alpha) = \alpha_T^{(n)}$ for a given t-norm T, then the pseudoinverse function $f^{(-1)}$ can be formally denoted as $f^{(-1)}(\alpha) = \alpha_T^{(1/n)}$, which thus provides a definition for fractional powers of α with respect to T. In particular, if T is a continuous Archimedean t-norm we obtain

$$\lim_{n\to\infty} \alpha_T^{(1/n)} = \begin{cases} 0, & \alpha = 0 \\ 1, & \alpha \in]0,1] \end{cases} \tag{14.18}$$

For every t-norm T, we can now define a whole family of related t-norms. When we have an increasing function $f : [0,1] \to [0,1]$ and a t-norm T such that

- for all $\alpha, \beta \in [0,1[$ we have

$$T(f(\alpha), f(\beta)) \in \mathrm{Ran}(f) \cup [0, f(0^+)]$$

  with $\mathrm{Ran}(f) = f([a,b]) = \{f(x), \forall x \in [a,b]\}$ and $f(0^+) = lim_{x \geq 0} f(x)$

- for all $(\alpha, \beta) \in [0,1]^2$ for which $T(f(\alpha), f(\beta)) \in \mathrm{Ran}(f)$ we have

$$f(f^{(-1)}(T(f(\alpha), f(\beta)))) = T(f(\alpha), f(\beta))$$

then we can define a new t-norm $T_{[f]} : [0,1]^2 \to [0,1]$ as

$$T_{[f]}(\alpha, \beta) = \begin{cases} f^{(-1)}(T(f(\alpha), f(\beta))) & \text{if} \quad (\alpha, \beta) \in [0,1[^2 \\ \min(\alpha, \beta) & \text{otherwise} \end{cases} \tag{14.19}$$

In particular, if f is a bijection (such that $f^{(-1)} = f^{-1}$), then it establishes an isomorphism between the semigroups $([0,1], T)$ and $([0,1], T_{[f]})$ since $T(f(\alpha), f(\beta)) = f(T_{[f]}(\alpha, \beta))$ for any $\alpha, \beta \in [0,1]$ by construction. For example, any nilpotent triangular norm T is isomorphic to $T_{\mathbf{L}}$, since there exists a bijection f such that $T_{[f]}$ is an isomorphism of $T_{\mathbf{L}}$.

We can also define new t-norms without any reference to a given t-norm, but rather by combining functions in one real variable with the elementary binary operations of addition and multiplication of real numbers. In the case of addition of real numbers, these functions are called additive generators, in the case of multiplication of real numbers we talk about multiplicative generators.

An additive generator $t : [0,1] \to [0, +\infty]$ of a t-norm T is a strictly decreasing function which is also right-continuous in 0 and satisfies $t(1) = 0$, such that for all

$(\alpha, \beta) \in [0,1]^2$ we have

$$t(\alpha) + t(\beta) \in \mathrm{Ran}(t) \cup [t(0), +\infty]\,, \tag{14.20}$$

$$T(\alpha, \beta) = t^{(-1)}(t(\alpha) + t(\beta))\,. \tag{14.21}$$

If a t-norm T has an additive generator $t : [0,1] \to [0, +\infty]$, then T is necessarily Archimedean. In addition, T is strictly monotone if and only if $t(0) = +\infty$. If $t(0) < +\infty$, then each $\alpha \in [0, 1[$ is a nilpotent element of T. The first condition on t strongly restricts possible choices of generators. In particular, t is continuous if and only if t is left-continuous in 1. A continuous generator t will produce a continuous Archimedean t-norm T.

A multiplicative generator of a t-norm T is a strictly increasing function $\theta : [0,1] \to [0,1]$ which is right-continuous in 0 and satisfies $\theta(1) = 1$, such that for all $(\alpha, \beta) \in [0,1]^2$ we have

$$\theta(\alpha) \cdot \theta(\beta) \in \mathrm{Ran}(\theta) \cup [0, \theta(0)]\,, \tag{14.22}$$

$$T(\alpha, \beta) = \theta^{(-1)}(\theta(\alpha) \cdot \theta(\beta))\,. \tag{14.23}$$

Clearly, an additive generator can be mapped to a multiplicative generator by setting $\theta(\alpha) = \exp(-t(\alpha))$, and similar properties hold. Since the product is a standard triangular norm $T_{\mathbf{P}}$, we can also interpret T as the triangular norm obtained through the construction in Eq. (14.19) applied to the triangular norm $T_{\mathbf{P}}$, *i.e.* $T = T_{\mathbf{P},[\theta]}$. When θ is a bijection, which requires that θ is continuous and $\theta(0) = 0$, the resulting t-norm T is thus isomorphic to $T_{\mathbf{P}}$. It can be shown that all t-norms T resulting from a bijection θ are strict continuous Archimedean t-norms.

14.3. Zadeh's extension principle: from theory to practice

Zadeh's extension principle [120] allows to extend any function f from a universe X to a universe Y, to a function from $\mathcal{F}(X)$ to $\mathcal{F}(Y)$ (where $\mathcal{F}(U)$ stands for the class of all fuzzy sets in a given universe U). More specifically, given a fuzzy set A in X, $f(A)$ is the fuzzy set in Y defined by

$$f(A)(y) = \sup_{f(x)=y} A(x) \tag{14.24}$$

(when no such x exists, the right-hand side evaluates to 0). Apart from its broad theoretical importance, the extension principle mainly finds application in the context of functions from $\mathbb{R}^n$ to $\mathbb{R}$. For a function f from $\mathbb{R}^n$ to $\mathbb{R}$ and n fuzzy

inputs $A_1, \ldots, A_n$, the fuzzy output $B = f(A_1, \ldots, A_n)$ is usually defined by

$$B(y) = \sup_{f(x_1,\ldots,x_n)=y} T(A_1(x_1), \ldots, A_n(x_n)) \,. \tag{14.25}$$

with T a t-norm that describes the interactivity between the n fuzzy inputs $A_1, \ldots, A_n$. $T(A_1(x_1), \ldots, A_n(x_n))$ thus represents the joint membership function of the n fuzzy inputs.

The supremum in Eq. (14.25) is taken over all $(x_1, \ldots, x_n)$ satisfying $f(x_1, \ldots, x_n) = y$. In general, the latter equation describes an $(n-1)$-dimensional manifold. For most functions, it is impossible to determine this manifold, let alone the supremum that the given function takes on it. Hence, there is a need for a more practical approach. Nguyen [121] offered a partial way out for a continuous function f and non-interactive upper semi-continuous fuzzy quantities A_i. Instead of determining $B(y)$ for each y separately, it is possible to determine the α-cuts of B, $\alpha \in]0, 1]$, directly as:

$$B_\alpha := f(A_1, \ldots, A_n)_\alpha = f((A_1)_\alpha, \ldots, (A_n)_\alpha) \tag{14.26}$$

where

$$f((A_1)_\alpha, \ldots, (A_n)_\alpha) = \{f(x_1, \ldots, x_n) \mid (x_1, \ldots, x_n) \in (A_1)_\alpha \times \ldots \times (A_n)_\alpha\} \,.$$

The support of B can be obtained in the same way from the supports of the fuzzy inputs A_i. Inverting the decomposition theorem for fuzzy sets, one can then reconstruct the fuzzy set B from its α-cuts B_α through

$$B(y) = \sup_{\alpha \in\,]0,1]} \alpha \cdot B_\alpha(y) \,, \tag{14.27}$$

where the α-cuts B_α are identified with their characteristic mapping. In this way, by determining a finite number of α-cuts B_α, and by restricting the supremum in Eq. (14.27) to the corresponding values of α, one can obtain an approximation of B (Figure 14.2):

$$B(y) = \max_{\alpha \in \{\alpha_1,\ldots,\alpha_k\}} \alpha \cdot B_\alpha(y) \,. \tag{14.28}$$

However, in order to determine the α-cuts in a feasible way, one additional restriction needs to be imposed: the fuzzy inputs $A_1, \ldots, A_n$ should be convex so that they represent fuzzy intervals. If all fuzzy inputs A_i are fuzzy intervals, then the fuzzy output B is a fuzzy interval as well. Hence, its α-cuts are closed intervals, *i.e.*

$$B_\alpha = [\underline{y}_\alpha, \overline{y}_\alpha] \,, \tag{14.29}$$

where $\underline{y}_\alpha$ and $\overline{y}_\alpha$ are the minimal and maximal values of the function f on the hyperrectangle

$$A_\alpha := [\underline{x}_{1,\alpha}, \overline{x}_{1,\alpha}] \times \cdots \times [\underline{x}_{n,\alpha}, \overline{x}_{n,\alpha}] \,, \tag{14.30}$$

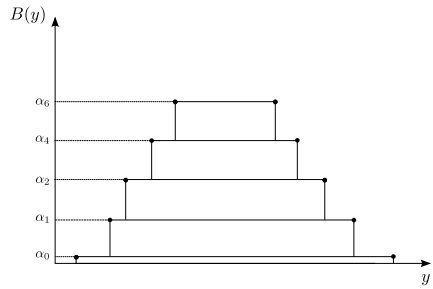


Figure 14.2: Approximation of B with a finite number of α-cuts.

where $(A_i)_\alpha = [\underline{x}_{i,\alpha}, \overline{x}_{i,\alpha}]$. The problem is thus reduced to finding $\underline{y}_\alpha$ and $\overline{y}_\alpha$ for a number of values of α. If the function f is monotone (increasing or decreasing in each variable) then only the vertices of the above hyperrectangle need to be scanned [144]. Moreover, if the function is increasing, then $\underline{y}_\alpha = f(\underline{x}_{1,\alpha}, \ldots, \underline{x}_{n,\alpha})$ and $\overline{y}_\alpha = f(\overline{x}_{1,\alpha}, \ldots, \overline{x}_{n,\alpha})$. The latter applies to the addition of fuzzy intervals, for instance.

For a general (non-monotone) continuous function f, the optima $\underline{y}_\alpha$ and $\overline{y}_\alpha$ can either be on the boundary or the interior of the corresponding search space, and optimization algorithms are needed to locate these optima efficiently [132, 133]. Moreover, the corresponding optimization problems are not independent, since $B_{\alpha_2} \subseteq B_{\alpha_1}$ if $\alpha_1 < \alpha_2$. The input values $\boldsymbol{x}_\alpha = (x_{1,\alpha}, \ldots, x_{n,\alpha})$ that result in the optima $\underline{y}_\alpha$ and $\overline{y}_\alpha$, are denoted as $\boldsymbol{x}_{\underline{y}_\alpha}$ and $\boldsymbol{x}_{\overline{y}_\alpha}$. In particular, we refer to $\boldsymbol{x}_{\underline{y}_\alpha}$ as the minimizer of the α-cut and to $\boldsymbol{x}_{\overline{y}_\alpha}$ as the maximizer of the α-cut. For each α-cut, depending on the function f, one or more local optima can be present in the search space. If we want to construct a general approach to solve these optimization problems, we cannot rely on an algorithm that tries to find a single nearby solution of the KKT conditions (Chapter 2, Section 2.2) such as SQP (Chapter 3, Section 3.1). A better strategy is to use an algorithm inspired by metaheuristics (Chapter 2, Section 2.4), such as PSO (Chapter 3, Section 3.2) or SIMPSA (Chapter 3, Section 3.3). In order to demonstrate this, results obtained with implementations of the Fuzzy Calculator based on these different optimization algorithms will be compared in Chapter 15.

In case of interactive fuzzy intervals A_i where the interactivity can be described by a t-norm, Fullér and Keresztfalvi (1991) [123] generalized Nguyen's theorem as follows

$$B_\alpha := f(A_1, \ldots, A_n)_\alpha = \bigcup_{T(\xi_1,\ldots,\xi_n) \geq \alpha} f((A_1)_{\xi_1}, \ldots, (A_n)_{\xi_n}) \tag{14.31}$$

where we have to take a union over all combinations $(\xi_1, \ldots, \xi_n) \in]0,1]^n$ that produce $T(\xi_1, \ldots, \xi_n) \geq \alpha$. But since $T(\xi_1, \ldots, \xi_n) \leq T_{\mathbf{M}}(\xi_1, \ldots, \xi_n) = \min(\xi_1, \ldots, \xi_n)$, we can restrict the search for possible combinations of $(\xi_1, \ldots, \xi_n)$ that produce

$T(\xi_1, \ldots, \xi_n) \geq \alpha$ to $[\alpha, 1]^n$. If we define the set A_α as

$$A_\alpha = \underset{T(\xi_1,\ldots,\xi_n)\geq\alpha}{\cup} (A_1)_{\xi_1} \times \cdots \times (A_n)_{\xi_n} \tag{14.32}$$

then we will have

$$A_\alpha \subset (A_1)_\alpha \times \cdots \times (A_n)_\alpha \tag{14.33}$$

where $(A_i)_\alpha = [\underline{x}_{i,\alpha}, \overline{x}_{i,\alpha}]$ is a closed interval. A_α will be a closed, simply connected set, *i.e.* a set for which any closed curve within the set can be continuously contracted to a point without leaving the set. Therefore, as in the case of non-interactive variables A_i, for a continuous function f, the α-cuts of the fuzzy output B are closed intervals. The fuzzy output B can thus be determined by determining $\underline{y}_\alpha$ and $\overline{y}_\alpha$ for a number of α-cuts as presented in Eqs. (14.29) and (14.30), with the additional constraint $T(A_1(x_1), \ldots, A_n(x_n)) \geq \alpha$. In this way, the identification of the fuzzy output of a continuous function of interactive variables described by fuzzy intervals can, just as in the case with non-interactive variables, be transformed to a number of (constrained) optimization problems, which now have one nonlinear constraint. Because of this constraint, the area where the function f has to be optimized, is no longer hyperrectangular. The search spaces for the optimization problems in two dimensions belonging to different α-cuts are presented in Figure 14.3 for the basic t-norms.

We conclude this section by noting that solving the constrained optimization problem is the most general strategy for dealing with interactivity described by t-norms. Next to this general approach, we could try to devise a specialised algorithm for one particular t-norm T. For any other t-norm T', we could then try to construct an isomorphism f such that $T'_{[f]} = T$. However, this is not always possible since different triangular norms can have different properties that are not changed by the isomorphism. For example, nilpotent t-norms will always be mapped to nilpotent t-norms. In particular, it will not be possible to map (in a reversible way) any t-norm T to the minimum norm $T_{\mathbf{M}}$, for which the optimization problem becomes most simple due to the hyperrectangular search spaces. These hyperrectangular search spaces can be easily handled by the optimization algorithm as boundary constraints. For discontinuous t-norms such as T_D, the search space is not generally convex, which means that it is not possible to connect any two points in the search space with a line segment that is completely located in this search space. In this type of search spaces, standard optimization algorithms might run into problems. For the particular case of T_D, the search space for any α-cut can however be divided into two (overlapping) hyperrectangles and the optimization problem can be solved in both hyperrectangles separately.

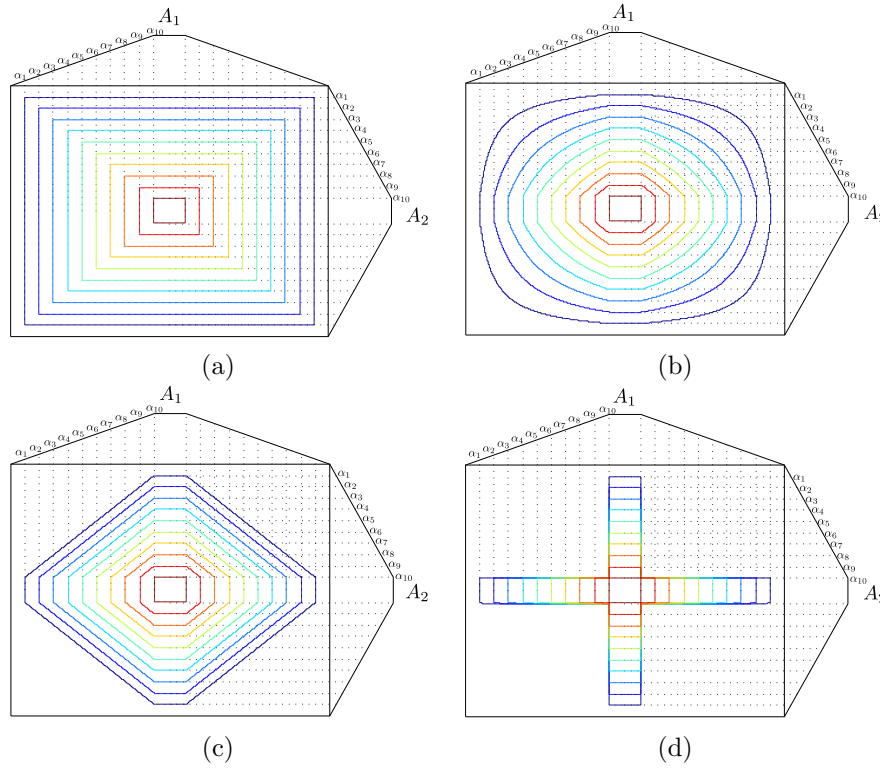


Figure 14.3: Search spaces for the optimization problems of a number of α-cuts for the basic t-norms in 2 dimensions: a) minimum t-norm, b) product t-norm, c) Łukasiewicz t-norm and d) drastic t-norm

14.4. Subdivision in α-cuts

As stated in the introduction, the objective of this part is to develop a Fuzzy Calculator to determine the membership function of the output of a continuous function of variables described by fuzzy intervals. Section 14.3 outlines how this problem can be handled through the application of the extension principle and a subdivision in α-cuts. Two sources of errors are connected with this approach. Firstly, for a finite number of α-cuts, only an approximation of the true membership function B is obtained. Increasing the number of α-cuts improves the approximation. Secondly, for each α-cut, the error on the determination of $\underline{y}_\alpha$ and $\overline{y}_\alpha$ depends on the optimization algorithm used. It is impossible to assess this error, unless the minimum and maximum of f for each α-cut can be exactly determined, e.g. through analytical techniques.

To determine the number of α-cuts, two approaches are followed. In the first approach, the number of α-cuts is set to a fixed number $m+1$, determined at the beginning of the algorithm. As the interval at $\alpha = 0$ is not closed, we choose the first α-cut at a small value $\alpha_0 = \delta > 0$, so that we can identify $\lim_{\delta \to 0} \alpha_0$ with $\overline{\text{supp}(A)}$. The remaining α-cuts are equidistantly distributed at values $\alpha_j = j/m$ for $j = 1, 2, \ldots, m$. In the second approach, we start with $m+1 = 3$ and gradually increase this number according to a criterion based on linear interpolation. More specifically, we compare α with a value $\tilde{\alpha}$ that is calculated by linear interpolation. In other words, for a given couple (α_j, y_{α_j}), with y_{α_j} either $\underline{y}_{\alpha_j}$ or $\overline{y}_{\alpha_j}$, an approximation $\tilde{\alpha}_j$ is calculated by linear interpolation through the points $(\alpha_{j-1}, y_{\alpha_{j-1}})$ and $(\alpha_{j+1}, y_{\alpha_{j+1}})$. If $|\alpha_j - \tilde{\alpha}_j| > \epsilon$, with ϵ a predefined tolerance value, y_α-values are searched at the new levels $\alpha = (\alpha_j + \alpha_{j-1})/2$ and $\alpha = (\alpha_j + \alpha_{j+1})/2$. This strategy is carried out for the left side (minima $\underline{y}_\alpha$) as well as for the right side (maxima $\overline{y}_\alpha$) independently. As a consequence, a higher density of points will be obtained where the membership function B is more complicated. Therefore, the number of optimizations needed to determine the left (# $\text{optimizations}_{\text{left}}$) or right (# $\text{optimizations}_{\text{right}}$) side of the membership function B respectively can be different. In this approach, by selecting a tolerance value ϵ with respect to the difference in α-values instead of the difference in y_α-values, an absolute tolerance value can be assured that is independent of the problem at hand. Figure 14.4 illustrates the linear interpolation for the left side of the membership function of the output variable of a function f.

The error made in determining $\underline{y}_\alpha$ or $\overline{y}_\alpha$, although unknown for a general function f, can be used to assess the different optimization algorithms. As better optimization algorithms result in lower $\underline{y}_\alpha$- and higher $\overline{y}_\alpha$-values, they will yield a wider membership function B. We can then use the area S under the membership function as a global quality measure. Using the approximation of B in Eq. (14.28), this area can be computed as follows:

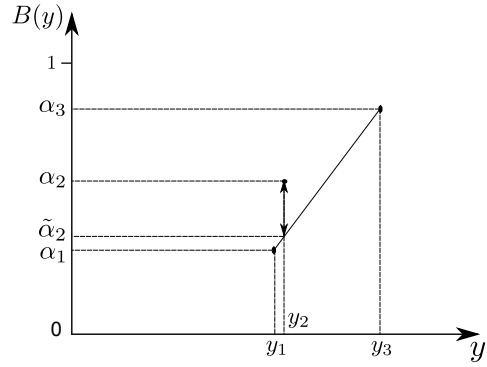


Figure 14.4: Linear interpolation of α_2 through α_1 and α_3

$$S_B = \alpha_0(\overline{y}_{\alpha_0} - \underline{y}_{\alpha_0}) + \sum_{i=1}^{m}(\alpha_i - \alpha_{i-1})(\overline{y}_{\alpha_i} - \underline{y}_{\alpha_i})\,. \tag{14.34}$$

In the limit $\alpha_0 = \delta \to 0$, the first term does not contribute, since $(\overline{y}_{\alpha_0} - \underline{y}_{\alpha_0})$ is expected to remain finite in order to have a compact support. As mentioned above, in the second α-cut approach it is possible that a part of one side of the membership function B is more complicated than the corresponding part of the other side of that membership function. In order to calculate S_B, the optima $\underline{y}_\alpha$ and $\overline{y}_\alpha$ of the α-cuts are used. When for a certain α-cut, there is only an optimum on one side of the membership function B and no corresponding optimum on the other side of the membership function, we work with the corresponding optimum of the nearest α'-cut with $\alpha' > \alpha$; this is presented by the red lines in Figure 14.5.

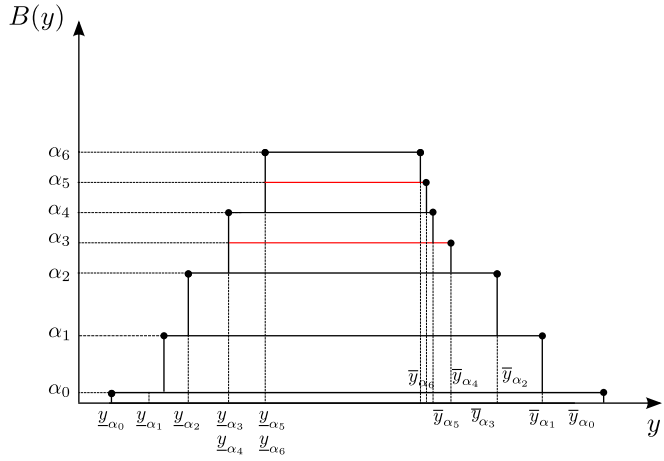


Figure 14.5: Rectangles used in order to calculate S_B. The red lines represent the case where for the α_3- and α_5-cut only an optimum on one side of the membership function B is present

Errors made by the optimization algorithms can result in inconsistencies. Given the definition of a membership function, it is impossible that $\underline{y}_{\alpha_{j-1}} > \underline{y}_{\alpha_j}$ or that $\overline{y}_{\alpha_{j-1}} < \overline{y}_{\alpha_j}$. Therefore the different α-cuts cannot be determined independently and we have to correct the optima of the α-cuts when this situation occurs. There are two ways to correct for these inconsistencies. If an inconsistency is discovered, e.g. because $\overline{y}_{\alpha_{j-1}} < \overline{y}_{\alpha_j}$, $\overline{y}_{\alpha_{j-1}}$ can be reset to $\overline{y}_{\alpha_j}$. Hence, we refer to this approach as the first correction approach. Another possibility is to discard the old result and recalculate $\overline{y}_{\alpha_{j-1}}$. If the optimization algorithm accepts a starting point, then providing the maximizer $\boldsymbol{x}_{\overline{y}_{\alpha_j}}$ will ensure that the inconsistency is solved. In case of an increasing number of α-cuts it is not necessary to recalculate the inconsistent α-cuts, since the linear interpolation automatically increases the number of α-cuts if necessary. Therefore, in case of an increasing number of α-cuts, we choose to remove the inconsistent α-cuts without recalculation of these α-cuts. Further on, we refer to this last approach as the second correction approach, which consists of recalculating the inconsistent optima in case of a fixed number α-cuts and of removing the inconsistent optima in case of an increasing number of α-cuts .

14.5. Non-Parallel versus Parallel methodology

The Fuzzy Calculator is implemented in the programming environment Octave [138] and accepts a general n-ary function f. Both a non-parallel and a parallel version were designed. As in Part II, for the parallelization of the Fuzzy Calculator, the Message Passing Interface (MPI) of Octave was used. Here too, we have chosen for a master-slave setup (Chapter 6, Section 6.2.5). In the next chapter, the non-parallel and parallel implementations will be compared at the level of accuracy, *i.e.* area under the membership function (see Eq. (14.34)) of the output of the function, and the number of function evaluations.

