Diss. ETH No. 15240

Multi-Objective Evolutionary Optimization of Gas Turbine Components

A dissertation submitted to the
SWISS FEDERAL INSTITUTE OF TECHNOLOGY
ZÜRICH

for the degree of
DOCTOR OF TECHNICAL SCIENCES

presented by
Dirk Büche

Dipl.-Ing. (Universität Stuttgart)
born on March 8th, 1974
citizen of Germany

accepted on the recommendation of
Prof. Dr. P. Koumoutsakos, examiner
Prof. Dr. R. Abhari, co-examiner
Dr. P. Jansohn, co-examiner
2003

Abstract

The world-wide increasing demand for energy is one of the key challenges of our century. However, this demand is in conflict to the Kyoto protocols of 1997 and 2001 that contain an international agreement for reducing green house gases. Energy generation concepts involving alternative energy sources and innovative technologies with high thermodynamical efficiencies are needed in order to address these issues.

Gas turbines are one of the key energy producing devices of our generation. Improved designs of gas turbine components are necessary in order to address the increasing demands of high performance and reduced emissions. The design of gas turbine components is a complex and time-consuming engineering task that involves meeting of several design objectives and constraints. This task is usually addressed in an iterative process. Advances in domain knowledge and in areas such as information technology offer the possibility of accelerating and improving this design cycle through automated optimization procedures.

This thesis addresses the key issue of automated optimization by presenting optimization algorithms that are implemented in realistic design processes of gas turbine components. The results of this work include algorithmic advances and the development of new and efficient designs for turbine components. The proposed optimization algorithms are analyzed and enhanced so that they are applicable to the design of compressor blades and burners. The thesis addresses a number of optimization difficulties such as multiple design objectives and constraints, high sensitivity of the objectives, and noise in the evaluation of the objectives.

Automated optimization requires a blend of efficient algorithms and information technology tools from in order to find optimal solutions in limited time frames. We find that it is not sufficient to use existing tools but that extensive, problem specific modifications of the algorithms are required. In this thesis, several new algorithms for single and multi-objective Pareto optimization are presented. The focus is on evolutionary algorithms, as they are robust optimization algorithms suitable for engineering applications where pointwise information is only available. In the context of Pareto optimization, several well known evolutionary algorithms are analyzed with regards to their convergence properties and convergence limits.

Furthermore, the efficiency of certain algorithms is enhanced by defining adaptive mutation and recombination operators based on self-organizing maps. The robustness to noisy objective functions is improved by defining a dominance dependent re-evaluation interval. In the field of single objective optimization, a new optimization algorithm is proposed that is suitable for engineering problems with relatively few decision variables but expensive cost function evaluations. The algorithm relies on the construction of an empirical model of the objective function. The model is then used to predict function values in order to reduce the number of function evaluations.
For the compressor optimization, an optimization procedure is defined that addresses the relevant design objectives and constraints. The procedure comprises a new blade parameterization and tools for the aerodynamical and mechanical analysis. For the aerodynamical analysis, an inexpensive method for estimating off-design behavior is presented. Four compressor blades of adjacent mid-stages of a gas turbine are optimized and the resulting profile shapes are discussed.
The gas turbine burner is optimized in an experimental test-rig. The objectives are to minimize thermo-acoustic pulsations and NO_x emissions. Both objectives are noisy as they result from measurements with limited time averaging. The Pareto optimization results in an approximation of the Pareto set, comprising a number of different trade-off solutions for the two conflicting objectives.
The thesis concludes by identifying a number of open questions and outlines directions for future work.