Non-parallel This version of the algorithm uses only the first α-cut approach. In this approach the number of α-cuts is fixed to $m+1$ with levels $\alpha_0 = \delta$ and $\alpha_j = j/m$ for $j = 1, 2, \ldots, m$. The non-parallel version of the algorithm first starts with searching the optima (the minimum and maximum) of the function f at level $\alpha_m = 1$. As the 1-cut is the smallest interval, chances are higher to find the correct minimum $\underline{y}_{\alpha_m}$ and maximum $\overline{y}_{\alpha_m}$. The algorithm then continues with the determination of the optima of the other α-cuts, for decreasing values of α. When allowed by the optimization algorithm, the minimizer $\boldsymbol{x}_{\underline{y}_{\alpha_{j+1}}}$ and the maximizer $\boldsymbol{x}_{\overline{y}_{\alpha_{j+1}}}$ of the preceding α_{j+1}-cut can be provided as starting point. In this way, the inconsistency mentioned in the last paragraph of Section 14.4 cannot occur.

Parallel This version of the algorithm uses both approaches for the determination of the number of α-cuts. The parallel version of the implementation is based on a master-slave configuration. For the first α-cut approach, *i.e.* a fixed number of $m+1$ α-cuts is employed, we used $2m+3$ processes: one master and $2(m+1)$ slaves for the determination of the left and right optima $\underline{y}_{\alpha_j}$ and $\overline{y}_{\alpha_j}$, for $j = 0, 1, 2, \ldots, m$. The master sends the hyperrectangles corresponding to the $m+1$ α-cuts to the slaves that optimize the function on these hyperrectangles and return these optima to the master. When the master receives the optima of all α-cuts, these values are compared to correct for inconsistencies. In this part, the first correction approach is applied. This parallel implementation is described in Algorithm 10. Clearly, by restricting the number of α-cuts to a fixed value, no attempt is made to accurately represent the fuzzy output. Rather, by comparing the parallel with the non-parallel implementation in terms of number of function evaluations, the advantage of having different optimization problems running together versus the advantage of having the previous optimizer can be evaluated. We choose $2m+1$ processes in order to maximize the advantage of having all optimization problems running simultaneously.

Algorithm 10: Parallel Fuzzy Calculator with a fixed number of α-cuts

```
Data: fuzzy intervals A_1, ..., A_n, m+1 α-levels, function f
Result: membership function B of the output
Initialize MPI;
if master process then
    while α-levels to be processed & slaves processing do
        Send A_α to empty slave processes;
        Receive optima y_α and ȳ_α;
        Update process status;
    Correct for inconsistencies if necessary;
    Send quitmessage;
else
    Timetoquit=false;
    while timetoquit is false do
        Receive messages from master;
        if message is A_α then
            Search for optima y_α and ȳ_α of the α-cut;
            Return result;
        else if message=quitmessage then
            Timetoquit=true;
```

In the second α-cut approach, we start from 3 α-cuts and increase this number through linear interpolation if necessary. This algorithm starts with searching for the optima of the α-cuts at levels $1, 0.5$ and δ. The obtained values are compared

and corrected if inconsistencies occur. Next, a linear interpolation is applied to examine whether the optima $\underline{y}_\alpha$ or $\overline{y}_\alpha$ of the intermediate α-cuts at the levels $\alpha = (\alpha_j + \alpha_{j+1})/2$ and $\alpha = (\alpha_j + \alpha_{j-1})/2$ need to be calculated. When the optimization algorithm accepts more than one starting point, the optimization algorithm is fed with the optimizers $\boldsymbol{x}_{y_{\alpha_j}}$ and $\boldsymbol{x}_{y_{\alpha_{j+1}}}$ respectively, if these optimizers are situated in the current search space. However, when the optimization algorithm only accepts a single starting point, the optimizer $\boldsymbol{x}_{y_{\alpha_{j+1}}}$ is provided. After the optima of the new α-cuts have been obtained, the loop is repeated. The results are checked for possible inconsistencies and corrected. In this part, both the first and the second correction approach is applied. Linear interpolation is again used to examine whether additional α-cuts are required. The algorithm stops when the convergence criterion of the linear interpolation is fulfilled for all α-cuts.

When the parallel Fuzzy Calculator is employed with the second α-cut approach, an arbitrary (fixed in advance) number of slave processes can be used. The master process will remember which slaves are processing, and will queue all new requests for the calculation of the optima of new α-cuts. When free slaves are available, they will be requested to search for the minimum or maximum for the next hyperrectangle A_α corresponding to the α-cut in the queue. The pseudo-code for the parallel implementation starting with 3 α-cuts can be found in Algorithm 11.

Algorithm 11: Parallel Fuzzy Calculator starting with 3 α-cuts

Data: fuzzy intervals $A_1, \ldots, A_n$, tolerance ϵ for linear interpolation, 3 α-levels, function f
Result: membership function B of the output
Initialize MPI;
if *master process* **then**
  while *α-levels to be processed & slaves processing* **do**
    Send A_α to empty slave processes;
    Receive minima $\underline{y}_\alpha$ and $\overline{y}_\alpha$;
    Update process status;
    while *candidate minima $\underline{y}_{\alpha_j}$ for linear interpolation* **do**
      Correct for inconsistencies if necessary;
      if $|\alpha_j - \tilde{\alpha}_j| > \epsilon$ **then**
        Send $A_{(\alpha_j+\alpha_{j-1})/2}$ with $\boldsymbol{x}_{\underline{y}_{\alpha_{j-1}}}$ and/or $\boldsymbol{x}_{\underline{y}_{\alpha_j}}$;
        Send $A_{(\alpha_j+\alpha_{j+1})/2}$ with $\boldsymbol{x}_{\underline{y}_{\alpha_j}}$ and/or $\boldsymbol{x}_{\underline{y}_{\alpha_{j+1}}}$;
        Receive minima $\underline{y}_{(\alpha_j+\alpha_{j-1})/2}$ and $\underline{y}_{(\alpha_j+\alpha_{j+1})/2}$;
        Update process status;
        Update array of candidate minima for linear interpolation;
    while *candidate maxima $\overline{y}_{\alpha_j}$ for linear interpolation* **do**
      Correct for inconsistencies if necessary;
      if $|\alpha_j - \tilde{\alpha}_j| > \epsilon$ **then**
        Send $A_{(\alpha_j+\alpha_{j-1})/2}$ with $\boldsymbol{x}_{\overline{y}_{\alpha_{j-1}}}$ and/or $\boldsymbol{x}_{\overline{y}_{\alpha_j}}$;
        Send $A_{(\alpha_j+\alpha_{j+1})/2}$ with $\boldsymbol{x}_{\overline{y}_{\alpha_j}}$ and/or $\boldsymbol{x}_{\overline{y}_{\alpha_{j+1}}}$;
        Receive maxima $\overline{y}_{(\alpha_j+\alpha_{j-1})/2}$ and $\overline{y}_{(\alpha_j+\alpha_{j+1})/2}$;
        Update process status;
        Update array of candidate maxima for linear interpolation;
  Send quitmessage;
else
  Timetoquit is false;
  while *timetoquit is false* **do**
    Receive messages from master;
    if *message is A_α* **then**
      Search for optima $\underline{y}_\alpha$ or $\overline{y}_\alpha$ of the α-cut;
      Return result;
    else if *message=quitmessage* **then**
      Timetoquit=true;

14.6. Delineation of the optimization algorithms

As outlined in Section 14.3, the identification of the fuzzy output of a continuous function of non-interactive variables described by fuzzy intervals can be converted into a number of optimization problems, namely determining the minima and maxima of the different α-cuts of the membership function of the output. Several optimization algorithms are employed and compared, namely a local optimization algorithm, Gradient Descent based on Sequential Quadratic Programming (GD), and two global optimization algorithms, the Simplex-Simulated Annealing (SIMPSA) algorithm and the Particle Swarm Optimization (PSO) algorithm. We now discuss the specifics of these methods for the problem at hand, in particular with respect to the possible advantage that can be obtained by having different dependent optimization processes running simultaneously in a parallel implementation.

14.6.1. Gradient Descent based on Sequential Quadratic Programming

Gradient Descent based on Sequential Quadratic Programming (GD), a local optimization algorithm, is described in Chapter 3, Section 3.1. This algorithm is a standard function of Octave, namely `sqp` [138]. This function has no extra parameters that have to be set and can thus be applied directly to the optimization problem at hand. This algorithm cannot be modified in order to benefit from information of other `sqp` problems running simultaneously.

14.6.2. Simplex-Simulated Annealing

The Simplex-Simulated Annealing (SIMPSA) algorithm [34] is an optimization algorithm based on a combination of the nonlinear Simplex algorithm [35] and the Simulated Annealing algorithm [36]. This algorithm is described in full detail in Chapter 3, Section 3.3. The implementation of SIMPSA was taken from [23] and [24, 25] with the author's permission. As mentioned in Chapter 3, Section 3.3, the maximal number of iterations, the cooling ratio, the freezing temperature and the tolerance of the convergence criterion have to be determined. After some test simulations we decided to set the maximal number of iterations for each simulated annealing step to 2500, the cooling ratio to 0.7 and the freezing temperature to 1. The tolerance level for convergence is set to 10^{-6}. Since this is also an optimization algorithm that is not population based, there is no obvious strategy to let it benefit from information about other SIMPSA processes.

14.6.3. Particle Swarm Optimization

Particle Swarm Optimization (PSO) [28], a population-based global optimization algorithm, is described in Chapter 3 (Section 3.2). The implementation of PSO was taken from Part III and has been appropriately modified. This algorithm requires the determination of a population size N, a cognitive parameter c_1, a social parameter c_2, an inertia weight w and a convergence criterion. As we want to evaluate the performance of PSO without exhaustive search for the ideal parameters, we just performed some test simulations, and we have decided to work with fixed parameter values $c_1 = 1$, $c_2 = 1.5$ and $w = 0.7$, while different population sizes of $N = 10$, $N = 15$ and $N = 20$ are used. The convergence criterion used requires that half of the population has the same position (with tolerance level 10^{-6}). Explicitly, the algorithm ends if the mean Euclidean distance between the particles of the best half of the population is smaller than 10^{-6}.

As mentioned in Chapter 3 (Section 3.2), each particle is attracted to its personal best position as well as to the global best position of the total population. The parallel Fuzzy Calculator can thus be interpreted as several swarms that search for different α-cuts at the same time. As the search space for $\alpha > \alpha'$ is part of the search space for α', any two swarms have part of the search space in common. Therefore, it would be beneficial if these swarms could communicate about candidate solutions. We adapted the PSO algorithm such that each swarm broadcasts its current global best position to the other running PSO processes. When a swarm receives a global best position that is located in its search space and that is better than its own global best position, the swarm changes its global best position. It is important to note that, whenever a new global best position is obtained through communication, the remaining particles are reinitialized at random because the current swarm might already be close to converging to a local optimum (far) away from the newly obtained optimum. It would hence last very long before all particles shift their position and converge, especially if the new optimum is close in value to the previous one. To account for this communication, a new parameter is introduced, namely the frequency of communication. We have varied this parameter by running instances in which the swarms communicate at every 2, 5 or 10 iterations of the PSO algorithm.

14.6.4. Combination of Particle Swarm Optimization and Gradient Descent

As it is not certain that PSO will converge to a local/global optimum [3], it may be recommended to combine PSO with a local optimization algorithm such as Gradient Descent (PSO_GD). Many ways exist to incorporate Gradient Descent in the PSO algorithm. In order to have a final best solution that is assured to be a local optimum, we first performed Gradient Descent on each of the initial

particles. All the particles will then be positioned in a local optimum, from which we only retain the particle with the best position in the population. The other particles are repositioned at their original position received during the initialization of PSO. Next, each time the swarm changes its global best position through communication, GD is performed on the particle with the new global best position. After convergence of the swarm, we perform GD one last time on the global best position of the swarm.

15

Experimental results

This chapter discusses the results obtained with different versions of the Fuzzy Calculator developed in Chapter 14. The goal of this chapter is to select the best configuration of the Fuzzy Calculator by studying the effect of varying the optimization algorithm and parameters thereof, the strategy for choosing the number of α-cuts and the correction approach. The different configurations are applied to a number of abstract test functions through which we would like to propagate fuzzy input variables as accurately as possible. The best configuration of the Fuzzy Calculator is then selected by performing a statistical analysis over these different functions, that are defined in different dimensions and have a different number of local minima and maxima. This configuration will then be applied to a practical case study in the next chapter. Section 15.1 describes the different test functions used to evaluate the Fuzzy Calculator. Section 15.2 presents and compares the results obtained with the Fuzzy Calculator using the optimization algorithms described in Chapter 14 (Section 14.6) for the different test functions of non-interactive variables described by fuzzy intervals. In Section 15.3, the application of the Fuzzy Calculator to test functions of interactive variables is discussed. In this chapter, we restrict ourselves to interactivity described by t-norms.

15.1. Test functions

In order to compare the performance of the different optimization algorithms in the Fuzzy Calculator, we have used some continuous functions restricted to certain domains. We use the one-dimensional cosine function on the interval $[0, 4\pi]$ as a first test function (test function 1) (Figure 15.1(a)) and a second two-dimensional test function with multiple minima and maxima on $[0, 4]^2$ (test function 2) (Figure 15.1(b)). Besides these simple functions we have also applied our algorithms to the alpine function in 2 dimensions on $[1, 5]^2$ (test function 3), $[-5, 0]^2$ (test function 4), $[-5, 0] \times [10, 15]$ (test function 5) and $[-5, 5] \times [25, 30]$ (test function 6) (Figure 15.1(c), (d), (e), (f)), to the alpine function in 3 dimensions on $[-5, 0]^3$ (test function 7), to the alpine function in 4 dimensions on $[-5, 0]^4$ (test function 8) and to the alpine function in 5 dimensions on $[-5, 0]^5$ (test function

9). The alpine function, which is often used as benchmark function for continuous optimization algorithms [3], is given by

$$f(x_1, ..., x_D) = \sum_{i=1}^{D} |x_i \sin(x_i) + 0.1x_i| \tag{15.1}$$

The fuzzy intervals describing non-interactive input variables are chosen to have a trapezoidal shape. The interval $[\underline{x}_{j,\delta}, \overline{x}_{j,\delta}]$ at $\alpha = \delta$ corresponds to the interval to which the test functions are restricted. At $\alpha = 1$, this interval is reduced to $[\underline{x}_{j,1}, \overline{x}_{j,1}]$ with

$$\underline{x}_{j,1} = \frac{\underline{x}_{j,\delta} + \overline{x}_{j,\delta}}{2} - \frac{1}{10}\left(\overline{x}_{j,\delta} - \underline{x}_{j,\delta}\right), \quad \overline{x}_{j,1} = \frac{\underline{x}_{j,\delta} + \overline{x}_{j,\delta}}{2} + \frac{1}{10}\left(\overline{x}_{j,\delta} - \underline{x}_{j,\delta}\right).$$

In general, our implementation can deal with any fuzzy interval, not necessarily of a trapezoidal shape. We only require an explicit method to construct $\overline{x}_{i,\alpha}$ and $\underline{x}_{i,\alpha}$ for any α.

15.2. Non-interactive input variables

The membership functions of the outputs are constructed with the Fuzzy Calculator using the aforementioned optimization algorithms. In a first approach, these membership functions are constructed for a fixed number of α-cuts. In a second approach, we start with 3 α-cuts and increase this number using a criterion based on linear interpolation. The non-parallel and parallel Fuzzy Calculator are compared on the level of accuracy and the number of function evaluations. To allow for a statistical comparison between the Fuzzy Calculators using different optimization algorithms or between the non-parallel and parallel implementation, each algorithm is repeated 50 times. As we are interested in knowing which of the optimization algorithms is most suited to construct the membership function of the output of an arbitrary continuous function of non-interactive variables described by fuzzy intervals, the choice of test functions should be considered as a random factor. Since we have 50 repetitions per function for each algorithm, *i.e.* a total of 450 observations per algorithm, we can rely on the central limit theorem for normality. Therefore, a mixed ANOVA model (ANalysis Of VAriance model) with random effects and a Satterthwaite correction for unequal variances with a confidence level of 95 % can be used to compare the algorithms [78]. A comparison is made between the area under the membership function and between the number of function evaluations needed to construct the membership function. In addition, the computational efficiency is also taken into account.

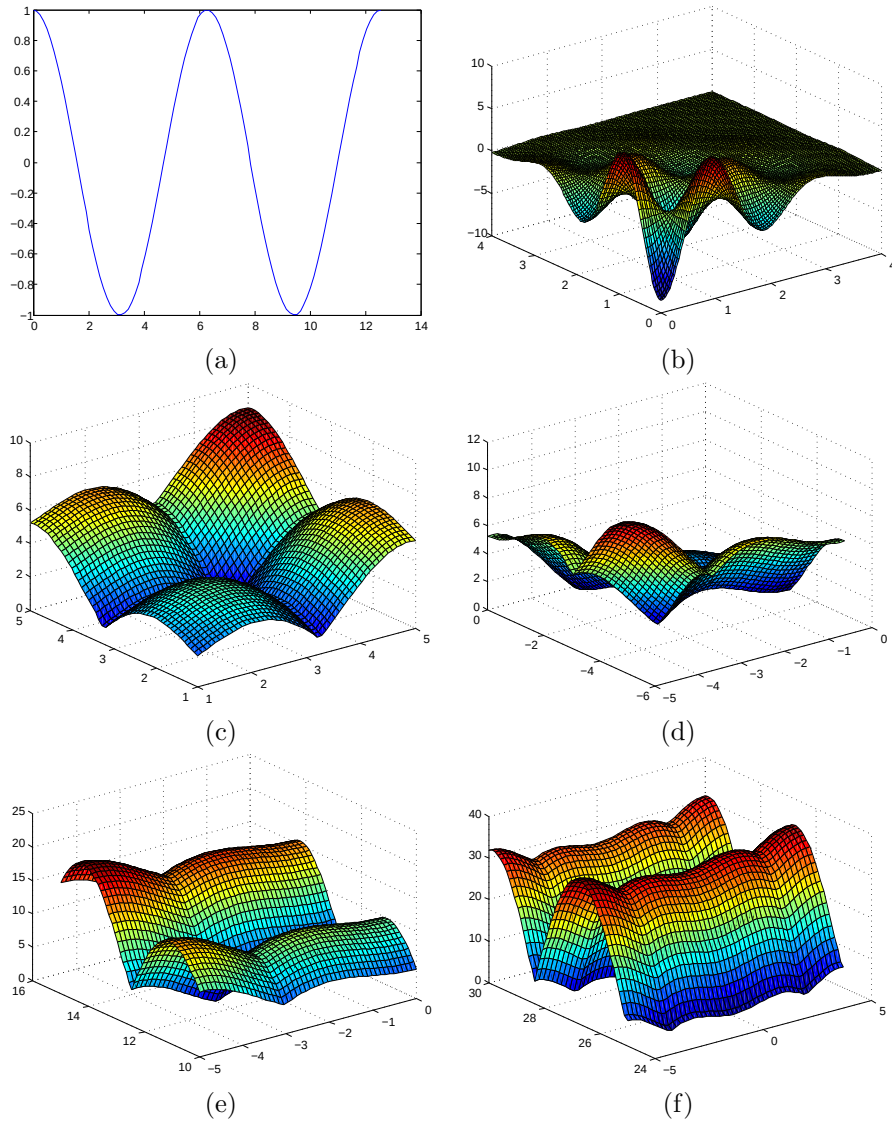


Figure 15.1: First test function on $[0, 4\pi]$ (test function 1)(a), second test function on $[0, 4]^2$ (test function 2)(b), the Alpine test function in 2 dimensions on $[1, 5]^2$ (test function 3)(c), $[-5, 0]^2$ (test function 4) (d), $[-5, 0] \times [10, 15]$ (test function 5) (e) and $[-5, 5] \times [25, 30]$ (test function 6) (f)

15.2.1. Fixed number of α-cuts

In this section, $m + 1 = 11$ α-cuts are used. Firstly, the performance of the different optimization algorithms is compared for the non-parallel Fuzzy Calculator. Secondly, the parallel Fuzzy Calculator is used, and the differences in the results with respect to the non-parallel Fuzzy Calculator are examined.

Non-parallel Fuzzy Calculator

The capability to construct the membership functions of the outputs for the 9 test functions is examined for the optimization algorithms GD, SIMPSA, PSO with a population of 10, 15 or 20 particles, and PSO_GD with a population of 10, 15 or 20 particles. In order to test the performance of the optimization algorithms, a mixed ANOVA model with the different test functions as random effect is applied to the data of the areas under the membership functions composed by the Fuzzy Calculator using these optimization algorithms. Using this test, we can see whether significant differences in performance between the optimization algorithms are present without taking into account each test function separately. As GD is not a stochastic algorithm and always leads to the same result for a fixed initial position, it is not possible to put the results of this algorithm in the mixed ANOVA model. Therefore, we will only compare the mean areas under the membership functions constructed by the Fuzzy Calculator using GD with that of the membership functions composed by the Fuzzy Calculator using the other algorithms.

Table 15.1 contains the mean areas under the membership functions for the outputs of the different test functions composed by the Fuzzy Calculator using the different optimization algorithms. This table illustrates that the application of GD results in a mean area that is much lower than the mean areas under the membership functions constructed by the Fuzzy Calculator using the other optimization algorithms. This was expected as GD is a local optimization algorithm and can only find local optima. Therefore, this algorithm is henceforth no longer used.

Table 15.2 presents the significance of the differences between the mean areas of the membership functions composed by the Fuzzy Calculator using the different optimization algorithms. When the p-value is smaller than 0.05 the difference is significant, which implies that the algorithm with the higher mean area is significantly better than the algorithm resulting in the smaller mean area. In Table 15.2, this is indicated by the symbols $<$, $>$ and $\approx$, whereby $>$ ($<$) denotes a significantly higher (lower) mean area for the left algorithm with respect to the algorithm above and $\approx$ is used in the absence of significant differences. Table 15.1 shows that the application of the method PSO_GD with a population size of 20 particles leads to the highest mean area. Table 15.2 illustrates that this result is significantly better than the results obtained with PSO. This result is, however, not

Table 15.1: Mean areas under the membership functions over the different test functions obtained with the Fuzzy Calculator using the different optimization algorithms.

Algorithm	Pop size	Mean area
GD		8.593
SIMPSA		11.323
PSO	10	10.886
	15	11.035
	20	11.152
PSO_GD	10	11.289
	15	11.323
	20	11.334

Table 15.2: Significance of the differences between the mean areas under the membership functions composed by the Fuzzy Calculator using the different optimization algorithms.

Method		PSO			PSO_GD			SIMPSA
	Pop size	10	15	20	10	15	20	
PSO	10		$<$	$<$	$<$	$<$	$<$	$<$
PSO	15	$>$		$<$	$<$	$<$	$<$	$<$
PSO	20	$>$	$>$		$<$	$<$	$<$	$<$
PSO_GD	10	$>$	$>$	$>$		$\approx$	$\approx$	$\approx$
PSO_GD	15	$>$	$>$	$>$	$\approx$		$\approx$	$\approx$
PSO_GD	20	$>$	$>$	$>$	$\approx$	$\approx$		$\approx$
SIMPSA		$>$	$>$	$>$	$\approx$	$\approx$	$\approx$	

significantly different from the results obtained with the Fuzzy Calculator using PSO_GD with a population size of 10 and 15 particles and SIMPSA.

The number of function evaluations needed to construct the membership functions of the outputs is a measure for the computational cost. The number of function evaluations of the Fuzzy Calculator using PSO_GD with a population size of 10, 15 and 20 particles and SIMPSA are compared.

Table 15.3 presents the mean numbers of function evaluations and the mean CPU times in unit of seconds for these three algorithms. The mean numbers of function evaluations is significantly different (p-value < 0.05). As the Fuzzy Calculator using SIMPSA has a very high mean number of function evaluations and needs much more time to construct the membership functions of the outputs, we can conclude that this algorithm is computationally inefficient. Therefore, the SIMPSA algorithm is not used in the parallel Fuzzy Calculator.

Parallel Fuzzy Calculator

Section 3.2 (Chapter 14) described how the parallel Fuzzy Calculator, using PSO and PSO_GD as optimization algorithm, can be interpreted as several swarms

Table 15.3: Mean numbers of function evaluations and CPU times (s) over the different test functions for the Fuzzy Calculator using PSO_GD with a population size of 10, 15 and 20 particles and SIMPSA.

Algorithm	Pop size	Mean number of function evaluations	Mean CPU time (s)
PSO_GD	10	19359	16.532
PSO_GD	15	29108	25.041
PSO_GD	20	38347	32.869
SIMPSA		71791	50.150

searching for the optima of different α-cuts at the same time. Furthermore, communication between the different swarms about candidate solutions is enabled. Except for the use of multiple processor units, this form of communication is the most important difference between the non-parallel and the parallel Fuzzy Calculator. In combination with the results of the non-parallel Fuzzy Calculator, this remark justifies the restriction of the parallel Fuzzy Calculator to the swarm-based optimization algorithms PSO and PSO_GD.

As we are interested in the difference between these two optimization algorithms for all test functions, a mixed ANOVA model with the test functions as random effect is performed. Since the factors population and communication are available in our two optimization algorithms, we can use them as nested factors in this model, which leads to a more correct estimate of the p-values.

Table 15.4 presents the mean areas under the membership functions obtained with the Fuzzy Calculator using PSO and PSO_GD, for different populations sizes and communication strategies. As described in Section 15.2.1, the population size varies between 10, 15 and 20 particles. We use 4 different communication strategies, namely communication at every 2, 5 or 10 iterations, or no communication at all. In all possible combinations of population size and communication strategy, PSO_GD is significantly better than PSO, when averaged over the different population sizes and communication strategies. Tables 15.5 and 15.6 show the significance of the differences between the mean area obtained with the Fuzzy Calculator using PSO and PSO_GD, between the different population sizes and the different communication strategies. As in Table 15.2, these differences are indicated by the symbols $<$, $>$ or $\approx$. Table 15.5 illustrates that PSO_GD with a population size of 20 particles is significantly better than PSO and PSO_GD with the other population sizes, when averaged over the different communication strategies. Table 15.6 shows that PSO_GD using communication at every 5 iterations is significantly better than PSO and PSO_GD using the other communication strategies, when averaged over the different population sizes. Important to note is that the interaction between the nested factors population and communication is significant. This means it is not possible to state that a population of 20 particles is significantly better

Table 15.4: Mean areas under the membership functions over the different test functions obtained with the parallel Fuzzy Calculator using PSO and PSO_GD, for a population size of 10, 15 or 20 particles, with communication between the swarms at the different α-cuts at every 2, 5 or 10 iterations or without communication.

Algorithm	Pop size	Communication	Mean area
PSO	10, 15, 20	at every 2 it, 5 it, 10 it, no	11.060
PSO_GD	10, 15, 20	at every 2 it, 5 it, 10 it, no	11.278
PSO	10	at every 2 it, 5 it, 10 it, no	10.96
PSO	15	at every 2 it, 5 it, 10 it, no	11.074
PSO	20	at every 2 it, 5 it, 10 it, no	11.150
PSO_GD	10	at every 2 it, 5 it, 10 it, no	11.206
PSO_GD	15	at every 2 it, 5 it, 10 it, no	11.289
PSO_GD	20	at every 2 it, 5 it, 10 it, no	11.325
PSO	10, 15, 20	at every 2 it	11.048
PSO	10, 15, 20	at every 5 it	11.139
PSO	10, 15, 20	at every 10 it	11.075
PSO	10, 15, 20	no	10.984
PSO_GD	10, 15, 20	at every 2 it	11.261
PSO_GD	10, 15, 20	at every 5 it	11.299
PSO_GD	10, 15, 20	at every 10 it	11.271
PSO_GD	10, 15, 20	no	11.255

for all communication strategies and that communication at every 5 iterations is significantly better for all population sizes. We nevertheless assume that PSO_GD with a population size of 20 particles and communication at every 5 iterations is the best algorithm to construct the membership functions of the outputs and will restrict to this configuration in the remainder of this chapter. This assumption is supported by the observation that the mean area under the membership function for this configuration is larger than for all other possible configurations of the communication strategy and the population size. It is clear why a communication frequency that is too low is unfavourable, but an understanding of the unfavourable effect of too frequent communication is less straightforward. A possible explanation is that whenever a swarm discovers a newly interesting region (either through communication or by own means), there should be enough time for a thorough exploitation of this region. If the communication frequency is too high, the swarm will constantly be drawn away from his own discoveries in the possibly interesting region.

The mixed ANOVA model, with the test functions as random factor and the population size and communication strategies as nested factors, is also performed to distinguish between the optimization algorithms with respect to the number of function evaluations. Table 15.7 contains the mean numbers of function evaluations

Table 15.5: Significance of the differences between the mean areas under the membership functions composed by the parallel Fuzzy Calculator using PSO and PSO_GD, for a population size of 10, 15 or 20 particles, over all communication strategies.

Algorithm		PSO			PSO_GD		
	Population size	10	15	20	10	15	20
PSO	10		$<$	$<$	$<$	$<$	$<$
PSO	15	$>$		$<$	$<$	$<$	$<$
PSO	20	$>$	$>$		$<$	$<$	$<$
PSO_GD	10	$>$	$>$	$>$		$<$	$<$
PSO_GD	15	$>$	$>$	$>$	$>$		$<$
PSO_GD	20	$>$	$>$	$>$	$>$	$>$	

Table 15.6: Significance of the differences between the mean areas under the membership functions composed by the parallel Fuzzy Calculator using PSO and PSO_GD, with communication between the swarms at the different α-cuts at every 2, 5 or 10 iterations or without communication, over all population sizes.

Algorithm		PSO				PSO_GD			
	Communication	2 it	5 it	10 it	no	2 it	5 it	10 it	no
PSO	2 it		$<$	$<$	$>$	$<$	$<$	$<$	$<$
PSO	5 it	$>$		$>$	$>$	$<$	$<$	$<$	$<$
PSO	10 it	$>$	$<$		$>$	$<$	$<$	$<$	$<$
PSO	no	$<$	$<$	$<$		$<$	$<$	$<$	$<$
PSO_GD	2 it	$>$	$>$	$>$	$>$		$<$	$\approx$	$\approx$
PSO_GD	5 it	$>$	$>$	$>$	$>$	$>$		$>$	$>$
PSO_GD	10 it	$>$	$>$	$>$	$>$	$\approx$	$<$		$\approx$
PSO_GD	no	$>$	$>$	$>$	$>$	$\approx$	$<$	$\approx$	

for the Fuzzy Calculator using the different optimization algorithms, population sizes and communication strategies. As the CPU time needed to construct the membership functions of the outputs is roughly proportional to the number of function evaluations and is dependent on the number of processors used, time is no longer a criterion in the decision on which Fuzzy Calculator performs best. The difference in the number of function evaluations between the Fuzzy Calculator using PSO and PSO_GD is on average not significant (p-value > 0.05). However, the effect of the population size and the communication strategy on the number of function evaluations is on average significant (p-value < 0.05). As Table 15.7 reveals, a larger population size needs significantly more function evaluations, when averaged over the different communication strategies. Important to note is, when averaged over the different population sizes, that in case of no communication, the number of function evaluations when using PSO_GD is, although not significantly, higher than the number of function evaluations when using PSO. Contrastingly, in case of communication the number of function evaluations when using PSO_GD is smaller than when using PSO. When communication is present, the different swarms communicate their global best position and replace this position when a better one is found. After this communication, the other particles of the swarm are reinitialized. If one swarm has discovered a new minimum, it will take a number of iterations before it has precisely located the exact position of this minimum. During

Table 15.7: Mean numbers of function evaluations over the different test functions obtained with the parallel Fuzzy Calculator using PSO and PSO_GD, for a population size of 10, 15 or 20 particles, with communication between the different swarms at every 2, 5 or 10 iterations or without communication.

Method	Pop size	Communication	Mean number of function evaluations
PSO	10, 15, 20	at every 2 it, 5 it, 10 it, no	30286
PSO_GD	10, 15, 20	at every 2 it, 5 it, 10 it, no	30860
PSO	10	at every 2 it, 5 it, 10 it, no	20638
PSO	15	at every 2 it, 5 it, 10 it, no	29879
PSO	20	at every 2 it, 5 it, 10 it, no	40342
PSO_GD	10	at every 2 it, 5 it, 10 it, no	21930
PSO_GD	15	at every 2 it, 5 it, 10 it, no	27757
PSO_GD	20	at every 2 it, 5 it, 10 it, no	42892
PSO	10, 15, 20	at every 2 it	34339
PSO	10, 15, 20	at every 5 it	32886
PSO	10, 15, 20	at every 10 it	33363
PSO	10, 15, 20	no	20557
PSO_GD	10, 15, 20	at every 2 it	30938
PSO_GD	10, 15, 20	at every 5 it	32318
PSO_GD	10, 15, 20	at every 10 it	31646
PSO_GD	10, 15, 20	no	28537

these iterations, a slowly decreasing objective function value is communicated to the other swarms at every 2, 5 or 10 iterations causing them to constantly reinitialize. In constrast, if a single gradient descent is applied to immediately determine the precise location of the minimum, only a single communication step is required.

In a last step, results obtained with the non-parallel and the parallel Fuzzy Calculator are compared. Using the mixed ANOVA model with the test functions as random effect, we compared the results of the non-parallel Fuzzy Calculator using PSO_GD with a population size of 10, 15 or 20 particles and the results of the parallel Fuzzy Calculator using PSO_GD with a population size of 20 particles with communication at every 5 iterations. The difference between the mean areas under the membership functions obtained with the different Fuzzy Calculators is significantly different (p-value < 0.05). Table 15.8 illustrates that the parallel Fuzzy Calculator using PSO_GD with a population of 20 particles and with communication at every 5 iterations leads to a significantly more accurate membership function of the output.

However, the number of function evaluations of the parallel Fuzzy Calculator using PSO_GD with a population size of 20 particles and with communication at every 5 iterations is significantly higher than the non-parallel Fuzzy Calculator using

Table 15.8: Mean areas under the membership functions and mean numbers of function evaluations over the different test functions obtained with the non-parallel Fuzzy Calculator using PSO_GD, for a population size of 10, 15 or 20 particles and the parallel Fuzzy Calculator using PSO_GD for a population size of 20 particles with communication between the different swarms at every 5 iterations.

Algorithm	Pop size	Mean area	Mean number of function evaluations
Non-parallel PSO_GD	10	11.289	19359
Non-parallel PSO_GD	15	11.324	29108
Non-parallel PSO_GD	20	11.334	38347
Parallel PSO_GD comm every 5 it	20	11.357	45127

PSO_GD with a population size of 10, 15 or 20 particles (Table 15.8).

15.2.2. Increasing number of α-cuts

A disadvantage of working with a fixed number of α-cuts is that this number needs to be determined in advance. Consequently, it is possible that too many α-cuts are used to determine the membership function of the output for a simple function, whereas too few α-cuts are used for more complex functions. On that account, a solution can be to start with 3 α-cuts and increase this number according to a linearity criterion. To this end, as explained in Section 14.5 (Chapter 14), we decided to set a tolerance of 0.01 between α and an approximation $\tilde{\alpha}$ obtained by linear interpolation. In order to compare the influence of the two α-cut approaches, the parallel Fuzzy Calculator using PSO_GD, with a population size of 20 particles, with communication at every 5 iterations is applied, with the fixed as well as the increasing number of α-cuts, to construct the membership functions of the outputs for all test functions.

Using the mixed ANOVA model with as random effect the different test functions, we compared the Fuzzy Calculator with a fixed number of α-cuts and the Fuzzy Calculator with an increasing number of α-cuts. Table 15.9 presents the mean areas under the membership functions and the mean numbers of function evaluations needed by these Fuzzy Calculators. The differences between the mean areas and the mean numbers of function evaluations are significant (p-value < 0.05), which leads to the conclusion that the Fuzzy Calculator with the increasing number of α-cuts gives a more accurate membership function than the Fuzzy Calculator with the fixed number of α-cuts. However, significantly more function evaluations are needed when the number of α-cuts is not fixed (Table 15.9). The probable explanation is that for most test functions, the number of optimized α-cuts is a lot higher than 11 when starting with 3 α-cuts and increasing this number according

Table 15.9: Mean areas under the membership functions and mean numbers of function evaluations over the different test functions obtained with the parallel Fuzzy Calculator using PSO_GD for a population size of 20 particles with communication between the different swarms at every 5 iterations, for a fixed number of α-cuts and increasing number of α-cuts.

Algorithm	Mean area	Mean number of function evaluations
Fixed number of 11 α-cuts	11.357	45127
Increasing number of α-cuts	11.664	85534

Table 15.10: Mean numbers of optimizations needed to construct the left (# $\text{optimization}_{\text{left}}$) and right (# $\text{optimization}_{\text{right}}$) membership function of the output over the different test functions with the Fuzzy Calculator starting with 3-α-cuts.

Test function	# $\text{optimizations}_{\text{left}}$	# $\text{optimizations}_{\text{right}}$
1	18.0	3.0
2	3.4	33.9
3	19.8	34.6
4	14.7	18.0
5	9.5	35.6
6	21.3	35.2
7	25.0	48.8
8	26.0	45.1
9	25.9	58.0

to a linearity criterion. This indicates that 11 α-cuts are not enough to accurately represent the output of the test functions. This premise is confirmed by the results in Table 15.10.

As mentioned in Section 14.5 (Chapter 14), there are two approaches to correct for inconsistencies between the optima of the α-cuts. In the previous results, the optima of all α-cuts were compared and replaced if necessary. Another approach is to recalculate the optima of the inconsistent α-cuts, with as starting point the optimizer $\boldsymbol{x}_{y_{\alpha'}}$ of the nearest α'-cut with $\alpha' > \alpha$. However, in the case of an increasing number of α-cuts, we can just remove the inconsistent α-cuts and create new α-cuts making use of the linearity criterion. Therefore, we used the latter approach in combination with the Fuzzy Calculator with an increasing number of α-cuts. We then compared these results, through the mixed ANOVA model, with those of the Fuzzy Calculator with an increasing number of α-cuts, with the first approach to correct for inconsistencies between the optima of the α-cuts. The difference between the mean areas under the membership functions is very small and not significant (p-value > 0.05) (Table 15.11).

Table 15.11: Mean areas under the membership functions and mean numbers of function evaluations over the different test functions obtained with the parallel Fuzzy Calculator with an increasing number of α-cuts using PSO_GD for a population size of 20 particles with communication between the different swarms at every 5 iterations, for the first and second correction approach for inconsistencies (see Section 14.4 of Chapter 14).

Algorithm	Mean area	Mean number of function evaluations
First correction approach	11.664	85534
Second correction approach	11.653	63357

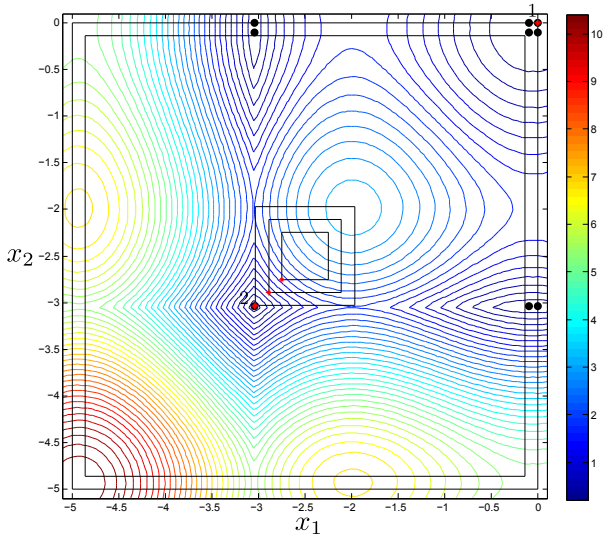
Table 15.12: Mean numbers of optimizations needed to construct the left (# $\text{optimizations}_{\text{left}}$) and right (# $\text{optimizations}_{\text{right}}$) membership function of the output over the different test functions with the Fuzzy Calculator starting with an increasing number of α-cuts with removing incorrectly found optima.

Test function	# $\text{optimizations}_{\text{left}}$	# $\text{optimizations}_{\text{right}}$
1	18.0	3.0
2	3.0	32.3
3	16.4	33.1
4	11.8	18.0
5	7.0	34.1
6	21.3	34.8
7	15.2	39.5
8	14.0	38.0
9	13.4	45.2

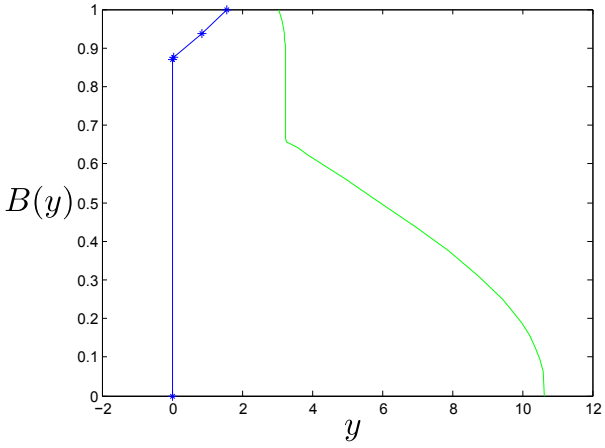
The difference between the mean numbers of function evaluations, however, is significant (p-value < 0.05). The second approach for dealing with inconsistent optima of α-cuts needs significantly less function evaluations. This is confirmed by comparing Table 15.10 and Table 15.12, which show that in general less α-cuts are needed when the second approach for dealing with inconsistencies between the optima of the α-cuts is employed.

15.2.3. Illustrative example of the Fuzzy Calculator

In order to elucidate the functioning of the Fuzzy Calculator, we illustrate in this section the results of the application of the Fuzzy Calculator with an increasing number of α-cuts, with removing incorrectly found optima (second correction approach) and using PSO_GD with a population size of 20 particles as optimization algorithm and with communication at every 5 iterations, to one of the test functions, namely the two-dimensional alpine function in the domain $[-5, 0]^2$ (Figure 15.1 (d), Section 15.1).



(a)



(b)

Figure 15.2: (a) Contour plot of the two-dimensional alpine function in the domain $[-5, 0]^2$, with the optimized α-cuts needed to construct the left membership function of the fuzzy output, (b) membership function of the fuzzy output of the the two-dimensional alpine function in the domain $[-5, 0]^2$, the left membership function corresponds to the blue part of this figure

Figure 15.2 (a) represents the contour plot of this test function, with indication of the optimized α-cuts that were generated for the construction of the left side of the membership function for the fuzzy output of this test function. Due to its special structure, the alpine function has different local minima that all have the same value $f = 0$ and are thus global minima. They are indicated with black solid circles. The optimum selected by the different α-cuts are indicated with red asterisks. For the α-cut at $\alpha = 0$, the global minimum at the point $(0, 0)$ (red asterisk number 1) is selected. This point is no longer in the feasible space for the next α-cut, but a different equivalent minimum around the point $(-3, -3)$ (red asterisk number 2) is selected instead. This minimum remains available in the feasible space all the way up to $\alpha \approx 0.88$. These results are in accordance with the left side of the membership function of the output of this test function in Figure 15.2 (b), as indicated by the vertical jump at function output value $z = 0$. As the value of α is now further increased above $\alpha \approx 0.88$, all global minima have disappeared from the feasible search space and the best optimum is located at the boundary of the search space along the upward slope of the objective function. This corresponds to the steady increase in the left side of the membership function in Figure 15.2 (b). The asterisks in Figure 15.2 (b) correspond to the optimized α-cuts.

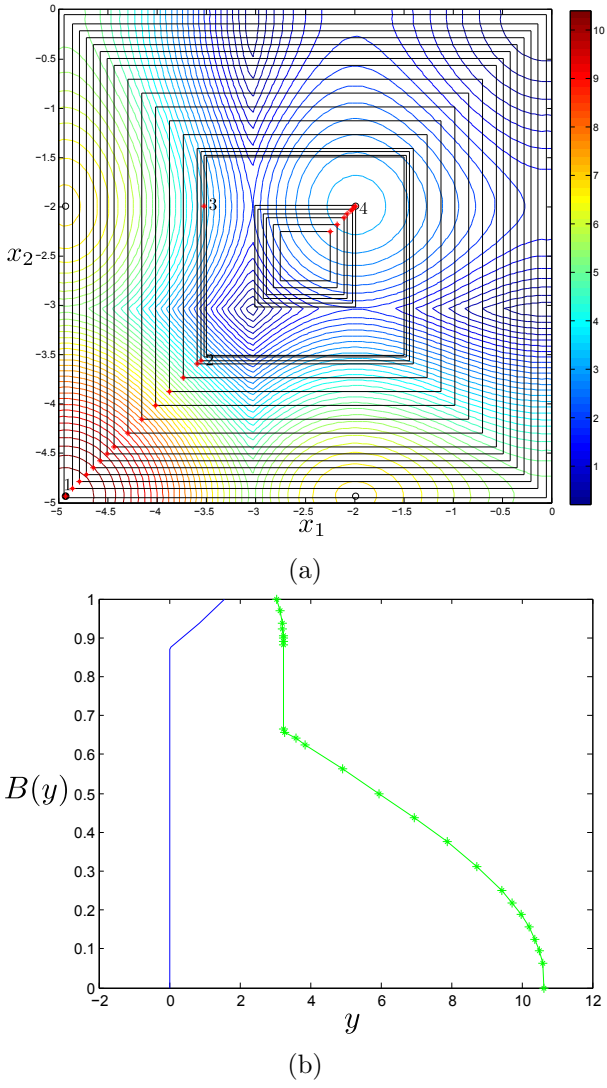


Figure 15.3: (a) Contour plot of the two-dimensional alpine function in the domain $[-5, 0]^2$, with the optimized α-cuts needed to construct the right membership function of the fuzzy output, (b) membership function of the fuzzy output of the two-dimensional alpine function in the domain $[-5, 0]^2$, the right membership function corresponds to the green part of this figure.

Figure 15.3 (a) represents the contour plot of the two-dimensional alpine test function in the domain $[-5, 0]^2$, with indication of the α-cuts that were generated for the construction of the right side of the membership function for the fuzzy output of this test function. There is now one global optimum at $(-4.93, -4.93)$ as indicated by the black solid square, and several local optima indicated by black open squares. Optima selected for the construction of the membership function are again indicated by red asterisks. For the α-cut at value $\alpha = 0$, the global optimum is in the search space and can be selected (red asterisk number 1). There is a small vertical jump in the corresponding right side of the membership function displayed in Figure 15.3 (b), up to the value $\alpha = 0.023$ where the global optimum is at the boundary of the search space. As α is further increased, the global optimum is no longer accessible and the optimum for the different α-cuts is located at the boundary of the search space along the downward slope of the objective function. The contour plot illustrates that the output of the function is continuously decreasing up to the point $(-3.6, -3.6)$ (red asterisk number 2), corresponding to $\alpha \approx 0.66$. At this point, the value of the function at this downward slope becomes smaller than the local maximum present at the point $(-2, -2)$. In fact, an intermediate α-cut is created which makes a side jump to a different optimum at $(-3.5, -2.0)$ (red asterisk number 3), before the next α-cut selects the local maximum at $(-2, -2)$ (red asterisk number 4). This local maximum now remains available as α is further increased, up to the value $\alpha \approx 0.89$, corresponding to another vertical jump in the right side of the membership function in Figure 15.3 (b). As α is further increased, this local maximum also becomes inaccessible and new optima at the boundary of the search space along the downward slope of the objective function are created. As in Figure 15.2 (b), the asterisks in Figure 15.3 (b) indicate the optimized α-cuts.

15.3. Interactive input variables

In this section, we apply the Fuzzy Calculator, with an increasing number of α-cuts, with removing incorrectly found optima (second correction approach), using PSO_GD with a population size of 20 particles and with communication at every 5 iterations as optimization algorithm, to the test functions (described in Section 15.1) for interactive input variables described by fuzzy intervals. We restrict ourselves to interactivity described by t-norms. Section 15.3.1 presents the modifications of the Fuzzy Calculator in order to deal with interactive input variables. Section 15.3.2 illustrates the results obtained with the Fuzzy Calculator in case of interactive input variables.

15.3.1. Modifications to the Fuzzy Calculator

As mentioned in Section 14.3, when the input variables A_i are interactive and this interactivity can be described by a t-norm, we can still practically implement the extension principle through the adapted α-cut approach of Nguyen. The α-cuts of the fuzzy output can then be determined by optimizing $\overline{y}_\alpha$ and $\underline{y}_\alpha$ for the corresponding α-cut, which is modelled through the additional constraint $T(A_1(x), \ldots, A_n(x)) \geq \alpha$. Therefore, the Fuzzy Calculator as described in Chapter 14 (Section 14.5) can be directly applied to the case of interactive input variables. However, the optimization algorithm PSO_GD used to find the optima $\underline{y}_\alpha$ and $\overline{y}_\alpha$ of the α-cuts in Algorithm 11 (Chapter 14, Section 14.5) requires some modifications in order to deal with the additional constraint. The search spaces for the optimization algorithm corresponding to the different t-norms are presented for a number of α-cuts in Figure 14.3 (Chapter 14, Section 14.3). All feasible solutions for PSO_GD are thus located within these spaces, depending on the α-cut.

The basic PSO algorithm (Chapter 3, Algorithm 1) is modified in the following manner. Firstly, an initial population of particles is randomly chosen in the hyperrectangular search space (the search space in case of non-interactive variables). For each of these particles i it is checked whether $T(A_1(x_{i,1}), \ldots, A_n(x_{i,n})) \geq \alpha$; if this is not the case then the particle is reinitialized. As explained in Section 14.5 (Chapter 14), during the optimization of the α-cut at the level $\alpha = (\alpha_j + \alpha_{j+1})/2$, PSO is fed with the optimizers $\boldsymbol{x}_{y_{\alpha_j}}$ and $\boldsymbol{x}_{y_{\alpha_{j+1}}}$, if these are located in the search space of the α-cut at level α, *i.e.* if the t-norm at these optimizers is greater than or equal to α. Secondly, the GD part of the PSO_GD algorithm is performed. The *sqp* function of Octave is able to deal with nonlinear constraints, thus the constraint $T(A_1(x_{i,1}), \ldots, A_n(x_{i,n})) \geq \alpha$ can be added as an argument to this function. Thirdly, the velocities and positions of the particles are updated. If after the position update the particle is located outside the search space of the current α-cut the position update is canceled. In order to move this particle all the way to the boundary where the t-norm becomes smaller than the α-value of the current α-cut, we have implemented a simple search region method. This method calculates how far the particle can move in the direction of velocity $\boldsymbol{v}$ without violating the constraint and positions the particle at the boundary. Fourthly, the personal best position of each particle and global best position of the total population is updated. At every 5 iterations, communication takes place with the other swarms that search for different α-cuts at the same time. Again, the global best position is only changed if a better position located in the current search space is discovered. On the particle with the new global position, GD is performed. These steps are repeated until convergence takes place. After convergence, GD, taking into account the nonlinear constraint, is performed on the global best position of the swarm. It is important to note that in case of discontinuous t-norms, e.g. the drastic t-norm, GD can no longer be used. In that case, only PSO is applied.

As mentioned in Section 14.3 (Chapter 14), in case of the drastic t-norm, the search space for any α-cut can be divided into a number of (depending on the dimensionality of the problem) (overlapping) hyperrectangles and the optimization problem can be solved in these hyperrectangles separately. The results of the optima of the different hyperrectangles are then compared and the minimum and maximum values are selected as the optima of the current α-cut. In the next section, in case of interactivity described by the drastic t-norm, we compare this approach to calculate the output of the test functions (drastic_2) with the approach of adding the additional constraint $T_{\mathbf{D}}(A_1(x), \ldots, A_n(x)) >= \alpha$ (drastic_1).

15.3.2. Results

For the results, we restricted ourselves to the four basic t-norms, namely the minimum t-norm, the product t-norm, the Łukasiewicz t-norm and the drastic t-norm. The results presented in the tables below are obtained by repeating the Fuzzy Calculator 50 times and taking the mean.

Table 15.13 illustrates the mean areas under the membership functions of the different test functions in case of interactive input variables described by the four basic t-norms. Drastic_1 indicates the results obtained with the adapted α-cut approach of Nguyen, drastic_2 on the other hand indicates the results obtained through dividing the search space into a number of (overlapping) hyperrectangles. As illustrated in Figure 14.3 (Chapter 14, Section 14.3), the search space corresponding to interactivity described by the drastic t-norm is smaller than and contained in the search space corresponding to the other t-norms for the same value of α. Interactivity described by the minimum t-norm results in the largest search space. The search space when interactivity is described by the Łukasiewicz t-norm is smaller than the search space when interactivity is described by the product and the minimum t-norm:

$$A_{\alpha,\text{drastic}} \subset A_{\alpha,\text{Lukasiewicz}} \subset A_{\alpha,\text{product}} \subset A_{\alpha,\text{minimum}} \,. \tag{15.2}$$

These differences in search space indicate the following order for the areas S under the membership functions of the outputs of the different test functions

$$S_{\text{drastic}} \leq S_{\text{Lukasiewicz}} \leq S_{\text{product}} \leq S_{\text{minimum}} \,, \tag{15.3}$$

with S_{drastic} the area under the membership function in case of interactivity described by the drastic t-norm, $S_{\text{Lukasiewicz}}$ the area under the membership function in case of interactivity described by the Łukasiewicz t-norm, and so on.

Table 15.13 shows that the relation in Eq. (15.3) is satisfied for all test functions, with the exception of test function 5. For this test function 5, the area under

Table 15.13: Mean areas under the membership functions for the different test functions in case of interactive input variables described by fuzzy intervals obtained with the parallel Fuzzy Calculator with an increasing number of α-cuts using PSO_GD for a population size of 20 particles with communication between the different swarms at every 5 iterations. The interactivity of the input variables is described by t-norms. Drastic$_1$ indicates that the interactivity of the input variables is described by the drastic t-norm and this is implemented through the adapted α-cut approach. Drastic$_2$ on the other hand refers to the application of the drastic t-norm but in this case the search space for every α-cut is divided into a number of (overlapping) hyperrectangles and the optimization problem is solved for each of these hyperrectangles separately.

	t-norm				
Test function	drastic$_1$	drastic$_2$	Łukasiewicz	product	minimum
1	1.6124	1.6124	1.6124	1.6124	1.6124
2	4.3573	4.3660	4.4142	4.7022	5.8843
3	3.4556	3.7514	4.3289	4.6340	5.9987
4	17.5392	17.5116	17.4962	17.8353	18.0874
5	31.4611	31.4693	31.4665	31.4720	31.4753
6	3.4758	3.4640	4.1564	4.9749	6.2153
7	4.5163	4.5025	5.5576	6.0207	9.0193
8	5.1520	5.2536	6.5128	6.9785	11.4731
9	6.0030	6.0047	7.5587	8.4237	14.8793

the membership function obtained with the drastic t-norm is larger than the area obtained with the Łukasiewicz t-norm. This implies that in case of interactivity described by the Łukasiewicz t-norm, the Fuzzy Calculator was not able to localize the global optima for certain α-cuts. For test function 1, the area under the membership function is the same for interactivity described by the different t-norms. This illustrates that the optima of this test function for the different α-cuts are located within the search space resulting from interactivity described by the drastic t-norm. The results obtained with the different implementations of the drastic t-norm indicate that, in general, drastic$_2$ results in slightly larger areas under the membership functions. However, this difference is negligible.

Table 15.14 shows the mean numbers of function evaluations to construct the membership functions of the different test functions in case of interactive input variables described by the four basic t-norms. The number of function evaluations is, for most test functions, of the same order of magnitude for the different t-norms. Nevertheless, the difference in implementation of the interactivity described by the drastic t-norm (drastic$_1$ versus drastic$_2$) has an influence on the number of function evaluations for all test functions. For test functions 3, 5 and 6, less function evaluations are needed and for the other test function a lot more evaluations are needed in the case of the drastic$_1$ implementation. In case of the implementation drastic$_2$, more runs of the optimization algorithm are needed since every optimization is

carried out over multiple search spaces. Therefore we would expect a higher number of function evaluations, in accordance with our observations for functions 3, 5 and 6. However, when using implementation drastic$_1$, it is possible that the discontinuous constraint, leading to a strongly non-convex search space, can lead to difficulties with converging and finding the correct optimum, resulting in a larger number of iterations and thus of function evaluations. Figure 15.4 illustrates the possible difficulties associated with a non-convex search space. The particle x is located in one 'arm' of the search space, while the personal best position is located in the middle of the search space and the global best position is located in another arm of the search space. This results in a movement of the particle as indicated by the black arrow with open arrow point. The particle tries to move outside the search space, which is not possible and will therefore be positioned on the boundary. When the global and personal best position are not replaced, the particle will remain on this boundary of the search space. If this takes place for a number of particles, the particles will take a long time to converge. This could explain the higher number of function evaluations for test functions 1, 2, 4, 7, 8 and 9.

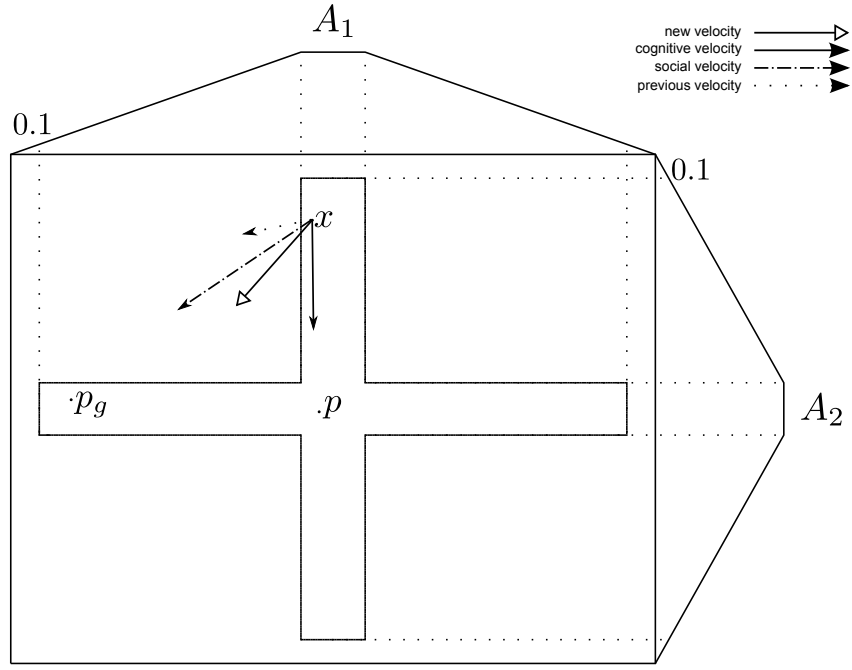


Figure 15.4: Application of PSO in the search space of the drastic t-norm

Table 15.14: Mean numbers of function evaluations for the different test functions in case of interactive input variables described by fuzzy intervals obtained with the parallel Fuzzy Calculator with an increasing number of α-cuts using PSO_GD for a population size of 20 particles with communication between the different swarms at every 5 iterations. The interactivity of the input variables is described by t-norms. Drastic$_1$ indicates that the interactivity of the input variables is described by the drastic t-norm and this is implemented through the adapted α-cut approach. Drastic$_2$ on the other hand refers to the application of the drastic t-norm but in this case the search space for every α-cut is divided into a number of (overlapping) hyperrectangles and the optimization problem is solved for each of these hyperrectangles separately.

	t-norm				
Test function	drastic$_1$	drastic$_2$	Łukasiewicz	product	minimum
1	23520	12236	12728	12474	9801
2	61180	23472	36872	23802	31177
3	52260	58516	74705	61877	71759
4	91580	53392	72520	69979	62407
5	70800	89744	89913	87215	91505
6	49240	59974	77830	87034	87866
7	181920	59331	109047	108491	86387
8	186220	54528	120191	101658	91385
9	361920	57375	123744	168333	111692

Tables 15.15 and 15.16 present the mean number of optimizations needed to construct the left and right part of the membership functions of the outputs for the different test functions when interactivity of the input variables is described by the 4 basic t-norms. More or less the same number of optimizations is necessary to construct the membership functions when interactivity is described by the Łukasiewicz, product and minimum t-norms. For interactivity described by the drastic t-norm, for some test functions the results deviate, especially for implementation drastic$_1$. This is of course also related to the non-convex search space. If the global optimum is not correctly determined, many additional α-cuts are required to satisfy the linearity criterion. This also contributed to the strongly increased number of function evaluations for the drastic$_1$ implementation applied to e.g. functions 7, 8 or 9.

Table 15.15: Mean numbers of optimizations needed to construct the left membership function of in case of interactive input variables described by fuzzy intervals obtained with the parallel Fuzzy Calculator with an increasing number of α-cuts using PSO_GD for a population size of 20 particles with communication between the different swarms at every 5 iterations. The interactivity of the input variables is described by t-norms. Drastic_1 indicates that the interactivity of the input variables is described by the drastic t-norm and this is implemented through the adapted α-cut approach. Drastic_2 on the other hand refers to the application of the drastic t-norm but in this case the search space for every α-cut is divided into a number of (overlapping) hyperrectangles and the optimization problem is solved for each of these hyperrectangles separately.

	t-norm				
Test function	drastic_1	drastic_2	Łukasiewicz	product	minimum
1	22.4	18.0	18.0	18.0	18.0
2	7.0	3.0	3.0	3.0	3.0
3	3.0	23.2	13.2	11.8	16.6
4	9.9	13.0	12.6	11.2	10.9
5	7.2	7.0	6.2	7.0	7.0
6	31.3	21.0	23.8	26.5	21.8
7	34.4	24.2	14.5	15.5	15.9
8	36.0	21.6	13.4	10.8	15.5
9	41.0	23.0	13.6	17.4	15.6

Table 15.16: Mean numbers of optimizations needed to construct the right membership function for the different test functions in case of interactive input variables described by fuzzy intervals obtained with the parallel Fuzzy Calculator with an increasing number of α-cuts using PSO_GD for a population size of 20 particles with communication between the different swarms at every 5 iterations. The interactivity of the input variables is described by t-norms. Drastic_1 indicates that the interactivity of the input variables is described by the drastic t-norm and this is implemented through the adapted α-cut approach. Drastic_2 on the other hand refers to the application of the drastic t-norm but in this case the search space for every α-cut is divided into a number of (overlapping) hyperrectangles and the optimization problem is solved for each of these hyperrectangles separately.

	t-norm				
Test function	drastic_1	drastic_2	Łukasiewicz	product	minimum
1	12.6	3.0	3.0	3.0	3.0
2	36.3	33.0	27.0	15.0	32.3
3	39.7	33.0	34.9	28.4	33.6
4	40.6	29.5	31.5	32.1	18.0
5	47.9	35.4	34.1	35.0	35.2
6	36.6	19.1	26.9	30.7	37.2
7	44.4	33.0	34.0	33.7	37.8
8	51.5	33.1	30.0	30.5	34.2
9	52.0	33.0	25.5	32.8	42.7

16

Case study

The surface soil moisture content is an important property in the understanding and modelling of meteorology, hydrology and agriculture [145, 146]. Soil moisture is responsible for the partitioning of precipitation in surface runoff and infiltration, as well as the partitioning of the incoming radiation in latent and sensible heat fluxes [147, 148]. Therefore, monitoring soil moisture is of high importance. As field measurements of soil moisture are not feasible for large areas, there is a need for other methods to retrieve information about the moisture content of the soil's surface. A solution to this problem is provided by the relation that exists between the backscatter coefficients of microwaves, obtained in radar images, and the surface soil moisture. Furthermore, microwave remote sensing is preferred for soil moisture retrieval as it is insensitive to clouds. However, in order to be useful for hydrological applications, the backscattering coefficient needs to be converted to the corresponding soil moisture content. Therefore, several types of models exist, ranging from purely empirical relationships to physically-based surface scattering models.

In this chapter, the Fuzzy Calculator, developed and thoroughly tested in Chapters 14 and 15, is applied in order to compute the fuzzy output of a physically-based surface scattering model that allows for predicting the backscatter of an incident microwave pulse on a rough natural surface. The model in this case study is the Integral Equation Model [149, 150], which was delivered to us by professor N. Verhoest of the Department of Forest and Water Management of Ghent University. This model is developed to describe the relation between the backscatter coefficient, the dielectric constant and the roughness parameters of a dielectric surface. The dielectric constant of a soil is related to its soil moisture content. This implies that when the backscatter coefficient and the roughness parameters, namely the root mean square height and the correlation length, are available, the surface soil moisture can be calculated by inverting the model. While backscatter coefficients can be obtained from radar images, the determination of surface roughness is more complex. The measuring of surface roughness is rather difficult and strongly depends on the profile length [141, 151, 152, 153, 154, 155] or the type of measuring device used [141, 151]. To incorporate this uncertainty in surface roughness, the use of possibility distributions to present the possible values of the roughness parameters was introduced by Verhoest *et al.* [116] and improved by Vernieuwe *et*

al. [141]. In [141], it is assumed that the Integral Equation Model is a locally monotone function of the rms height and the correlation length, indicating that the minimum and maximum values of the soil moisture result from roughness parameters located on the boundary of the joint possibility distribution. In this case study, we no longer follow this assumption and also take into account the interior of the joint possibility distribution in order to construct the possibility distribution of the soil moisture. We will restrict ourselves to a certain roughness type resulting from a 'rotary tillage'. We use the membership functions for the roughness parameters, which act as the fuzzy inputs to the inverse IEM, as they were obtained by Vernieuwe *et al.* [141]. We follow the same approach as in [141] and also make a distinction between assuming non-interactive roughness parameters and working with a joint possibility distribution describing the interactivity between the roughness parameters. While Vernieuwe *et al.* [141] only concentrated on the boundaries of the search space constructed by the membership functions of the roughness parameters, in our Fuzzy Calculator the total search space is taken into account in order to construct the membership function of the output of the inverse Integral Equation Model. Further on, we refer to the method applied in [141] as the algorithm of Vernieuwe *et al.*

16.1. Integral Equation Model

As mentioned in the introduction, the IEM describes the relation between the backscatter coefficient, the dielectric constant and the roughness parameters of a dielectric surface. This model is based on the important observation that the dielectric constant (ϵ) of the soil varies considerably as the soil moisture content changes. Several models exist that describe the relation between the dielectric constant and the soil moisture content. This dielectric constant, a measure of the polarizability of a material under applied electric fields, can be obtained from radar images. Radar is an active system that emits microwave pulses. For the application of the IEM, we are interested in measuring the reflected or backscattered waves of radar pulses that were emitted by a satellite. The backscattering intensity is a function of the dielectric constant of the scattering medium, which is mainly the earth's surface. The backscattering intensity is of course a complicated quantity that also depends on the roughness of the scattering medium and the incidence angle. Since the incidence angle is typically known and the dielectric constant can be calculated using soil moisture observations [156], the only additional variables that are required to compute the backscatter coefficient (σ^0), are related to the roughness of the surface.

The surface roughness can be modelled by two statistical roughness parameters, namely the root mean square (rms) height s and the correlation length ℓ. s is the standard deviation of the surface height and ℓ is defined as the minimal horizontal

distance between two points for which those points can be considered statistically independent of one another. When the surface is described by a function $z(x, y)$ that measures the height above an xy reference plane at point (x, y), the mean height of the surface can be calculated as

$$\overline{z} = \frac{1}{L_x L_y} \int_{-L_x/2}^{+L_x/2} \int_{-L_y/2}^{+L_y/2} z(x, y)\, \mathrm{d}x\mathrm{d}y\,, \tag{16.1}$$

with L_x and L_y the dimensions of a statistically representative segment of the soil surface centered at the origin. The variance of the height is then given by

$$\Delta z^2 = \frac{1}{L_x L_y} \int_{-L_x/2}^{+L_x/2} \int_{-L_y/2}^{+L_y/2} \left[z(x, y) - \overline{z}\right]^2 \mathrm{d}x\mathrm{d}y\,, \tag{16.2}$$

and defines the rms height s as

$$s = \sqrt{\Delta z^2}. \tag{16.3}$$

The correlation between two points as a function of their separation distance x can be obtained as

$$\rho(x) = \frac{\int_{-L_x/2}^{+L_x/2} \left[z(x') - \overline{z}\right]\left[z(x' + x) - \overline{z}\right] \mathrm{d}x'}{\int_{-L_x/2}^{+L_x/2} \left[z(x') - \overline{z}\right]^2 \mathrm{d}x'}\,. \tag{16.4}$$

In this definition of the autocorrelation function $\rho(x)$, a cut along a line parallel to the x-axis has been used. In principle, this cut can be taken along any direction. The mean value $\overline{z}$ can itself be spatially dependent when the terrain is not flat. However, $\overline{z}$ then varies over a much slower scale than z itself. By approximating the autocorrelation function $\rho(x)$ by a theoretical model, such as an exponential or Gaussian function, a definition of the correlation length ℓ can be obtained. Typically, the surface correlation length ℓ is defined as the displacement x such that $\rho(x = \ell) = 1/e$. Finally, we can also define a surface slope as $m = s/\ell$ for an exponentially autocorrelated surface and $m = \sqrt{2}s/\ell$ for a Gaussian autocorrelated function.

Having obtained accurate estimates of the surface roughness parameters, a model can be constructed that determines the relation between the dielectric constant of the surface, its roughness parameters and the resulting backscattering coefficient. Different models exist for different ranges of the parameters, *i.e.* linear regression, the model of Oh *et al.*, the model of Dubois *et al.* and the Integral Equation Model [149, 150, 157, 158]. The Integral Equation Model is the most generally applicable but also most complicated model. However, in the domain $ks < 3$ and $m < 0.4$, with $k = 2\pi/\lambda$ the wavenumber of the microwaves used by the radar and λ the corresponding wavelength, the Integral Equation Model can be approximated by the single scattering term. Put differently, in this regime it is safe

to ignore multi-scattering terms that describe microwaves that scatter multiple times with the surface before being reflected back to the radar sensor. Nevertheless, the relationship between the dielectric constant and the backscattering coefficient is still quite technical, and we refer to [159] for more details about this approach.

16.2. Site and data description

When backscatter coefficients and information of the roughness of a medium are available, the dielectric constant of this medium can thus be calculated using the inverse IEM, and can then be related to the soil moisture content, which is the quantity of interest. On March 19th, April 23th and May 28th of 2003, the second European Remote Sensing Satellite (ERS-2) collected C-band (frequency 5.4 Ghz) radar imagery with a 30 m resolution for the Zwalm catchment (south of Ghent, Belgium) and for the loamy region (southeast of Brussels, Belgium) with a local incidence angle between 20° and 24°. Backscatter coefficients were obtained from the Precision Resolution Images (PRI) where both the incident and the scattered wave were vertically polarized (VV). These images were georeferenced using a first-order affine transformation with an overall accuracy of less than 50 m. In addition to the backscatter data, average soil moisture values were also available and were used in [141] to check whether the membership functions that were derived for the roughness parameters produce a membership function for soil moisture that is compatible with the observations. As for the Zwalm catchment average soil moisture values are available for fields of 4 to 5 ha, average backscatter coefficients were calculated for these fields. For the loamy regions, average soil moisture values are available for fields of 10 ha and therefore average backscatter coefficients were calculated for fields of 10 ha. In total, we have 16 backscatter coefficients and corresponding soil moisture values. The soil texture of these fields was loamy and the soil roughness resulted from a rotary tillage. This data was delivered to us by professor N. Verhoest of the Department of Forest and Water Management of Ghent University.

While no roughness measurements are available for these fields, [141] had at their disposal roughness measurements from other fields. These were used by Vernieuwe *et al.* to generate four different synthetic data sets for the roughness class rotary tillage, namely three data sets of 1000 (s,ℓ)-couples corresponding to a profile length of 25, 4 and 1 m, based on roughness measurements from a bare loamy sand field in the center of Eastern Flanders, Belgium, and one data set of 1000 (s,ℓ)-couples for a profile length of 1 m, based on roughness measurements from six different field campaigns at five European test sites [160]. In case no interaction is taken into account between the rms height and the correlation length, [141] generated possibility distributions for the rms height and the correlation length for these different data sets, by applying a probability-possibility transformation

[161, 162] and the method of Ban, which makes a trapezoidal approximation of a fuzzy number while preserving the expected interval of the fuzzy number [163]. As illustrated in Figure 16.1 there is a clear interaction between the rms height and the correlation length. Because of the elliptical shape of the (s,ℓ)-couples in the four data sets (see Figure 16.1 for the data set of the (s,ℓ)-couples generated with a profile length 4 m), the possibilistic Gustafson-Kessel fuzzy clustering algorithm is used in [141] to determine the joint possibility distribution for the rms height and the correlation length. Henceforth, we concentrate on this particular data set.

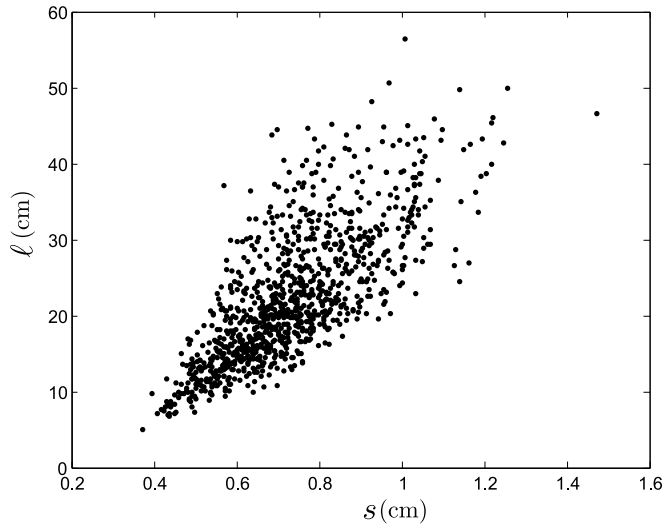


Figure 16.1: Plot of the 1000 (s,ℓ)-couples generated with a profile length 4 m

16.3. Possibilistic Gustafson-Kessel algorithm

The possibilistic Gustafson-Kessel clustering algorithm differs from the probabilistic Gustafson-Kessel clustering algorithm in the way that the probabilistic constraint, which forces the membership degrees of a data point across the different classes to sum up to one, is no longer present [164]. In this way, the membership degrees can be interpreted as degrees of compatibility. In the probabilistic Gustafson-Kessel algorithm the following objective function needs to be minimized

$$\sum_{i=1}^{C}\sum_{j=1}^{N} u_i(\boldsymbol{x}_j)^m d(\boldsymbol{x}_j,\boldsymbol{c}_i)^2\,, \tag{16.5}$$

with $d(\boldsymbol{x}_j,\boldsymbol{c}_i)$ the distance of data point $\boldsymbol{x}_j$ to the cluster center $\boldsymbol{c}_i$ of cluster C_i, N

the total number of data points, C the number of clusters, $u_i(\boldsymbol{x}_j)$ the membership degree of the data point $\boldsymbol{x}_j$ in cluster C_i and $m \in]1, \infty[$ a weighing exponent called the fuzzifier. The membership degrees $u_i(\boldsymbol{x}_j)$ have to satisfy following conditions

$$u_i(\boldsymbol{x}_j) \in [0,1]\,, \quad \text{for all} \quad i = 1, \ldots, C, \quad j = 1, \ldots, N \tag{16.6}$$

and

$$\sum_{i=1}^{C} u_i(\boldsymbol{x}_j) = 1 \quad \text{for all} \quad j\,. \tag{16.7}$$

The probabilistic constraint $\sum_{i=1}^{C} u_i(\boldsymbol{x}_j) = 1$ for all j was added to the fuzzy clustering algorithm in order to prevent the trivial solution, namely a membership degree of zero for all feature points. Nevertheless, through the following adaptation of the objective function Eq. (16.5) it is possible to prevent this trivial solution without adding the probabilistic constraint [164]

$$\sum_{i=1}^{C}\sum_{j=1}^{N} u_i(\boldsymbol{x}_j)^m d(\boldsymbol{x}_j, \boldsymbol{c}_i)^2 + \sum_{i=1}^{C} \eta_i \sum_{j=1}^{N} (1 - u_i(\boldsymbol{x}_j))^m\,, \tag{16.8}$$

where η_i is a parameter that determines the distance at which the membership value of a point in a cluster becomes 0.5

$$\eta_i = K \frac{\sum_{k=1}^{N} u_i(\boldsymbol{x}_k)^m d(\boldsymbol{x}_k, \boldsymbol{c}_i)^2}{\sum_{k=1}^{N} u_i(\boldsymbol{x}_k)^m}\,, \tag{16.9}$$

with $u_i(\boldsymbol{x}_k)$ the membership degrees determined by an initial probabilistic fuzzy clustering algorithm and K typically chosen to be one. Since the aim here is to construct a joint possibility distribution, a single cluster needs to be determined for a given roughness class, which makes it impossible to determine the initial membership degrees $u_i(\boldsymbol{x}_k)$ with a probabilistic fuzzy clustering algorithm, since the latter would always create at least 2 clusters [141, 164]. Therefore, η_1 is initially determined as follows

$$\eta_1 = K \frac{\sum_{j=1}^{N} d(\boldsymbol{x}_j, \boldsymbol{c}_1)^2}{N}\,. \tag{16.10}$$

Then, the possibilistic clustering algorithm is applied with this value and the value of η_1 is recalculated by Eq. (16.9) using the membership degrees to the possibilistic cluster, which are calculated as follows

$$u_1(\boldsymbol{x}_j) = \frac{1}{1 + (\frac{d(\boldsymbol{x}_j, \boldsymbol{c}_1)^2}{\eta_1})^{\frac{1}{m-1}}}\,. \tag{16.11}$$

With this new value of η_1 the possibilistic clustering algorithm is again applied.

These steps are repeated until the value of η_1 converges.

The distance measure used in the Gustafson Kessel algorithm is

$$d(\boldsymbol{x}_j, \boldsymbol{c}_i)^2 = (\boldsymbol{x}_j - \boldsymbol{c}_i)^T A_i (\boldsymbol{x}_j - \boldsymbol{c}_i)\,, \tag{16.12}$$

with $\boldsymbol{c}_i$ the center of cluster C_i, which can be updated as follows

$$\boldsymbol{c}_i = \frac{\sum_{j=1}^{N} u_i(\boldsymbol{x}_j)^m \boldsymbol{x}_j}{\sum_{j=1}^{N} u_i(\boldsymbol{x}_j)^m}\,, \tag{16.13}$$

and A_i a positive definite matrix associated with each cluster C_i, which implies that data points with equal distances to the cluster centers are positioned on ellipsoids. A_i can be calculated as follows

$$A_i = |F_i|^{1/r} (F_i)^{-1}\,, \tag{16.14}$$

with r the dimension of the data and F_i the fuzzy covariance matrix

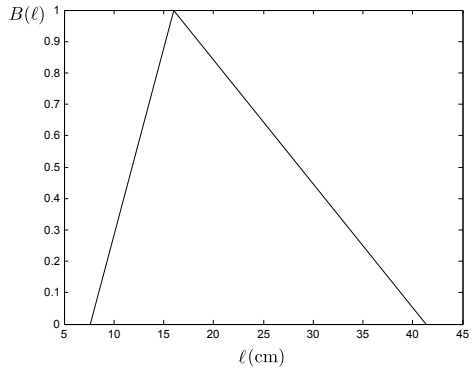
$$F_i = \frac{\sum_{j=1}^{N} u_i(\boldsymbol{x}_j)^m (\boldsymbol{x}_j - \boldsymbol{c}_i)(\boldsymbol{x}_j - \boldsymbol{c}_i)^T}{\sum_{j=1}^{N} u_i(\boldsymbol{x}_j)^m}\,. \tag{16.15}$$

16.4. Results

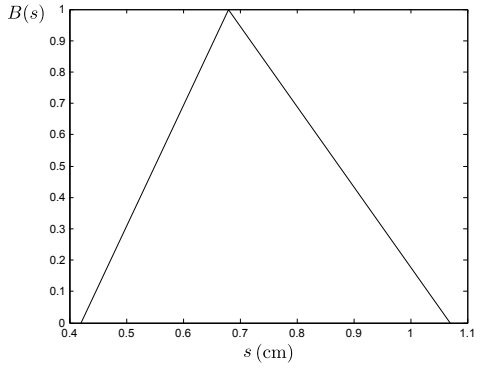
In this section, the results obtained with the Fuzzy Calculator are discussed. While in Section 16.4.1 it is assumed that no interactivity is present between the correlation length ℓ and the rms height s, Section 16.4.2 presents the results when the interactivity between ℓ and s is taken into account.

16.4.1. Non-interactive roughness parameters

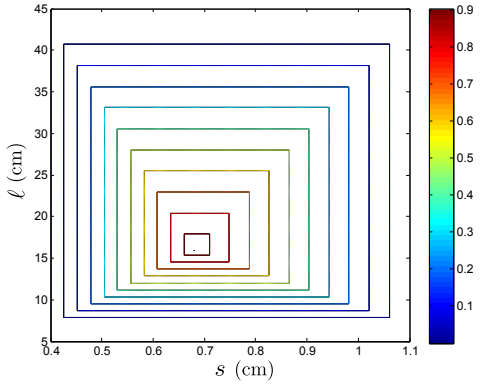
As the algorithm of Vernieuwe *et al.* [141] works with a fixed number of α-cuts ($m = 11$), we apply the Fuzzy Calculator with the same fixed number of α-cuts as well as the Fuzzy Calculator starting with an increasing number of α-cuts. Both Fuzzy Calculators, using PSO_GD with a population size of 20 particles as optimization algorithm (as described in Chapter 14), can directly be applied to the case of non-interactive roughness parameters with membership functions calculated by the method of Ban [163]. The roughness parameters ℓ and s are described by the membership functions presented in Figure 16.2(a) and (b) [141]. The search area resulting from non-interactive roughness parameters is presented in Figure 16.2(c).



(a)



(b)



(c)

Figure 16.2: (a) Membership function of the correlation length ℓ calculated with the method of Ban [141], (b) membership function of the rms height s calculated with the method of Ban [141] and (c) search area in case of non-interactive ℓ and s.

Table 16.1 shows the areas under the membership functions of the soil moisture, calculated with the IEM for 16 different backscatter coefficients, as constructed with the Fuzzy Calculator with a fixed number of α-cuts using the first correction approach to deal with inconsistencies (Fuzzy Calculator$_1$), with the Fuzzy Calculator starting with 3 α-cuts using the second correction approach to deal with inconsistencies (Fuzzy Calculator$_2$), and using PSO_GD with a population size of 20 particles as optimization algorithm and the algorithm of Vernieuwe *et al.* This table illustrates that only a small difference is present between the different algorithms. Although the difference is negligible, the Fuzzy Calculator starting with 3 α-cuts leads to slightly higher mean areas under the membership function. Since the membership functions of the roughness parameters are triangular, *i.e.* they start in a single point at $\alpha = 1$, it is possible to construct a correct membership function for the output variable by only looking at the boundary of the search spaces, provided that one starts at $\alpha = 1$ and finds the optima for successively lower α-cuts for different α values which are sufficiently densely spaced. When for a certain α-cut the exact optimum is not situated at the boundary but in the interior, the boundary search will of course result in an incorrect value. But this better optimum should already have been found for a higher value α' for which it was lying at the boundary of the corresponding search space. Hence, this higher value $\alpha' > \alpha$ would have a lower minimum or a higher maximum, which is of course impossible and requires to conclude that this better value is also to be used at level α. Since we started with a search space that is a single point at $\alpha = 1$, all optima in the interior of an α-cut α will be lying at the boundary of a certain α-cut $\alpha' > \alpha$. Of course, this is not a feasible approach in general, since we can never know whether the α-cuts are sufficiently densely spaced, and since it is in general much more complex to search exactly at the boundary of a region than in the interior. Therefore, we can conclude that for non-monotone functions our Fuzzy Calculator will be less complicated and more generally applicable than the algorithm of Vernieuwe *et al.* Figure 16.3 presents the membership function of the soil moisture obtained with the algorithm of Vernieuwe *et al.*, with Fuzzy Calculator$_1$ and with Fuzzy Calculator$_2$ for $\sigma^0 = -7,1148$. This figure also illustrates the negligible difference between the different algorithms.

16.4.2. Interactive roughness parameters

In this section, we take the interactivity between s and ℓ into account. As shown in Figure 16.1 the shape of the (s, ℓ) data can be approximated by elliptic clusters and therefore the possibilistic Gustafson-Kessel clustering algorithm is used in order to describe the interactivity between s and ℓ. In case of this interactivity, the Fuzzy Calculator developed for interactivity described by t-norms (Chapter 15, Section 15.3.1) can be directly applied. The only difference is that here the search spaces are the ellipsoids generated by the possibilistic Gustafson-Kessel clustering

Table 16.1: Area under the membership functions of the soil moisture calculated with the IEM for 16 different backscatter coefficients σ^0 in the case of non-interactive roughness parameters, as constructed with the Fuzzy Calculator with a fixed number of α-cuts using the first correction approach to deal with inconsistencies (Fuzzy Calculator$_1$), with the Fuzzy Calculator starting with 3 α-cuts using the second correction approach to deal with inconsistencies (Fuzzy Calculator$_2$), using PSO_GD with a population size of 20 particles as optimization algorithm and with the algorithm of Vernieuwe *et al.*

σ^0	Vernieuwe *et al.* [141]	Fuzzy Calculator$_1$	Fuzzy calculator$_2$
-7.1765	0.2622	0.2622	0.2623
-6.7528	0.2347	0.2333	0.2350
-9.1138	0.2871	0.2865	0.2876
-6.8223	0.2729	0.2719	0.2734
-6.7210	0.2530	0.2520	0.2532
-5.4604	0.1606	0.1562	0.1552
-7.2405	0.2881	0.2883	0.2882
-7.1442	0.2877	0.2880	0.2878
-6.3311	0.2760	0.2757	0.2764
-6.2298	0.2713	0.2697	0.2712
-5.6377	0.2548	0.2551	0.2556
-5.3827	0.2377	0.2354	0.2371
-7.5065	0.2884	0.2876	0.2888
-6.1353	0.2732	0.2727	0.2736
-6.9085	0.2854	0.2845	0.2859
-7.1148	0.2866	0.2861	0.2872

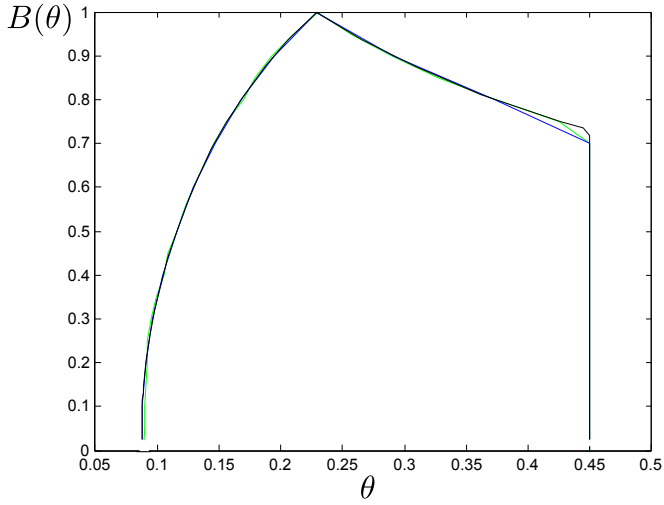


Figure 16.3: Membership function of the soil moisture constructed with the algorithm of Vernieuwe *et al.* (green), with the Fuzzy Calculator$_1$ (blue) and with the Fuzzy Calculator$_2$ (black) for $\sigma^0 = -7,1148$.

algorithm. This leads to the search space presented in Figure 16.4. The color bar in Figure 16.4 differentiates between the different α-cuts

Table 16.2 presents the areas under the membership functions of the soil moisture calculated with the IEM for 16 different backscatter coefficients σ^0, in case of interactive roughness parameters where the interactivity is described by the clusters generated possibilistic Gustafson-Kessel clustering algorithm, as constructed with the Fuzzy Calculator with a fixed number of α-cuts using the first correction approach to deal with inconsistencies (Fuzzy Calculator$_1$), with the Fuzzy Calculator starting with 3 α-cuts using the second correction approach to deal with inconsistencies (Fuzzy Calculator$_2$), using PSO_GD with a population size of 20 particles as optimization algorithm and with the algorithm of Vernieuwe *et al.* As in Section 16.4.1, we can conclude that the difference in area under the membership function between the different algorithms is negligible.

Figure 16.5 illustrates the search space in case of non-interactive roughness parameters described by the method of Ban [163], the search space in case of interactive roughness parameters where the interactivity is described by the clusters generated with the possibilistic Gustafson-Kessel algorithm and the synthetically generated (s, ℓ) data. This figure shows that the search space in case of non-interactive roughness parameters only contains a part of the search space in case of interactive roughness parameters. Therefore it is not possible to draw some conclusions about the difference in area under the membership function of the soil moisture resulting from these different search spaces. Important to note is that the search space resulting from non-interactive roughness parameters only contains a part of the

Table 16.2: Area under the membership functions of the soil moisture calculated with the IEM for 16 different backscatter coefficients σ^0 in case of interactive roughness parameters described by the clusters generated with the possibilistic Gustafson-Kessel clustering algorithm, as constructed with the Fuzzy Calculator with a fixed number of α-cuts using the first correction approach to deal with inconsistencies (Fuzzy Calculator$_1$), with the Fuzzy Calculator starting with 3 α-cuts using the second correction approach to deal with inconsistencies (Fuzzy Calculator$_2$), using PSO_GD with a population size of 20 particles as optimization algorithm and with the algorithm of Vernieuwe *et al.*.

σ^0	Vernieuwe *et al.* [141]	Fuzzy Calculator$_1$	Fuzzy calculator$_2$
-7.1765	0.1390	0.1389	0.1392
-6.7528	0.0822	0.0809	0.0809
-9.1138	0.1080	0.1074	0.1082
-6.8223	0.1544	0.1539	0.1543
-6.7210	0.1215	0.1215	0.1220
-5.4604	0.0293	0.0281	0.0280
-7.2405	0.1322	0.1331	0.1333
-7.1442	0.1322	0.1332	0.1333
-6.3311	0.1518	0.1525	0.1528
-6.2298	0.1524	0.1530	0.1533
-5.6377	0.1355	0.1359	0.1363
-5.3827	0.0980	0.0971	0.0971
-7.5065	0.1259	0.1249	0.1254
-6.1353	0.1523	0.1529	0.1531
-6.9085	0.1421	0.1416	0.1420
-7.1148	0.1389	0.1397	0.1372

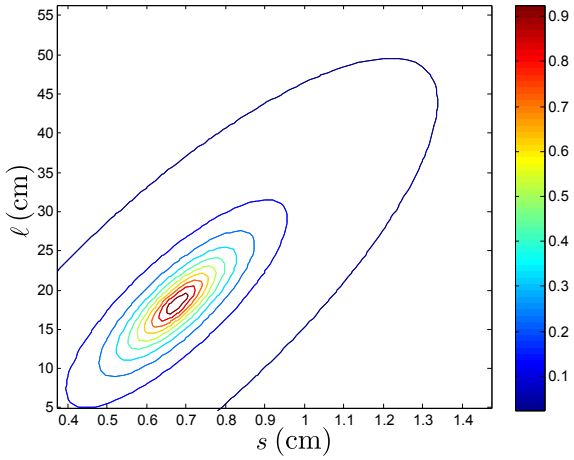


Figure 16.4: Search area for the different α-cuts. The color bar differentiates between the different α-cuts.

synthetically generated data [141]. Consequently, not all information is present in the membership function of the roughness parameters generated by the method of Ban. When we want to work with non-interactive roughness parameters that include almost all information of the synthetically generated data, a possible solution is to work in the rectangles that enclose the ellipsoids and are parallel to the s- and ℓ-axis. This search space is presented in Figure 16.6.

Table 16.3 presents the areas under the membership functions of the soil moisture calculated with the IEM for 16 different backscatter coefficients σ^0, in case of non-interactive roughness parameters described by the rectangles that enclose the ellipsoids corresponding to the different α-cuts and parallel to the s- and ℓ-axis, as constructed with the Fuzzy Calculator with a fixed number of α-cuts using the first correction approach to deal with inconsistencies (Fuzzy Calculator$_1$), with the Fuzzy Calculator starting with 3 α-cuts using the second correction approach to deal with inconsistencies (Fuzzy Calculator$_2$), using PSO_GD with a population size of 20 particles as optimization algorithm and with the algorithm of Vernieuwe *et al.* When we compare the different algorithms to construct the membership function of the soil moisture, we can again conclude that the Fuzzy Calculator$_2$ leads to the higher area under the membership function but that the differences are negligible. As this search space completely includes the search space in case of interactive roughness parameters, we expect that the membership functions in case of this non-interactivity also include the membership functions in case of interactive roughness parameters. In Figure 16.7, the membership function for $\sigma^0 = -7,1148$ is illustrated and confirms this expectation.

Table 16.3: Area under the membership functions of the soil moisture calculated with the IEM for 16 different backscatter coefficients σ^0, in case of non-interactive roughness parameters described by the rectangles that enclose the clusters generated by the possibilistic Gustafson-Kessel clustering algorithm and are parallel to the s- and ℓ-axis, as constructed with the Fuzzy Calculator with a fixed number of α-cuts using the first correction approach to deal with inconsistencies (Fuzzy Calculator$_1$), with the Fuzzy Calculator starting with 3 α-cuts using the second correction approach to deal with inconsistencies (Fuzzy Calculator$_2$), using PSO_GD with a population size of 20 particles as optimization algorithm and with the algorithm of Vernieuwe *et al.*.

σ^0	Vernieuwe *et al.* [141]	Fuzzy Calculator$_1$	Fuzzy calculator$_2$
-7.1765	0.2314	0.2310	0.2348
-6.7528	0.1953	0.1950	0.1955
-9.1138	0.2490	0.2488	0.2507
-6.8223	0.2507	0.2512	0.2539
-6.7210	0.2185	0.2189	0.2215
-5.4604	0.1145	0.1094	0.1132
-7.2405	0.2665	0.2662	0.2681
-7.1442	0.2657	0.2658	0.2678
-6.3311	0.2566	0.2568	0.2599
-6.2298	0.2502	0.2497	0.2523
-5.6377	0.2247	0.2250	0.2285
-5.3827	0.2005	0.2006	0.2020
-7.5065	0.2623	0.2629	0.2642
-6.1353	0.2534	0.2534	0.2565
-6.9085	0.2662	0.2665	0.2689
-7.1148	0.2670	0.2671	0.2694

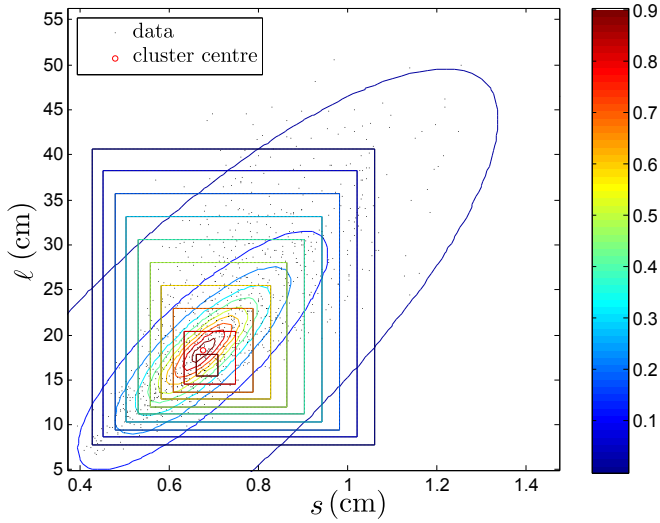


Figure 16.5: Search area for the different α-cuts in case of non-interactive roughness parameters described by the method of Ban (rectangles) and in case of interactivity described by the possibilistic Gustafson-Kessel algorithm (ellipsoids). The color bar differentiates between the different α-cuts.

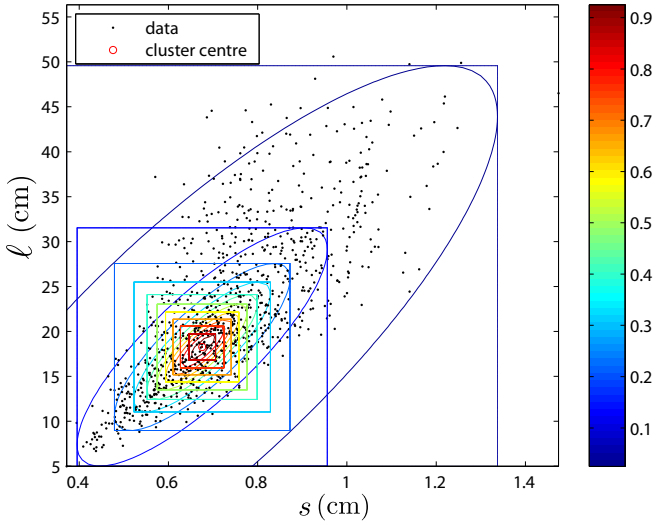


Figure 16.6: Search area for the different α-cuts in case of interactivity described by the clusters generated with the possibilistic Gustafson-Kessel algorithm (ellipsoids) and in case of non-interactivity with a rectangular search space enclosing the ellipsoids and parallel to the s- and ℓ-axis. The color bar differentiates between the different α-cuts.

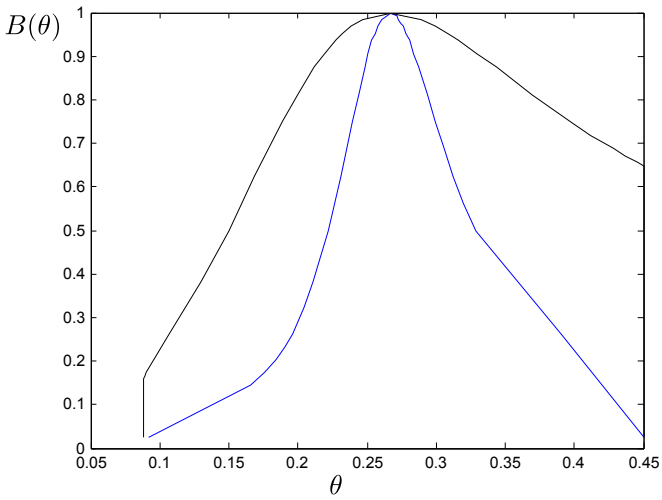


Figure 16.7: Membership function of the soil moisture constructed with the Fuzzy Calculator$_2$, in case of interactive roughness parameters where the interactivity is described by the clusters generated with the possibilistic Gustafson Kessel algorithm (blue line) and in case of non-interactivity with a rectangular search space enclosing the ellipsoids and parallel to the s- and ℓ-axis (black line).

17

Conclusion

In this part of this disseration, a Fuzzy Calculator was designed to efficiently compose the membership function of the output of a continuous function of non-interactive as well as interactive input variables described by fuzzy intervals. The solution to this problem is given by Zadeh's extension principle. However, a direct implementation of this principle is computationally infeasible for practical applications. Therefore, based on the α-cut approach, we transformed this problem into a number of optimization problems (Chapter 14).

In Chapter 15, different optimization algorithms were compared for the case of non-interactive input variables described by fuzzy intervals: Gradient Descent based on Sequential Quadratic Programming (GD), Simplex-Simulated Annealing (SIMPSA), Particle Swarm Optimization (PSO) and Particle Swarm Optimization in combination with Gradient Descent (PSO_GD). In addition two approaches were followed to determine the number of α-cuts. Both a non-parallel and a parallel implementation of the Fuzzy Calculator were designed.

In a first test configuration, the number of α-cuts was fixed to 11. As accuracy measure, we used the area under the membership function of the fuzzy output interval. Both for the non-parallel as well as for the parallel Fuzzy Calculator, the employment of PSO_GD with a population size of 20 particles resulted in a significantly more accurate membership function than the Fuzzy Calculator with any of the other optimization algorithms or parameter settings. In addition, in the parallel version, it was shown that it is beneficial to use communication. In particular communication at every 5 iterations leads to a significantly more accurate membership function than the other communication strategies. The corresponding number of function evaluations, however, is significantly higher for the parallel Fuzzy Calculator.

In a second test configuration, we started with 3 α-cuts and added additional α-cuts according to a linearity criterion. We continued on our previous results by restricting the Fuzzy Calculator to PSO_GD, with a population size of 20 particles and communication at every 5 iterations. The membership function composed by the Fuzzy Calculator with an increasing number of α-cuts is significantly more accurate than the membership function composed by the Fuzzy Calculator with a fixed number of 11 α-cuts. The number of function evaluations is, however,

significantly higher for the Fuzzy Calculator with the increasing number of α-cuts.

Two different approaches were described to correct for inconsistencies between obtained optima for subsequent α-cuts. The first approach replaces the optima of the inconsistent α-cuts by the optima of the nearest α'-cut with $\alpha' > \alpha$. The second approach recalculates the optima of the inconsistent α-cuts starting from the optimizers of the nearest α'-cut with $\alpha' > \alpha$. In the case of an increasing number of α-cuts, it is not necessary to recalculate the optima of the inconsistent α-cuts and it is thus sufficient to just remove these α-cuts. The first and second approach were compared for the Fuzzy Calculator starting with 3 α-cuts. No significant differences are detected between the accuracy of the membership functions composed by the Fuzzy Calculators using these different approaches. The number of function evaluations, however, is significantly lower for the Fuzzy Calculator using the second approach, which is a pleasing result.

We thus conclude that the best approach to construct the membership function of the output is to use the Fuzzy Calculator with an increasing number of α-cuts, with PSO_GD as optimization algorithm, using a population size of 20 particles and communication at every 5 iterations, using the second correction approach if inconsistencies between subsequent α-cuts occur. The number of function evaluations, however, can be quite high, depending on the number of α-cuts that will be constructed. This can be regulated by the tolerance level in the criterion for determining whether additional α-cuts are required. In addition, as the implementation is parallel and several processors can be used, an elevated number of function evaluations will not pose a major problem for most applications if a high performance facility is available.

We then applied this Fuzzy Calculator to the case of interactive input variables described by fuzzy intervals (Section 15.3, Chapter 15). In this chapter, we restricted ourselves to interactivity described by the four basic triangular norms. We modeled this interactivity by transforming the optimization problem into a constrained optimization problem. This is accomplished by adding the nonlinear constraint $T(A_1(x_{i,1}), \ldots, A_n(x_{i,n})) \geq 0$ to the optimization algorithm PSO_GD. For the drastic t-norm, the search region can also be divided into a number of overlapping hyperrectangles and the optimization problem can be solved in these hyperrectangles separately. This implementation (drastic_2) was compared with the implementation that deals with the interactivity by adding the nonlinear constraint to PSO (drastic_1). As implied theoretically, the area under the membership function is largest for the minimum t-norm (non-interactivity), followed by the product t-norm, followed by the Łukasiewicz t-norm and finally the drastic t-norm. The numbers of function evaluations are, except for drastic_1, approximately the same as in case of non-interactive input variables (*i.e.* minimum t-norm. For some test functions, drastic_1 requires a lot more function evaluations, which can be

attributed to the non-convex search region of this t-norm that leads to convergence difficulties for the PSO algorithm. This is also confirmed by the required number of optimizations to construct the left and right side of the membership function.

To conclude, in Chapter 16, the Fuzzy Calculator was applied to a case study. The objective of this case study was to investigate the practical applicability of the Fuzzy Calculator described and thoroughly tested in Chapters 14 and 15. This case study consisted of the application of the inverse IEM model that is able to calculate the soil moisture content of a medium when information is available about the backscatter coefficient σ^0 and the roughness parameters s and L. Backscatter coefficients can be obtained from radar images, but the determination of surface roughness is more complex leading to uncertainty in surface roughness. In order to propagate this uncertainty in surface roughness through the IEM model with as a result a membership function for the soil moisture, we applied the Fuzzy Calculator with a fixed number of α-cuts as well as the Fuzzy Calculator with an increasing number of α-cuts. Both Fuzzy Calculators, using the second correction approach if inconsistencies between subsequent α-cuts occur, using PSO_GD with a population size of 20 particles are used as optimization algorithm. We made use of the same data as in Vernieuwe *et al.* [141] and compared our results with the results presented in this work. The major difference with our algorithm is that in Vernieuwe *et al.* only the boundary of the search region is taken into account in order to construct the membership function of the soil moisture. Our approach takes into account the complete search region and is therefore more generally applicable. The difference in area under the membership function, constructed by the different algorithms, is negligible for the model under study, but because of the more general applicability of our Fuzzy Calculator we can conclude that the Fuzzy Calculator might be better suited than the algorithm of Vernieuwe *et al.* for other applications.

General conclusions and outlook

We now summarize the main conclusions that can be drawn from the work in this dissertation, and highlight some aspects that might be interesting for further research. As suggested by the title of this dissertation, the overall objective was to examine the capability of general metaheuristic optimization strategies versus those of problem-specific strategies. In particular, we have focussed on this research question in the context of the biosciences and with the emphasis on applicability.

In Part II, we solved a special type of subset selection problem, more specifically the selection of a subset from a multi-experiment data set. Several problem-specific techniques were already developed for the problem of subset selection. We compared these techniques with more general strategies based on Genetic Algorithms (GA) or Ant Colony optimization (ACO). Of course, slight modifications of these heuristic algorithms to the specific problem were in order. But this does not eradicate the general heuristics lying at the heart of the optimization algorithms that were used to solve this specific problem. We can conclude that, for this subset selection problem, both GA and ACO lead to better results than the conventional techniques, since these were not constructed to take the multi-experiment aspect into account and cannot easily be generalized as such. The best results were obtained with GA, which thus outperformed ACO. We had to modify the ACO algorithm in order to eliminate a negative bias that was present in the original formulation. We have also introduced slight modifications to the GA algorithm, in order to impose a fixed number of selected samples.

Part III focused on the calibration of a water and energy balance hydrologic model. The aim of the calibration process is to determine the best set of model parameters, such that the hydrologic model is able to predict a faithful value for the output variables for given input data. This best set of parameters is chosen by comparing the model output to measured data for the output variables. The method Multistart Weight-Adaptive Recursive Parameter Estimation (MWARPE) was developed for the specific problem of finding the set of parameter values for which the model transforms given input data to output data that best resemble the measured output corresponding to that input. In this algorithm, all output variables are explicitly taken into account. As this method has a number of disadvantages, we compared this method with a more general approach where we formulate the calibration process as an optimization problem. We then had to determine a proper objective function for which the global minimizer corresponds to the best set of model parameters. For this problem, we chose to work with the 'Root Mean Square Error' (RMSE) between the measured output and the model

prediction. The RMSE should be calculated for each output variable separately. There are two possible configurations for taking into account the different output variables. The first configuration starts by rescaling the data of the individual output variables so that no difference in order of magnitude is present between the data corresponding to the different output variables. Then, the different RMSE values of the standardized output variables are merged into a single-objective function by summation. The second configuration consists of handling the different objective functions separately in a multiple-objective framework by the construction of a Pareto front. This approach is more complex, especially when a large number of output variables are present. As our objective was to compare MWARPE with a simple and general approach, we restricted ourselves to the first configuration. The resulting minimization of the overall RMSE was solved with Particle Swarm Optimization (PSO). From this part of this dissertation, we can conclude that both approaches yield comparable results. However, given the flexibility and generality of PSO, as well as the simplicity of its implementation, there are few reasons to choose for the more complex and restrictive MWARPE approach. Note that MWARPE was already working at its computational limits in the case study of part III, and was not able to incorporate all test data due to its high computational demands.

For the last part of this dissertation (Part IV), the goal was set to develop a Fuzzy Calculator that propagates uncertainty through complex functions and models. In case of monotone functions or models, several specific strategies exist to propagate uncertainty and to determine the membership function of the output. However, in case of non-monotone functions or models, no general strategy was present to complete this task. Such a general strategy was here developed by transforming the uncertainty propagation problem to an optimization problem, making use of the α-cut approach of Nguyen for non-interactive input variables. For solving this optimization problem, we used several optimization algorithms: a local optimization algorithm, namely Gradient Descent based on Sequential Quadratic Programming (GD) and two global optimization algorithms, namely Simplex-Simulated Annealing (SIMPSA) and PSO. Two different strategies were used to determine the number of α-cuts. In the first approach, the number of α-cuts was fixed to a predetermined number, while in the second approach the number of α-cuts was initially chosen very small and subsequently increased until a criterion based on linear interpolation was satisfied. Two approaches were also developed to deal with inconsistencies between the α-cuts, which are caused by the optimization algorithm not having found the correct global optimum for one of the α-cuts. In the first approach, the optima of the inconsistent α-cuts were replaced by the optima of the nearest α'-cut with $\alpha' > \alpha$. In the second approach, the optima of the inconsistent α-cuts were recalculated starting from the optimizers of the nearest α'-cut with $\alpha' > \alpha$. A parallel as well as a non-parallel implementation was developed. When PSO was used as optimization algorithm, the parallel implementation was extended to

support communication between the different swarms about possible candidate solutions. We compared the different Fuzzy Calculators based on 9 benchmark functions, for which the area under the membership function was determined. The best results were obtained with the Fuzzy Calculator with an increasing number of α-cuts, with PSO_GD as optimization algorithm, using a population size of 20 particles and communication at every 5 iterations, and by recalculating the optima if inconsistencies between subsequent α-cuts occur. Then, based on a generalisation of Nguyen's α-cut approach for interactive variables, we applied this Fuzzy Calculator to the same benchmark functions in case of interactive input variables. Firstly, we restricted ourselves to interactivity described by the four basic t-norms and recovered the expected relation between the different areas under the membership function obtained with our Fuzzy Calculator. Finally, in order to examine the practical applicability, the Fuzzy Calculator was applied to a case study. This case study consisted of the application of the inverse Integral Equation Model (IEM) to calculate the soil moisture content from a rough bare surface with as input backscatter values and roughness parameters of this surface. The roughness parameters are the uncertain parameters that have to be propagated through the model. We compared the results of the Fuzzy Calculator with a specific approach developed for this problem by Vernieuwe *et. al.* [141]. Since the difference in the results was negligible, the aspect of general applicability votes in favour of our Fuzzy Calculator.

For the optimization problems studied in this dissertation, it seems that we can conclude that the metaheuristic optimization approaches have defeated the problem-specific approaches. While the results are not always significantly better, the fact that the metaheuristic optimization approaches are more general and can often be very simply implemented, allows to prefer them over the specific algorithms in practical situations. Therefore, it would be very interesting to examine the added value of these metaheuristic approaches in several other optimization problems in different applications. Nevertheless, the quest for better optimization algorithms, either general or problem specific, should not be terminated, as further improvement is always possible.

One direction in which the improvement of solutions for certain problems can be sought is by taking into account the multiple-objective nature of these problems. In this dissertation, we have only dealt with single-objective optimization problems. However, in the biosciences, many optimization problems have more than one objective function. We have already encountered the problem of multiple-objective functions in the calibration of the hydrologic model (Part III). Other examples include the multiple objectives of forest management (economic benefits of timber production, environmental benefits such as carbon sequestration, biodiversity and water quality, and social benefits such as recreation, public health and community involvement) [165], multiple-objective planning in agriculture (maximum income with a minimum of irrigation water, use of fertilizers and number of workers), opti-

mization of the removal of multiple compounds in water purification installations, *etc.* The construction of a single aggregate objective function, as we did for the calibration of the hydrological model (Part III), is only one possible approach to deal with multiple-objective optimization problems. For different weight factors in the aggregate objective function, we obtain different optimizers, which are all lying within a set of points $\mathcal{P}$ that is called the Pareto-optimal set. A point $\boldsymbol{x} \in \mathcal{P}$ is Pareto optimal if there are no other points $\boldsymbol{x}'$ within the set of feasible points such that $f_i(\boldsymbol{x}') \leq f_i(\boldsymbol{x})$ for all i, with at least one inequality being strict and where f_i denote the different objective functions. The image of the Pareto-optimal set $\boldsymbol{f}(P)$ is called the Pareto front [166]. The equivalent of finding an optimizer and corresponding optimum for a single-objective optimization problem is to construct the Pareto-optimal set and corresponding Pareto front in the case of a multiple-objective optimization problem. The construction of the Pareto front becomes very complex when the multiple-objective optimization problem has a large number of different objectives. Many algorithms exist for constructing the Pareto front, e.g. the normal-boundary intersection method [167], the normal constraint method [168], the Successive Pareto Optimization [169], *etc.* Next to these classical methods, evolutionary algorithms were also modified in order to solve multi-objective optimization problems [170], resulting in e.g. multiple-objective Genetic Algorithms [171, 172], multiple-objective Particle Swarm Optimization [3], multiple-objective Ant Colony Systems [173], *etc.*

For future research, it would thus be interesting to perform a similar comparison between specific methods and metaheuristic (evolutionary) algorithms for the case of multiple-objective optimization problems. Despite the greater mathematical complexity, multiple-objective approaches can also result in an improvement over the results obtained with single-objective methods, such as in the case of the hydrologic model. Also in the case of the subset selection problem it can be useful to include a second objective, namely the deviation of the mean, which was in this dissertation only used as a verification of the resulting subset after the subset had been constructed from a single-objective optimization problem.

Dutch summary — Nederlandstalige samenvatting

Het centrale thema dat doorheen deze doctoraatsthesis terugkeert is het begrip optimalisatie. Meer specifiek behandelen we in deze thesis de zogenaamde metaheuristische optimalisatietechnieken, met speciale aandacht voor de populatiegebaseerde metaheuristische optimalisatietechnieken. Het doel van dit doctoraat is om de meerwaarde van deze algemene optimalisatietechnieken ten opzichte van probleemspecifieke aanpakken te bepalen. Dit werd bestudeerd voor een aantal biologische toepassingen. Dit doctoraat is opgebouwd uit vier delen. Deel I biedt een theoretisch overzicht van optimalisatie en vormt dus de noodzakelijke achtergrond voor het vervolg van de thesis. Deel II bespreekt hoe optimalisatie kan gebruikt worden om een subset te selecteren uit een multi-experimentele data set. In Deel III worden optimalisatietechnieken toegepast voor de calibratie van een water- en energiebalans model. Deel IV introduceert het begrip onzekerheidspropagatie en toont aan dat ook hierbij optimalisatietechnieken een essentiële rol spelen.

Deel I: Overzicht van optimalisatie

Deel I tracht een beknopte doch op zichzelf berustende inleiding tot het begrip optimalisatie te geven, met de nodige aandacht voor de methoden die verder in dit doctoraat toegepast worden. In Hoofdstuk 2 worden de theoretische concepten in verband met continue optimalisatie en discrete optimalisatie uiteengezet. Verder wordt het begrip "metaheuristiek" geïntroduceerd en worden verschillende metaheuristieken besproken. De nadruk ligt in deze thesis op het verschil tussen metaheuristieken gebaseerd op een populatie van oplossingen versus deze gebaseerd op een enkelvoudige oplossing. De concrete optimalisatiealgoritmen van beide categorieën die in deze thesis worden gebruikt, komen in detail aan bod in Hoofdstuk 3 voor continue optimalisatieproblemen, en in Hoofdstuk 4 voor discrete combinatorische optimalisatieproblemen. Voor de continue optimalisatieproblemen die we in deze thesis behandelen, gebruiken we Gradient Descent gebaseerd op Sequential Quadratic Programming (GD), Particle Swarm Optimization (PSO) en Simplex Simulated Annealing (SIMPSA). Zoals de naam aangeeft is Particle Swarm Optimization een populatiegebaseerde techniek. Voor combinatorische optimalisatieproblemen worden eveneens twee populatiegebaseerde technieken beschreven: Genetische Algoritmen (GA) en Ant Colony Systems (ACO). Elk van deze populatiegebaseerde technieken zijn geïnspireerd op biologische concepten.

Deel II: Subsetselectie uit multi-experimentele data

Het tweede deel van deze thesis behelst het uitwerken van een concreet combinatorisch probleem, meer bepaald het selecteren van een subset uit een multi-experimentele data set. Zoals besproken in Hoofdstuk 5 heeft subsetselectie nut in toepassingen waar een grote hoeveelheid data beschikbaar is. Dit is een recente ontwikkeling die volgt uit de vooruitgang in het verwerven van data alsook uit het bestaan van betere en eenvoudigere manieren om data te delen. Uiteraard leidt de aanwezigheid van een grotere hoeveelheid informatie tot een onmiskenbaar voordeel. Maar dit kan ook nadelen meebrengen: sommige experimentele stappen uit het dataverwerkingsproces nemen vaak veel tijd in beslag en zijn vaak heel duur. Het is dan noodzakelijk deze stappen te beperken tot een welgekozen subset van de volledige dataset. Belangrijk hierbij is dat de subset even informatief is als de totale dataset, wat inhoudt dat alle variabiliteit van de totale data set ook aanwezig moet zijn in de subset. In deze thesis bestuderen we eveneens een bijkomend aspect: we beschouwen een multi-experimentele dataset en verwachten dus dat het aantal geselecteerde samples van een bepaald experiment in de subset ongeveer evenredig is met de grootte van dit experiment in de totale set. Het doel is dat de samples uit de subset uniform verdeeld zijn over dat deel van de parameterruimte waar samples uit de oorspronkelijke dataset voorkomen, zelfs wanneer de oorspronkelijke dataset een hogere dichtheid heeft in bepaalde gebieden. De distributie van die subset zal dus een afgeplatte versie zijn van de distributie van de totale data set. De meer uniforme distributie van de subset impliceert dat deze een hogere variantie heeft, wat toelaat om het subsetselectieprobleem te transformeren naar een optimalisatieprobleem met als objectief het maximaliseren van de variantie van elke variabele voor de geselecteerde subset

Hoofdstuk 6 bespreekt verschillende methoden die kunnen toegepast worden voor de selectie van een optimale subset van samples, zoals het Kennard and Stone algoritme, het Optimizable k-Dissimilarity Selection algoritme en een algoritme gebaseerd op clustering van data. Dit zijn allen specifieke algoritmes die werden ontwikkeld om het standaard subsetselectieprobleem op te lossen. Ze streven dus ook naar het bekomen van een subset met een meer uniforme distributie, maar zonder dat hiervoor een specifieke grootheid wordt geoptimaliseerd. Door de transformatie naar een optimalisatieprobleem is het mogelijk om ook combinatorische optimalisatiealgoritmen toe te passen op het subsetselectieprobleem. Genetische Algoritmen (GA) lijkt hiervoor een ideale kandidaat. Genetische algoritmen werken met een populatie van chromosonen met een binair karakter dat perfect kan gebruikt worden om de selectie van samples aan te duiden. Alvorens Genetische Algoritmen kan toegepast worden op het subsetselectieprobleem, is het nodig om een nieuwe mutatie- en kruisingoperatie te definiëren, zodat het totaal aantal geselecteerde samples constant kan worden gehouden. Om Genetische Algoritmen te vergelijken met een ander biologisch geïnspireerd algoritme, hebben we ervoor gekozen om

het subsetselectieprobleem ook op te lossen met Ant Colony Systems (ACO). Dit algoritme werd ontwikkeld voor een beperkte klasse van subsetselectieproblemen, de zogenaamde knapzakproblemen. Net zoals bij de chromosonen van GA, wordt ook bij ACO de selectie van samples voorgesteld aan de hand van een reeks bits. Deze representatie gaat echter gepaard met een bias. Bij de constructie van mogelijke subsets, werd het originele ACO algoritme aangepast om dit probleem op te lossen. Tevens hebben we gekozen voor een parallelle implementatie van dit algoritme, aangezien ACO computationeel veeleisend is wanneer het op dit type subsetselectieproblemen wordt toegepast. Een groot voordeel van de herformulering van het subsetselectieprobleem als een optimalisatieprobleem is dat aan de hand van de objectieffunctie het multi-experimentele aspect van de data in rekening kan gebracht worden. De objectieffunctie kan de selectie van samples zo sturen dat een aantal samples uit elk experiment aanwezig zijn in de resulterende subset. Zonder het aantal samples van elk experiment expliciet te moeten bepalen, wordt automatisch bekomen dat het aantal samples van elk experiment in de subset ongeveer proportioneel is met de grootte van elk experiment in de totale dataset. Bescheiden afwijkingen zijn mogelijk wanneer bepaalde experimenten relatief meer samples nodig hebben voor een even informatieve representatie in een subset dan andere experimenten. Bij de klassieke subsetselectiealgoritmen kan het multi-experimentele aspect van de data enkel in rekening gebracht worden door het aantal geselecteerde samples van elk experiment op voorhand vast te leggen. De meest voor de hand liggende manier om dit te doen, is het onafhankelijk beschouwen van de verschillende experimenten. Afwijkingen worden dan onmogelijk zodat we verwachten dat een minder optimale subset bekomen wordt.

De verschillende algoritmen worden toegepast op een gevalstudie waarbij de dataset bestaat uit de concentratie van 45 vetzuren in 1033 melkstalen. Deze melkstalen zijn afkomstig uit 6 verschillende experimenten. Het objectief is het selecteren van een subset van melkstalen op een manier dat deze subset informatief is voor de totale data set, en waarbij elk van de experimenten voldoende vertegenwoordigd is. De beste resultaten werden verkregen met Genetisch Algoritmen in combinatie met onze voorgestelde objectieffunctie. De conventionele technieken produceren minder goede resultaten. Bovendien blijkt het vooraf vastleggen van het aantal geselecteerde samples per experiment een groot nadeel. We testten zowel het gebruik van een gelijk aantal samples voor elk experiment als een aantal samples proportioneel aan de grootte van het experiment. De resultaten waren in beide gevallen slechter dan de resultaten van GA en ACO, die zelf dynamisch de optimale waarde voor het aantal samples per experiment bepalen. Hieruit kunnen we concluderen dat het niet vanzelfsprekend is om manueel een juist aantal samples per experiment te kiezen. De resultaten met het aangepaste ACO algoritme waren veel beter dan de resultaten verkregen met het originele algoritme en met de conventionele technieken, wat erop duidt dat het verwijderen van de bias het algoritme sterk verbetert. De resultaten verkregen met GA zijn significant beter dan de resultaten met het ACO algoritme.

Bovendien blijft het ACO algoritme, ondanks de parallellisatie, computationeel heel veeleisend.

Deel III: Kalibratie van een water- en energiebalans model

In het derde deel van dit doctoraat wordt optimalisatie aangewend bij de kalibratie van een complex wiskundig model. We spitsen ons hierbij toe op hydrologisch modellen, die vaak werken met een complexe verzameling van ingangsvariabelen en uitgangsvariabelen bestaande uit meteorologische data gecombineerd met een groot aantal topografische parameters, landbedekkingsparameters en bodemparameters. Zoals beschreven wordt in Hoofdstuk 9 hangen dergelijke modellen bovendien af van een aantal modelparameters die moeten worden geschat aan de hand van gemeten data, alvorens het model kan worden toegepast om toekomstige data te voorspellen. Deze schattingen zijn essentieel opdat de geconstrueerde modellen nuttig zouden kunnen worden toegepast. In het ideale geval zouden deze parameters verkregen worden door observaties ter plaatse. In de praktijk is dit meestal onmogelijk omwille van verschillende redenen: een verschil in ruimtelijke schaal tussen de meting van de modelparameters en de toepassing van het model, de niet-fysische betekenis van een aantal parameters, een inconsistentie tussen de modelparameters en de ter plaatse geobserveerde parameters als gevolg van een simplificatie van de realiteit in het model of door het hoge aantal van model parameters in bijvoorbeeld ruimtelijk gedistribueerde modellen. Daarom worden de parameters numeriek geschat met behulp van gemeten data, waarbij getracht wordt de uitgang die het model voorspelt voor de gemeten ingangsvariabelen zo goed mogelijk in overeenstemming te krijgen met de gemeten waarden voor de uitgangsvariabelen.

In Hoofdstuk 10 bespreken we de methoden die in deze thesis worden gebruikt voor de kalibratie van dergelijk hydrologisch model. Opnieuw stellen we ons tot doel om een probleemspecifieke methode te vergelijken met een algemeen toepasbaar optimalisatiealgoritme. Een specifieke kalibratiemethode voor de schatting van parameters in een model is de zogenaamde 'Multistart Weight-Adaptive Recursive Parameter Estimation' (MWARPE) methode. Bij de MWARPE methode worden alle uitgangsvariabelen expliciet in rekening gebracht tijdens het updaten van de parameters. Deze methode maakt op iteratieve wijze gebruik van de lineaire recursieve filtervergelijkingen in een Monte-Carlo structuur. Een voordeel is dat deze methode automatisch rekening houdt met de typische grootteorde van de verschillende uitgangsvariabelen en het dus geen probleem vormt indien de verschillende modeluitgangen sterk verschillende grootteordes hebben. Het belangrijkste nadeel van deze methode is echter de computationele kost. MWARPE vereist de inversie van matrices met een dimensie gelijk aan het totaal aantal observaties. Deze factor stelt een grote beperking op de hoeveelheid trainingsdata die in de kali-

bratie van het model kan worden gebruikt. Om het kalibratieprobleem op te lossen met een algemeen optimalisatiealgoritme kunnen verschillende objectieffuncties worden geconstrueerd. Allen komen ze neer op het minimaliseren van het verschil tussen de modeluitgangen en de geobserveerde waarde voor de uitgangsvariabelen. Indien zou getracht worden om de gemiddelde kwadratische fout (RMSE, *root mean square error*) voor elk van de verschillende uitgangsvariabelen afzonderlijk te optimaliseren, dient een Pareto-front opgesteld te worden. In dat geval wordt de verschillende grootteorde van de verschillende uitgangsvariabelen eveneens automatisch in rekening gebracht. Echter, voor een groot aantal uitgangsvariabelen leidt dit tot een hoog dimensionaal Pareto-front, wat vanuit praktisch oogpunt heel complex is. We besloten daarom om de verschillende objectieffuncties samen te voegen tot één objectieffunctie door hun waarden op te tellen. Hierbij moeten we wel rekening houden met de mogelijke verschillen in grootteorde tussen de verschillende uitgangsvariabelen, zodat de verschillende RMSE waarden correct genormaliseerd worden. Als bijbehorende optimalisatiealgoritme wordt gekozen voor Particle Swarm Optimization (PSO), dat heel eenvoudig kan worden toegepast op het gegeven probleem.

In Hoofdstuk 11 worden beide methoden met elkaar vergeleken aan de hand van een gevalstudie met een eenvoudig hydrologisch model dat 11 modelparameters bevat. Dit model wordt kort geïntroduceerd. De uitgang van het model bestaat uit 8 variabelen, namelijk de energiebalanstermen en het vochtgehalte van de bodem op 4 verschillende dieptes, waarvoor uurlijkse observaties beschikbaar zijn. Voor de kalibratie van het model hebben we data onderverdeeld in 2 periodes die elk 50 % van de beschikbare data bevatten. De beschikbare data in deze periodes zullen afwisselend als trainingsdata en validatiedata gebruikt worden. Omwille van de hoge dimensionaliteit van de te inverteren matrix in de MWARPE methode is het niet mogelijk om alle uurlijkse observaties te gebruiken en worden enkel de observaties om 1 uur en 13 uur gebruikt voor de termen van de energiebalans en observaties om 12 uur voor het vochtgehalte van de bodem. Ook al heeft PSO niet hetzelfde nadeel, toch wordt in de eerste plaats ook de gereduceerde dataset gebruikt. Dit maakt een eerlijke vergelijking mogelijk tussen de optimalisatiecapaciteit van beide methoden. Nadien wordt de meerwaarde van de volledige dataset bestudeerd met PSO. De resultaten tonen aan dat in meeste gevallen de gemiddelde RMSE voor de trainingsperiode significant lager is wanneer MWARPE is gebruikt. Echter, dit significant verschil verdwijnt in de meeste gevallen wanneer gekeken wordt naar de RMSE in de corresponderende validatieperiode. Dit leidt tot de conclusie dat beide methoden even bekwaam zijn voor het kalibreren van het hydrologisch model. Wanneer PSO de volledige dataset kan gebruiken, worden iets betere, maar niet significant betere, RMSE waarden bekomen. Beide methoden resulteren dus in min of meer vergelijkbare resultaten. Als we de zwaardere computationele kost van MWARPE in rekening brengen, dan kan wel besloten worden dat PSO in de praktijk te verkiezen valt.

Deel IV: Onzekerheidspropagatie

In de praktijk faalt de aanname dat alle parameters en variabelen in een model overeenstemmen met fysische grootheden die in theorie exact kunnen gemeten worden. Bepaalde parameters brengen steeds een inherente onzekerheid met zich mee, zodat veel ingenieurstoepassingen baat hebben bij een correcte beschrijving van onzekerheid en de propagatie ervan doorheen een model. In Hoofdstuk 13 beschrijven we hoe onzekerheid kan opgedeeld worden in twee groepen, met name aleatorische en epistemische onzekerheid. Waar aleatorische of statistische onzekerheid verbonden is met de natuurlijke willekeur gerelateerd aan een bepaald proces en kan beschreven worden met behulp van probabiliteitstheorie, is epistemische of systematische onzekerheid gerelateerd aan de beperkte hoeveelheid kennis die aanwezig is over een bepaald systeem. De aannames en benaderingen die een model maakt als gevolg van dergelijke incomplete kennis van de realiteit, dienen behandeld te worden als onzekere of vage variabelen. In plaats van een unieke waarde of een scherpe verzameling van waarden wordt er aan de variabele een vaagverzameling van waarden toegekend. De mogelijkheid dat de variabele die bepaalde waarde aanneemt wordt beschreven door de lidmaatschapsgraad van die waarde tot de vage verzameling. In dit deel van de thesis richten we ons op de propagatie van vaagheid doorheen wiskundige modellen.

Het objectief van dit deel van dit doctoraat is het ontwikkelen van een computationeel efficiënte 'Fuzzy Calculator' die in staat is om vage ingangsvariabelen met gegeven lidmaatschapsfuncties doorheen een willekeurige functie of model te propageren en de lidmaatschapsfunctie van de vage uitgang te bepalen. De noodzakelijke methodologie wordt geïntroduceerd in Hoofdstuk 14. De theoretische grondslagen voor de propagatie van onzekerheid van niet-interactieve vage ingangsvariabelen zijn vervat in het extensiebeginsel van Zadeh [120]. Een rechtstreekse toepassing van het extensiebeginsel is echter te complex in de meeste gevallen. Nguyen ontwikkelde daarom een meer praktische aanpak gebaseerd op α-sneden voor niet-interactieve vage ingangsvariabelen [121]. Een α-snede is gedefinieerd als de scherpe verzameling van elementen die behoren tot de vage variabele met een minimale lidmaatschapsgraad α. Deze aanpak is enkel mogelijk in het geval van bovensemicontinue, convexe vaagverzamelingen met een compacte drager. Variabelen die aan deze eigenschappen voldoen worden vaagintervallen genoemd. In het geval dat alle ingangsvariabelen van de functie vaagintervallen zijn, dan is de uitgangsvariabele eveneens een vaaginterval. De α-sneden van de uitgang worden begrensd door het minimum en maximum van de functie in de hyperrechthoek, gevormd door de α-sneden van de vaagintervallen voor de ingangsvariabelen. Het probleem is dus getransformeerd in een optimalisatieprobleem waarbij we het minimum en het maximum van een functie in een bepaald gebied moeten bepalen. Deze praktische aanpak werd uitgebreid door Fullér en Keresztfalvi [123] voor het geval van interactieve vaagintervallen. Hierbij wordt aangenomen dat de interactiviteit

wordt gemodelleerd aan de hand van triangulaire normen (t-normen), een begrip dat eveneens wordt geïntroduceerd in Hoofdstuk 14. We definiëren vier basis t-normen, met name de minimum t-norm, de product t-norm, de Łukasiewicz t-norm en de drastic t-norm. De minimum t-norm komt overeen met niet-interactiviteit. Bij interactieve variabelen komen α-sneden niet langer overeen met een hyperrechthoek maar met een meer algemeen gebied waarbinnen het minimum en het maximum van de functie moeten worden bepaald.

In het geval van monotone continue functies van niet-interactieve vage ingangsvariabelen bestaan verschillende praktische implementaties van het extensiebeginsel gebaseerd op α-sneden in de literatuur. Essentieel voor de werking van al deze methoden is dat in het geval van een monotone functie, gedefinieerd op een hyperrechthoek, de extreme waarden van deze functie gelokaliseerd zijn op de hoekpunten van deze hyperrechthoek. In het geval van niet-monotone functies moeten het minimum en het maximum van de functie, die samen de α-sneden van de uitgangsvariabelen bepalen, gezocht worden binnen de hyperrechthoek. Bij interactieve variabelen wordt het zoekgebied vervormd, wat in rekening kan gebracht worden door niet-lineaire bindingsvoorwaarden toe te voegen aan het optimalisatieprobleem. In het meest algemene geval moeten we dus een gebonden optimalisatieprobleem oplossen. Voor de ontwikkeling van deze 'Fuzzy Calculator' werden vier optimalisatiealgoritmen vergeleken, meer bepaald Gradient Descent gebaseerd op Sequential Quadratic Programming (GD), Simplex-Simulated Annealing (SIMPSA), Particle Swarm Optimization (PSO) en Particle Swarm Optimization gecombineerd met Gradient Descent gebaseerd op Sequential Quadratic Programming (PSO_GD). Voor het aantal α-sneden werden twee methodologieën vooropgesteld: ofwel wordt gewerkt met een vast aantal sneden, ofwel met een variabel aantal α-sneden. In het laatste geval wordt gestart met een klein aantal α-sneden en worden nieuwe sneden aangemaakt zolang niet voldaan is aan een convergentiecriterium op basis van lineaire interpolatie. We hebben zowel een niet parallelle als een parallelle implementatie de 'Fuzzy Calculator' ontwikkeld. De parallelle implementatie is enkel belangrijk wanneer PSO gebruikt wordt als optimalisatiealgoritme. Aangezien het zoekgebied voor de verschillende optimalisatieproblemen corresponderend met de verschillende α-sneden grotendeels overlappen, kunnen de verschillende PSO zwermen van elkaar de locatie van mogelijke extrema leren indien ze in staat zijn om met elkaar te communiceren. Het doel is dus om via communicatie de extrema meer nauwkeurig en sneller te lokaliseren. Tot slot controleren we ook of er geen inconsistenties optreden in de uiteindelijke lidmaatschapsfunctie van de uitgang: gegeven de definitie van de lidmaatschapsfunctie is het onmogelijk dat het minimum (maximum) van een hoger gelegen α-snede kleiner (groter) is dan het minimum (maximum) van een lager gelegen α-snede. We stellen twee mogelijke technieken voor om dergelijke inconsistenties te corrigeren indien ze zich toch voordoen. Een eerste mogelijkheid is om het minimum (maximum) van de betreffende α-snede gelijk te stellen aan het minimum (maximum) van de hoger gelegen α-snede. Een andere oplossing is om

het oude resultaat weg te gooien en het minimum (maximum) van de betreffende α-snede opnieuw te berekenen.

Om na te gaan welke configuratie van de 'Fuzzy Calculator' het best presteert, hebben we deze eerst toegepast op 9 continue testfuncties in Hoofdstuk 15. Initieel hebben we ons hierbij beperkt tot niet-interactieve ingangsvariabelen. Om de verschillende optimalisatiealgoritmen met elkaar te vergelijken werd in een eerste testconfiguratie het aantal α-sneden vast gezet op 11. Als nauwkeurigheidsmaat werd gekozen voor de oppervlakte onder de lidmaatschapsfunctie van het vaaginterval voor de uitgangsvariabele. Zowel voor de parallelle als voor de niet parallelle implementatie resulteert de 'Fuzzy Calculator' die gebruik maakt van PSO_GD met een populatiegrootte van 20 partikels als optimalisatiealgoritme tot significant betere resultaten dan de 'Fuzzy Calculator' gebruik makend van de andere optimalisatiealgoritmen. De resultaten tonen ook aan dat communicatie, met een communicatiefrequentie om de 5 iteraties, in de parallelle versie van de 'Fuzzy Calculator' een significant voordeel oplevert. Het corresponderende aantal functie-evaluaties is echter significant hoger voor de parallelle 'Fuzzy Calculator', zodat een betere lokalisatie van de extrema wel gepaard gaat met een hoger aantal iteraties. In een tweede testconfiguratie werd gewerkt met een variabel aantal α-sneden, startende van 3 α-sneden. De lidmaatschapsfunctie opgebouwd door de 'Fuzzy Calculator' startende met 3 α-sneden is significant nauwkeuriger dan wanneer het aantal α-sneden gefixeerd is op 11. Uiteraard gaat dit opnieuw gepaard met een significant hoger aantal α-sneden. Tot slot werden de twee voorstellen voor de correctie van inconsistenties tussen de α-sneden vergeleken. Er is geen verschil aanwezig in nauwkeurigheid van de lidmaatschapsfuncties opgebouwd in het geval van deze twee aanpakken. Het aantal functie-evaluaties is wel significant lager wanneer de inconsistente α-sneden opnieuw worden berekend. We kunnen dus besluiten dat de 'Fuzzy Calculator' met een variabel aantal α-sneden, met PSO_GD met een populatiegrootte van 20 en communicatie elke 5 iteraties en met het herberekenen van inconsistente α-sneden tot de meest nauwkeurige lidmaatschapsfunctie voor de vage uitgangsvariabele leidt. Deze 'Fuzzy Calculator' werd dan ook toegepast op de 9 testfuncties wanneer interactiviteit tussen de ingangsvariabelen wordt ingeschakeld. Hierbij hebben we ons beperkt tot interactiviteit beschreven door de vier basis t-normen. Zoals kon verwacht worden is de oppervlakte onder de lidmaatschapsfunctie het grootst voor de minimum t-norm, gevolgd door de product norm, gevolgd door de Łukasiewicz t-norm and tot slot de drastic t-norm. De toevoeging van niet-lineaire bindingsvoorwaarden aan het optimalisatieprobleem heeft geen invloed op het aantal functie-evaluaties.

Om de praktische toepasbaarheid van de 'Fuzzy Calculator' na te gaan hebben we deze ook toegepast op een meer complex en praktisch model in Hoofdstuk 16. Hiervoor hebben we gekozen voor het 'Integral Equation Model' (IEM), dat aan de hand van informatie over de achterwaartse verstrooiingscoëfficiënt en de ruwheidsparameters van een medium in staat is om het vochtgehalte van dit medium te

berekenen. Terwijl de achterwaartse verstrooiingscoëfficiënt kan bekomen worden uit RADAR beelden, is het bepalen van de ruwheidsparameters van het medium veel complexer, wat resulteert in onzekerheid omtrent de precieze waarde. Deze onzekerheid kunnen we doorheen het IEM propageren aan de hand van de 'Fuzzy Calculator'. Onze resultaten werden vergeleken met een vroegere studie van de onzekerheidspropagatie doorheen dit model door Vernieuwe *et. al.* [141]. In tegenstelling tot ons algoritme, zoekt het algoritme dat gebruikt werd in Vernieuwe *et. al.* enkel op de rand van de zoekruimte. Voor een niet monotoon model verwachten we dus grote verschillen en een betere prestatie voor onze 'Fuzzy Calculator'. Aangezien het verschil in oppervlakte onder de lidmaatschapsfuncties tussen deze twee methoden heel klein is, hebben we waarschijnlijk te maken met een model met geringe niet-monotoniteit. We kunnen dus wel besluiten dat onze 'Fuzzy Calculator' eveneens goed presteert wanneer de optima zich op de rand bevinden, en wegens zijn algemeenheid dus heel ruim inzetbaar is.

Besluiten en vooruitzichten

De verschillende delen van dit doctoraat laten toe te besluiten dat de heuristische optimalisatietechnieken de competitie aankunnen met probleemspecifieke algoritmen. De heuristische optimalisatietechnieken hebben daarbij het grote voordeel van algemeenheid, zodat ze vaak op een ruimere klasse van problemen toepasbaar zijn, en vallen bovendien vaak vrij eenvoudig te implementeren.

In deze doctoraatsthesis hebben we ons geconcentreerd op problemen die één objectief nastreven. Echter, in vele toepassingen bestaan er problemen waarbij meerdere objectieven worden nagestreefd. Voorbeelden hiervan zijn onder andere het grote aantal objectieven in bosbeheer (houtproductie, biodiversiteit, waterkwaliteit, recreatie, enzovoort), de aanwezigheid van meerdere objectieven in landbouw (maximale inkomsten met een minimum aan irrigatie, meststoffen en aantal werkmensen), de optimalisatie van de verwijdering van meerdere componenten in waterzuiveringinstallaties, enzovoort. Er bestaan verschillende manieren om een probleem die meerdere objectieven nastreven te optimaliseren. Eén manier is samenvoegen van de verschillende objectieffuncties tot één objectieffunctie, zoals we reeds gedaan hebben voor de kalibratie van het hydrologisch model (Deel III). Een meer complete aanpak is de constructie van een Pareto-front. Hiervoor bestaan er verschillende algoritmen, maar wanneer vele objectieven aanwezig zijn wordt dit toch heel complex. Een interessant toekomstig onderzoeksproject is dus om ook in het geval van problemen waarbij meerdere objectieven worden nagestreefd, een vergelijking te maken tussen probleemspecifieke methoden en metaheuristische algoritmen. Ondanks de hogere wiskundige complexiteit, kan het rekening houden met de meerdere objectieven ook tot een verbetering van de resultaten leiden voor de problemen die in dit doctoraat met behulp van n objectief werden bestudeerd.

Zo kan ook in het subsetselectieprobleem (Deel II) gebruik gemaakt worden van een tweede objectief, meer bepaald de afwijking van het gemiddelde, welke we in dit doctoraat slechts gebruikt hebben als verificatie voor de geselecteerde subset.

Bibliography

[1] L. Pauling. *The Nature of the Chemical Bond.* Cornell University Press, 1960.

[2] A. Neumaier. Global optimization and constraint satisfaction. In *Proceedings of GICOLAG workshop (of the research project Global Optimization, Integrating Convexity, Optimization, Logic Programming and Computational Algebraic Geometry)*, 2006.

[3] A. P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence.* John Wiley & Sons Ltd., 2006.

[4] A. Antoniou and W.S. Lu. *Practical optimization: algorithms and engineering applications.* Springer Science+Business Media, 2007.

[5] A.D. Belegundu and T.R. Chandrupatla. *Optimization Concepts and Application in Engineering.* Cambridge University Press, 1999.

[6] A. Kwok and D. Norrie. Intelligent agent systems for manufacturing applications. *Journal of Intelligent Manufacturing*, 4:285–293, 1993.

[7] W. Shen, Q. Hao, H. Yoon, and D. Norrie. Applications of agent-based systems in intelligent manufacturing: an updated review. *Advanced Engineering Informatics*, 2:415–431, 2006.

[8] P. Brandimarte. *Numerical Methods in Finance and Economics: A MATLAB-based introduction.* John Wiley & Sons, Inc., 2006.

[9] K. Chen and Y.L. Yao. Process optimisation in pulsed laser micromachining with application in medical device manufacturing. *International Journal of Advanced Manufacturing Technology*, 4:243–249, 2000.

[10] Z.W. Geem and J.Y. Choi. Music composition using harmony search algorithm. In *Lecture Notes in Computer Science*, 2007.

[11] A.K. Hartmann and H. Rieger. *Optimization Algorithms in Physics.* Wiley-VCH Verlag Berlin, 2002.

[12] C.A. Floudas and P.M. Pardalos. *Optimization in Computational Chemistry and Molecular Biology-Local and Global Approaches.* Kluwer Academic Publishers, 2000.

[13] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer Verlag, 1999.

[14] A. Schrijver. *Combinatorial Optimization Volume A.* Springer, 2003.

[15] A. Schrijver. *Combinatorial Optimization Volume B.* Springer, 2003.

[16] A. Schrijver. *Combinatorial Optimization Volume C.* Springer, 2003.

[17] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co, 1979.

[18] M. Dorigo and T. Stützle. *Ant Colony Optimization.* Massachusetts Institue of Technology, 2004.

[19] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: algorithms and complexity.* Dover Publications, Inc., 1998.

[20] E. Talbi. *Metaheuristics: From Design to Implementation.* John Wiley & Sons Inc., 2009.

[21] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Computing Surveys*, 35:268–308, 2003.

[22] T. Stützle. *Local Search Algorithms for Combinatorial Problems–Analysis, Algorithms and New Applications.* Infix, Sankt Augustin, 1999.

[23] B. M. R. Donckels. *Optimal experimental design to discriminate among rival dynamic mathematical models.* PhD thesis, Ghent University, 2009.

[24] B. M. R. Donckels, D. J. W. De Pauw, P. A. Vanrolleghem, and B. De Baets. A kernel-based method to determine optimal sampling times for the simultaneous estimation of the parameters of rival mathematical models. *Journal of Computational Chemistry*, 30:2064–2077, 2009.

[25] B. M. R. Donckels, D. J. W. De Pauw, P. A. Vanrolleghem, and B. De Baets. An ideal point method for the design of compromise experiments to simuly estimate the parameters of several rival models. *Chemical Engineering Science*, 65:1705–1719, 2010.

[26] Boggs P.T. and Tolle J.W. Sequential quadratic programming. *Acta Numerica*, 4:1–52, 1995.

[27] J. Kennedy and R. C. Eberhart. *Swarm Intelligence.* Morgan Kaufmann Publishers, 2001.

[28] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. IEEE International Conference on Artificial Neural Networks*, pages 1942–1948, 1995.

[29] C.W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21:25–34, 1987.

[30] F. Heppner and U. Grenander. *A stochastic nonlinear model for coordinated bird flocks.* AAAS Publications, 1990.

[31] M. Clerc. *Particle Swarm Optimization.* ISTE-Ltd, 2006.

[32] Y. del Valle. Particle swarm optimization: Basic concepts, variants and applications in power systems. *IEEE Transactions on evolutionary computation*, 12:171–195, 2008.

[33] X. Hu. Particle Swarm Optimization. In *Tutorial of the IEEE Swarm Intelligence Symposium*, 2006.

[34] M. F. Cardoso, R. Salcedo, and S. Feyo de Azevedo. The simplex-simulated annealing approach to continuous non-linear optimization. *Computers and Chemical Engineering*, 20:1065–1080, 1996.

[35] J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[36] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.

[37] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of chemical physics*, 21:1087–1092, 1953.

[38] E. H. L. Aarst and P. J. M. van Laarhoven. Statistical cooling: a general approach to combinatorial optimization problems. *Philips Journal of Research*, 40:193–226, 1985.

[39] W. H. Press and S. A. Teukolsky. Simulated annealing optimization over continuous spaces. *Computational Physics*, 5:426–430, 1991.

[40] D. E. Goldberg. *Genetic Algorithms in Search, Optimization & Machine Learning.* Kluwer Academic Publishers, 1989.

[41] L. Davis and M. Steenstrup. *Genetic Algorithms and Simulated Annealing.* Morgan Kaufmann, 1987.

[42] Z. Michalewicz. A perspective on evolutionary computation. In *Selected papers from the AI'93 and AI'94 Workshops on Evolutionary Computation, Process in Evolutionary Computation*, 1995.

[43] P.-P. Grassé. La reconstruction du nid et les coordinations inter-individuelles chez Belicositermes natalensis et Cubitermes sp. La théorie de la Stigmergie : Essai d'interprétation du comportement des termites constructeurs. *Insectes Sociaux*, 6:41–81, 1959.

[44] P.-P. Grassé. *Les insectes dans leur univers.* 1946.

[45] J.-L. Deneubourg, S. Aron, S. Goss, and J.-M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3:159–168, 1990.

[46] M. Dorigo, V. Manniezzo, and A. Colorni. Ant systems: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:29–41, 1996.

[47] M. Dorigo and L.M. Gambardella. Ant colony systems: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1:53–66, 1997.

[48] J. Montgomery, M. Randall, and T. Hendtlass. Solution bias in ant colony optimization: Lessons for selecting pheromone models. *Computers & Operations Research*, 35:2728–2749, 2008.

[49] G. Leguizamon and Z. Michalewicz. A new version of ant system for subset problems. In *IEEE Congress on Evolutionary Computation*, 1999.

[50] S. Fidanova. Ant colony optimization for multiple knapsack problems and heuristic model. *Lecture Notes in Computer Science*, 3401:290–287, 2005.

[51] R. Hinterding. Mapping, order-independent genes and the knapsack problem. In *1st IEEE International Conference on Evolutionary Computation*, 1994.

[52] J. Montgomery, M. Randall, and T. Hendtlass. Search bias in constructive metaheuristics and implications for ant colony optimization. In *ANTS 2004, Ant Colony Optimization and Swarm Intelligence*, 2004.

[53] M. Kong, P. Tian, and Y. Kao. A new ant colony optimization algorithm for the multidimensional knapsack. *Computers & Operations Research*, 35:2672–2683, 2008.

[54] S. Khuri, T. Back, and J. Heitkotter. The zero/one multiple knapsack problem and genetic algorithms. In *ACM Symposium on Applied Computing*, 1994.

[55] J. Verwaeren, K. Scheerlinck, and B. De Baets. Countering the negative search bias of ant colony optimization in subset selection problems. *Submitted to Internation Journal of Operational Research*, 2011.

[56] M. Daszykowski, B. Walczak, and D. L. Massart. Representative subset selection. *Analytica Chimica Acta*, 468:91–103, 2002.

[57] R. W. Kennard and L. A. Stone. Computer aided design of experiments. *Technometrics*, 11:137–148, 1969.

[58] R. D. Clark. Optisim: An extended dissimilarity selection method for finding diverse representative subsets. *Journal of Chemical Information and Modeling*, 6:1181–1188, 1997.

[59] Y. Tominaga. Representative subset selection using genetic algorithms. *Chemometrics and Intelligent Laboratory Systems*, 43:157–163, 1998.

[60] J. R. Cano, F. Herrera, and M. Lozano. Using evolutionary algorithms as instance selection for data reduction in KDD: An experimental study. *IEEE Transactions on Evolutionary Computation*, 7:561–575, 2003.

[61] J. R. Cano, F. Herrera, and M. Lozano. Evolutionary stratified training set selection for extracting classification rules with trade-off precision-interpretability. *Data and Knowledge Engineering*, 60:90–108, 2007.

[62] R. D. Snee. Validation of regression models: methods and examples. *Technometrics*, 19:415–428, 1977.

[63] L. Aristidis, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. *Pattern Recognition*, 36:451–461, 2003.

[64] S. Torquato and F. H. Stillinger. New conjectural lower bounds on the optimal density of sphere packings. *Experimental Mathematics*, 15:307–331, 2006.

[65] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[66] S. Novkovic and D. Šverko. The minimal deceptive problem revisited: the role of 'genetic waste'. *Computers & Operations Research*, 25:895–911, 1998.

[67] Silva L.M. and Buyya R. *High Performance Cluster Computing*, chapter Parallel Programming Models and Paradigms, pages 4–27. Prentice Hall PTR, 1999.

[68] B. Vlaeminck, C. Dufour, A. M. van Vuuren, A. R. J. Cabrita, R. J. Dewhurst, D. Demeyer, and V. Fievez. Use of odd and branched chain fatty acids in rumen contents and milk as a potential microbial marker. *Journal of Dairy Science*, 88:1031–1042, 2005.

[69] B. Vlaeminck, V. Fievez, A. R. J. Cabrita, A. J. M. Fonseca, and R. J. Dewhurst. Factors affecting odd and branched-chain fatty acids in milk: a review. *Animal Feed Science and Technology*, 131:389–417, 2006.

[70] Y. N. T. Van Haelst, A. Beeckman, A. T. M. Van Knegsel, and V. Fievez. Elevated concentrations of oleic acid and long chain fatty acids in milk fat of multiparous subclinical ketotic cows. *Journal of Dairy Science*, 91:4683–4686, 2008.

[71] B. Vlaeminck, J. Harynuk, V. Fievez, and P. Marriott. Comprehensive two-dimensional gas chromatography for improved separation of fatty acids in milk. *European Journal on Lipid Science and Technology*, 109:757–766, 2007.

[72] J. K. G. Kramer, M. Hernandez, C. Cruz-Hernandez, J. Kraft, and M. E. R. Dugan. Combining results of two gc separations partly achieves determination

of all cis and trans 16:1, 18:1, 18:2 and 18:3 except cla isomers of milk fat as demonstrated using ag-ion spe fractionation. *Lipids*, 43:259–273, 2008.

[73] G. Van Ranst, V. Fievez, M. Vandewalle, C. Van Waes, J. De Riek, and E. Van Bockstaele. Influence of damaging and wilting red clover on lipid metabolism during ensiling and in vitro rumen incubation. Submitted (Journal of Dairy Science), 2009.

[74] M. Craninx, A. Steen, H. Van Laar, T. Van Nespen, J. Martin-Tereso, B. De Baets, and V. Fievez. Effect of lactation stage on the odd- and branched-chain milk fatty acids of dairy cattle under grazing and indoor conditions. *Journal of Dairy Science*, 91:2662–2677, 2008.

[75] E. Colman, W. B. Fokkink, M. Craninx, J. R. Newbold, and V. Fievez. Effect of induction of sub-acute ruminal acidosis (SARA) on milk fat profile and rumen parameters. *Submitted (Journal of Dairy Science)*, 2010.

[76] C. Boeckaert, B. Vlaeminck, J. Dijkstra, A. Issa-Zacharia, T. Van Nespen, W. Van Straalen, and V. Fievez. Effect of dietary starch or micro algae supplementation on rumen fermentation and milk fatty acid composition. *Journal of Dairy Science*, 91:4714–4727, 2008.

[77] B. Vlaeminck, M. Hostens, G. Opsomer, and V. Fievez. Milk fat parameters show delayed response to dietary micro algae supplementation in early lactation. In *International Symposium on Ruminant Physiology*, Clermont-Ferrand, France, September 2009.

[78] J. Neter, M. H. Kutner, Ch.J. Nachtsheim, and W. Wasserman. *Applied Linear Statistical Models*. McGraw-Hill/Irwin, 2004.

[79] J. A. Santanello Jr., C. D. Peters-Lidard, M. E. Garcia, D. M. Mocko, M. A. Tischler, M. S. Moran, and D. P. Thoma. Using remotely-sensed estimates of soil moisture to infer soil texture and hydraulic properties across a semi-arid watershed. *Remote Sensing of Environment*, 110(1):79–97, 2007.

[80] A. V. M. Ines and B. P. Mohanty. Near-Surface Soil Moisture Assimilation for Quantifying Effective Soil Hydraulic Properties Using Genetic Algorithms: ii. With air-borne remote sensing during SGP97 and SMEX02. *Water Resources Research.*, 45(1), 2009.

[81] A. V. M. Ines and B. P. Mohanty. Near-Surface Soil Moisture Assimilation for Quantifying Effective Soil Hydraulic Properties Using Genetic Algorithms: i. Conceptual modeling. *Water Resources Research.*, 44(6), 2008.

[82] A. V. M. Ines and B. P. Mohanty. Parameter conditioning with a noisy Monte Carlo genetic algorithm for estimating effective soil hydraulic properties from space. *Water Resources Research.*, 44(8), 2008.

[83] V. R. N. Pauwels, A. Balenzano, G. Satalino, H. Skriver, N. E. C. Verhoest, and F. Mattia. Optimization of soil hydraulic model parameters using Synthetic Aperture Radar data: an integrated multidisciplinary approach. *IEEE Trans. Geosci. Remote Sensing*, 47(2):455–467, 2009.

[84] H. L. Fang, S. L. Liang, and A. Kuusk. Retrieving leaf area index using a genetic algorithm with a canopy radiative transfer model. *Remote Sensing of Environment*, 85(3):257–270, 2003.

[85] M. Meroni, R. Colombo, and C. Panigada. Inversion of a radiative transfer model with hyperspectral observations for LAI mapping in poplar plantations. *Remote Sensing of Environment*, 92(2):195–206, 2004.

[86] R. Casa and H. G. Jones. LAI retrieval from multiangular image classification and inversion of a ray tracing model. *Remote Sensing of Environment*, 98(4):414–428, 2005.

[87] J. Doherty. *PEST-ASP Users' manual*. Watermark Numerical Computing, Brisbane, Australia, 2001.

[88] Q. Y. Duan, S. Sorooshian, and V. K. Gupta. Optimal use of the SCE-UA global optimization method for calibrating watershed models. *Journal of Hydrology*, 158(3-4):265–284, 1994.

[89] J. A. Vrugt, H. V. Gupta, W. Bouten, and S. Soorooshian. A shuffled complex evolution Metropolis algorithm for optimization and uncertainty assessment of hydrologic models. *Water Resources Research.*, 39(8), 2003.

[90] P. Reed, B. S. Minsker, and D. E. Goldberg. Simplifying multiobjective optimization: An automated design methodology for the nondominated sorted genetic algorithm-II. *Water Resources Research.*, 39(7), 2003.

[91] H. V. Gupta, L. A. Bastidas, S. Sorooshian, W. J. Shuttleworth, and Z. L. Wang. Parameter estimation of a land surface scheme using multicriteria methods. *Journal of Geophysical Research*, 104(D16):19,491–19,503, 1999.

[92] P. R. Houser, H. V. Gupta, W. J. Shuttleworth, and J. S. Famiglietti. Multiobjective calibration and sensitivity of a distributed land surface water and energy balance model. *Journal of Geophysical Research*, 106(D24):33,421–33,433, 2001.

[93] W. T. Crow, E. F. Wood, and M. Pan. Multiobjective calibration of land surface model evapotranspiration predictions using streamflow observations and spaceborne surface radiometric temperature retrievals. *Journal of Geophysical Research*, 108(D23), 2003.

[94] H. Madsen. Parameter estimation in distributed hydrological catchment modelling using automatic calibration with multiple objectives. *Advances in Water Resources*, 26(2):205–216, 2003.

[95] V. R. N. Pauwels. A multistart weight-adaptive recursive parameter estimation method. *Water Resources Research.*, 44(4), 2008.

[96] M. K. Gill, Y. H. Kaheil, A. Khalil, M. McKee, and L. Bastidas. Multiobjective particle swarm optimization for parameter estimation in hydrology. *Water Resources Research.*, 42(7), 2006.

[97] K. W. Chau. Particle swarm optimization training algorithm for anns in stage prediction of shing mun river. *Journal of Hydrology*, 329:363–367, 2006.

[98] G. Welch and G. Bishop. An introduction to the Kalman filter. Technical Report TR 95-041, Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, 1995.

[99] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the American Society of Mechanical Engineers, Series D, Journal of Basic Engineering*, 82:35 – 55, 1960.

[100] M. A. Abido. Optimal design of power-system stabilizers using particle swarm optimization. *IEEE Transactions on Energy Conversion*, 17(3):406–413, 2002.

[101] L. A. Richards. Capillary conduction of liquids through porous media. *Physics*, 1:318–333, 1931.

[102] R. H. Brooks and A. T. Corey. Hydraulic properties of porous media, 1964.

[103] K. J. Beven and M. J. Kirkby. A physically based, variable contributing area model of basin hydrology. *Hydrological Sciences Bulletin*, 24(1):43–69, 1979.

[104] W. J. Shuttleworth. *Evaporation*, pages 4.1–4.53. McGraw-Hill, 1992.

[105] V. R. N. Pauwels, W. Timmermans, and A. Loew. Comparison of the estimated water and energy budgets of a large winter wheat field during AgriSAR 2006 by multiple sensors and models. *Journal of Hydrology*, 349(3-4):425–440, 2008.

[106] W. Kohler and L. Johnson. *Elementary differential equations.* Greg Tobin, 2005.

[107] E. L. Andreas and B. A. Cash. A new formulation for the bowen ratio over saturated surfaces. *Journal of Applied Meteorology*, 35:1279–1289, 1996.

[108] I. C. Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85(6):317–325, 2003.

[109] Triola M.F. *Elementary Statistics.* Addison Wesley, 2009.

[110] K. Beven and A. Binley. The future of distributed models: model calibration and uncertainty prediction. *Hydrological Processes*, 6(3):279–298, 1992.

[111] H.-J. Zimmerman. *Fuzzy Set Theory and Its Applications.* Kluwer Academic Publishers, 1992.

[112] C. Baudrit, I. Couso, and D. Dubois. Joint propagation of probability and possibility in risk analysis: Towards a formal framework. *International Journal of Approximate Reasoning*, 45:82–105, 2007.

[113] D. Guyonnet, B. Bourgine, D. Dubois, H. Fargier, B. Come, and JP. Chiles. Hybrid approach for addressing uncertainty in risk assessments. *Journal of Environmental Engineering*, 129:68–78, 2003.

[114] I. Karimi, E. Hüllermeier, and K. Meskouris. A fuzzy-probabilistic earthquake risk assessment system. *Soft Computing*, 11:229–238, 2007.

[115] N. E. C. Verhoest, B. De Baets, and H. Vernieuwe. A Takagi-Sugeno fuzzy rule-based model for soil moisture retrieval from sar under soil roughness uncertainty. *IEEE Transactions on Geoscience and Remote Sensing*, 45, 2007.

[116] N. E. C. Verhoest, B. De Baets, G. Satalino, C. Lucau, and P. Defourny. A possibilistic approach to soil moisture retrieval from ERS synthetic aperture radar backscattering under soil roughness uncertainty. *Water Resources Research*, 43, 2007.

[117] S. Ferson and L. R. Ginzburg. Different methods are needed to propagate ignorance and variability. *Reliability Engineering & System Safety*, 54:133–144, 1996.

[118] B. M. Ayubb and G. J. Klir. *Uncertainty modelling and analysis in engineering and the sciences*. Chapman & Hall, 2006.

[119] A. P. Jacquin. Possibilistic uncertainty analysis of a conceptual model of snowmelt runoff. *Hydrology and Earth Systems Sciences*, 14:1681–1695, 2010.

[120] L. A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences*, 8:199–249, 1975.

[121] H. T. Nguyen. A note on the extension principle for fuzzy sets. *Mathematical Analysis and Applications*, 64:369–380, 1978.

[122] R. E. Moore, R. B. Kearfott, and M. J. Cloud. *Introduction to Interval Analysis*. SIAM, 2009.

[123] R. Fullér and T. Keresztfalvi. On generalization of nguyen's theorem. *Fuzzy Sets and Systems*, 41:371–374, 1991.

[124] D. Dubois, E. Kerre, R. Mesiar, and H. Prade. *Fundamentals of Fuzzy Sets*, chapter Fuzzy Interval Analysis, pages 483–561. Kluwer Academic Publishers, 2000.

[125] B. De Baets and A. Marková–Stupňanová. Analytical expressions for the addition of fuzzy intervals. *Fuzzy Sets and Systems*, 91:203–213, 1997.

[126] W. Dong and H. C. Shah. Vertex method for computing functions of fuzzy variables. *Fuzzy Sets and Systems*, 24:65–78, 1987.

[127] W. M. Dong and F. S. Wong. Fuzzy weighted averages and implementation of the extension principle. *Fuzzy Sets and Systems*, 21:183–199, 1987.

[128] K. N. Otto, A. D. Lewis, and E. K. Antonsson. Approximating α-cuts with the vertex method. *Fuzzy Sets and Systems*, 55:43–50, 1993.

[129] M. Hanss. The transformation method for the simulation and analysis of systems with uncertain parameters. *Fuzzy Sets and Systems*, 130:277–289, 2002.

[130] J. Fortin, D. Dubois, and H. Fargier. Gradual numbers and their application to fuzzy interval analysis. *IEEE Transactions on Fuzzy Systems*, 16:388–402, 2008.

[131] D. Dubois and H. Prade. Gradual elements in a fuzzy set. *Soft Computing*, 12:165–175, 2008.

[132] S. Maskey, V. Guinot, and R. Price. Treatment of precipitation uncertainty in rainfall-runoff modelling: a fuzzy set approach. *Advances in Water Resources*, 27:889–898, 2004.

[133] R. R. Shrestha, A. Bárdossy, and F. Nestmann. Analysis and propagation of uncertainties due to the stage-discharge relationship: a fuzzy set approach. *Hydrological Sciences*, 52:595–610, 2007.

[134] Y. Chalco-Cano, H. Román-Flores, M. Rojas-Medar, O. R. Saavedra, and M. D. Jiménez-Gamero. The extension principle and a decomposition of fuzzy sets. *Information Sciences*, 177:5394–5403, 2007.

[135] Y. Chalco-Cano, M. T. Misukoshi, H. Román-Flores, and A. Flores-Franulic. Spline approximation for Zadeh's extensions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 17:269–280, 2009.

[136] A. Klimke, B. Wohlmuth, and K. Willner. Uncertainty modeling using fuzzy arithmetics based on sparse grids: Application to dynamic systems. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 12:745–759, 2004.

[137] A. Klimke and B. Wohlmuth. Computing expensive multivariate functions of fuzzy numbers using sparse grids. *Fuzzy Sets and Systems*, 153:432–453, 2005.

[138] J. W. Eaton. *GNU Octave Manual*. Network Theory Limited, 2002.

[139] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[140] D. Dubois and H. Prade. *Fuzzy Sets and Systems*. Academic Press, 1988.

[141] H. Vernieuwe, N. E. C. Verhoest, H. Lievens, and B. De Baets. Possibilistic soil roughness identification for uncertainty reduction on SAR-retrieved soil moisture. *IEEE Transactions on Geoscience and Remote Sensing*, 49:628–638, 2011.

[142] E. P. Klement, R. Mesiar, and E. Pap. Triangular norms. position paper i: basic analytical and algebraic properties. *Fuzzy Sets and Systems*, 143:5–26, 2004.

[143] E. P. Klement, R. Mesiar, and E. Pap. *Triangular norms*. Kluwer Academic Publishers, 2000.

[144] M. Hanss. *Applied Fuzzy Arithmetic*. Springer-Verlag Berlin Heidelberg, 2005.

[145] J. B. Stewart, E. T. Engman, R. A. Feddes, and Y. Kerr. *Remote sensing and scaling in hydrology, scaling in hydrology using remote sensing*. John Wiley & Sons Inc., 1996.

[146] P. J. De Roo, R. J. E. Offermans, and N. H. Cremers. Lisem: a single-event, physically based hydrological and soil erosion model for drainage basins. ii: sensitivity analysis, validation and application. *Hydrological Processes*, 10:1119–1126, 1996.

[147] D. Entekhabi, H. Nakamura, and E G. Njoku. Solving the inverse problem for soil moisture and temperature profiles by sequential assimilation of multi-frequency remotely sensed observations. *IEEE Transactions on Geoscience and Remote Sensing*, 32(2):438–48, 1994.

[148] T. J. Jackson, T. J. Schmugge, and E. T. Engman. Remote sensing applications to hydrology: soil moisture. *Hydrological Sciences*, 41:517–530, 1996.

[149] A. K. Fung and K. S. Chen. Bistatic signal statistics of randomly roguh surfaces. In *Proc. International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 183–185, 1992.

[150] A. K. Fung, Z. Li, and K. S. Chen. Backscattering from a randomly rough dielectric surface. *IEEE Transactions on Geoscience and Remote Sensing*, 30:356–369, 1992.

[151] N. E. C. Verhoest, H. Lievens, W. Wagner, J. Álvarez-Mozos, M. S. Moran, and F. Mattia. On the soil roughness parameterization problem in soil moisture retrieval of bare surfaces from synthetic aperture radar. *Sensors*, 8:4213–4248, 2008.

[152] H. Lievens, H. Vernieuwe, J. Álvarez-Mozos, B. De Baets, and N. E. C. Verhoest. Error in radar-derived soil moisture due to roughness parameterization. *Sensors*, 9:1067–1093, 2009.

[153] M. Callens, N. E. C. Verhoest, and M. W. J. Davidson. Parameterization of tillage-induced single-scale soil roughness from 4-m profiles. *IEEE Transactions on Geoscience and Remote Sensing*, 44(4):878–888, 2006.

[154] Y. Oh and Y. C. Kay. Condition for precise measurement of soil surface roughness. *IEEE Transactions on Geoscience and Remote Sensing*, 36:691–695, 1998.

[155] M. W. J. Davidson, T. Le Toan, F. Mattia, G. Satalino, and M. Borgeaud. On the characterization of agricultural soil roughness for radar remote sensing studies. *IEEE Transactions on Geoscience and Remote Sensing*, 38:630–640, 2000.

[156] M.C. Dobson, F.T. Ulaby, M.T. Hallikainen, and M.A. El-Rayes. Microwave dielectric behavior of wet soil, part ii: Dielectric mixing models. *IEEE Transactions on Geoscience and Remote Sensing*, 33:35–46, 1985.

[157] A. K. Fung. *Microwave Scattering and Emission Models and their Applications, 1994*. Artech House, 1994.

[158] K. S. Chen, S. K. Yen, and W. P. Huang. A simple model for retrieving bare soil moisture from radar-scattering coefficients. *Remote Sensing Environment*, 54:121–126, 1995.

[159] N. E. C. Verhoest. *Retrieval of soil moisture information from synthetic aperture radar data at the point and the catchment scale*. PhD dissertation, Ghent University, Faculty of Agricultural and Applied Biological Sciences, 2000.

[160] M. W. J. Davidson, F. Mattia, G. Satalino, N. E. C. Verhoest, T. Le Toan, M. Borgeaud, J.MB. Louis, and E. Attema. Joint statistical properties of rms height and correlation length from multisite 1-m roughness measurements. *IEEE Transactions on Geoscience and Remote Sensing*, 41:1651–1658, 2003.

[161] D. Dubois and H. Prade. *Fuzzy Information and Decision Processes*, chapter On several representations of an uncertain body of evidence, pages 167–181. Elsevier Science Ltd, 1982.

[162] D. Dubois, L. Foulloy, G. Mauris, and H. Prade. Probability-possibility transformations, triangular fuzzy sets, and probabilistic inequalities. *Reliable Computing*, 10:273–297, 2004.

[163] A. Ban. Approximation of fuzzy numbers by trapezoidal fuzzy numbers preserving the expected intervals. *Fuzzy Sets and Systems*, 159:1327–1344, 2008.

[164] R. Krishnapuram and J.M. Keller. A possibilistic approach to clustering. *IEEE Transactions on Fuzzy Systems*, 1:98–110, 1993.

[165] T. Pukkala. *Multi-Objective Forest Planning*. Kluwer Acadamic Publishers, 2003.

[166] R.E. Steur. Multiple criteria optimization: Theory, computations, and applications. 1986.

[167] I. Das and J.E. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization. *SIAM Journal on Optimization*, 8:631–657, 1998.

[168] A. Messac and C.A. Mattson. Normal constraint method with guarantee of even representation of complete pareto frontier. *American Institute of Aeronautics and Astronautics Journal*, 42:2101–2111, 2004.

[169] D. Mueller-Gritschneder, H. Graeb, and U. Schlichtmann. A successive approach to compute the bounded pareto front of practical multiobjective optimization problems. *SIAM Journal on Optimization*, 20:915–934, 2009.

[170] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons, 2001.

[171] K. Deb. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

[172] E. Ducheyne. *Multiple objective forest management using GIS and genetic optimisation techniques*. PhD thesis, Ghent University, 2003.

[173] D. Angus and C. Woodward. Multiple objective ant colony optimisation. *Swarm Intelligence*, 3:69–85, 2009.

Curriculum Vitae

Personalia

Naam	Karolien Scheerlinck
Geslacht	vrouwelijk
Nationaliteit	Belg
Geboortedatum	05/07/1984
Geboorteplaats	Aalst
Burgelijke stand	gehuwd
Kinderen	geen
e-mail adres	karolien.scheerlinck@gmail.com

Opleiding

2002-2007: Bio-ingenieur Land- en Bosbeheer, Universiteit Gent

Scriptie: Evaluatie van strategieën voor de aanmaak van een vegetatiekaart van Schiermonnikoog met hyperspectraal beeldmateriaal.

Promotor: Prof. dr. ir. R. De Wulf

Loopbaanoverzicht-werkervaring

2008-2011: Doctoraatsstudent faculteit Bio-ingenieurswetenschappen, Universiteit Gent

Vakgroep: Vakgroep Wiskundige Modellering, Statistiek en Bio-informatica

Promotor: Prof. dr. Bernard De Baets

Conferenties

2010	European Conference on Operational Research, 11-14 juli, Lissabon, Portugal, Oral presentation
2010	International Conference on Fuzzy Computation, 24-26 oktober, Valencia, Spanje, Oral presentation

Wetenschappelijke output

Gepubliceerd

K. Scheerlinck, V.R.N. Pauwels, H. Vernieuwe and B. De Baets (2009). Calibration of a water and energy balance model: recursive parameter estimation versus particle swarm optimization. Water Resources Research, 45, W10422, doi: 10.1029/2009WR008051.

K. Scheerlinck, B. De Baets, I. Stefanov and V. Fievez (2010). Subset selection from multi-experiment data sets with application to milk fatty acid profiles. Computers and Electronics in Agriculture, 73, pp. 200-212.

K. Scheerlinck, H. Vernieuwe and B. De Baets (2011). Zadeh's extension principle for continuous functions of non-interactive variables: a parallel optimization approach. Accepted in IEEE Transactions on Fuzzy Systems, doi: 10.1109/TFUZZ.2011.2168406

Ingediend

J. Verwaeren, K. Scheerlinck and B. De Baets (2011). Countering the negative search bias of ant colony optimization in subset selection problems. Submitted to International Journal of Operational Research.

Proceedings

K. Scheerlinck, H. Vernieuwe and B. De Baets (2010). Development of a fuzzy calculator for continuous functions of non-interactive fuzzy variables. 2010 Proceedings of International Conference on Fuzzy Computation (ICFC

2010) and International Conference on Neural Computation (ICNC 2010), pp. 14-20.
Award: Best paper award