Zusammenfassung

Der weltweit steigende Bedarf an Energie ist eine der zentralen Herausforderungen unseres Jahrhunderts. Die Bereitstellung der Energie ist schwierig angesichts des in den Kyoto Protokollen von 1997 und 2001 unterzeichneten internationales Abkommens zur Reduzierung der Treibhausgase. Deshalb werden zur Energiegewinnung alternative Energiequellen und innovative Kraftwerke mit hohem thermodynamischen Wirkungsgrad benötigt.
Gasturbinen sind eine der zentralen Kraftwerkstechnologien unserer Zeit. Kontinuierlich verbesserte Gasturbinenkomponenten werden benötigt um die steigenden Anforderungen an Wirkungsgrad und Emissionen zu erfüllen. Die Entwicklung von Gasturbinenkomponenten ist eine komplexe und zeitintensive Ingenieursaufgabe, die zahlreiche Zielfunktionen und Zwangsbedingungen enthält. Diese Aufgabe wird in der Regel in einem iterativen Prozess gelöst. Fortschritte im Bereich der Physik von Gasturbinen und in Bereichen wie der Informationstechnologie ermöglichen den Entwicklungsprozess zu beschleunigen und durch automatische Optimierungsverfahren zu verbessern.
Die vorliegende Dissertation beschäftigt sich mit diesem zentralen Problem der automatischen Optimierung und präsentiert Optimierungsverfahren, die in reale Entwicklungsprozesse von Gasturbinenkomponenten implementiert werden. Die Ergebnisse dieser Arbeit beinhalten Verbesserungen der Verfahren und die Entwicklung von neuen und effizienten Komponentendesigns. Dabei werden die entwickelten Optimierungsalgorithmen analysiert und verbessert, sodass sie auf die Entwicklung von Kompressorschaufeln und Brenner angewandt werden können. Herausforderungen an die Optimierung sind u.A. die Vielzahl der Zwangsbedingungen und Zielfunktionen, die hohe Senitivität und das Rauschen in den Zielfunktionen.
Automatische Optimierung benötigt eine Auswahl effizienter Algorithmen und Werkzeuge der Informationstechnologie, um optimale Lösungen in vorgeschriebenen Zeitgrenzen zu erhalten. Wir erachten es als unzureichend existierende Optimierungsverfahren zu verwenden, sondern erachten aufwendige und problemspezifische Modifikationen als Grundvoraussetzung für eine erfolgreiche Optimierung. In dieser Arbeit werden mehrere neue Optimierungsalgorithmen für die Ein-

und Mehrkriterienoptimierung vorgestellt. Dabei liegt der Schwerpunkt auf evolutionären Algorithmen, welche sich als robust und effizient in Ingenieursanwendungen erweisen. Im Bereich der Mehrkriterienoptimierung werden Standardalgorithmen bezüglich ihrer Konvergenzeigenschaften und Konvergenzbeschränkungen analysiert. Des weiteren wird die Effizienz dieser Algorithmen durch die Entwicklung adaptiver Mutations- und Rekombinationsoperatoren auf Basis von selbstorganisierenden Netzwerken verbessert. Die Robustheit der Algorithmen gegenüber verrauschten Zielgrössen wird durch die Definition von dominanzabhängigen Reevaluationsintervallen verbessert. Im Bereich der Einkriterienoptimierung wird ein neuer Optimierungsalgorithmus vorgeschlagen, welcher empirische Modelle der Zielfunktion konstruiert. Dieser Algorithmus eignet sich insbesondere für Optimierungsprobleme mit einer geringen Zahl freier Variablen bei gleichzeitig teuren Funktionsauswertungen. Das Modell wird zur Vorhersage von Funktionswerten verwendet, um die Zahl der Auswertungen zu reduzieren.
Für die Kompressoroptimierung wird eine Prozedur definiert, welche alle relevanten Zielfunktionen und Zwangskriterien eines realistischen Entwurfsprozesses beachtet. Die Prozedur beinhaltet eine neue Schaufelparametrisierung und Berechnungsprogramme für die Aerodynamik und die Mechanik der Schaufel. Für die Bewertung der Aerodynamik ausserhalb des Betriebspunktes wird eine kostengünstige Methode entwickelt. Insgesamt werden vier Kompressorschaufeln aus benachbarten Reihen des mittleren Kompressorbereichs optimiert und die Ergebnisse diskutiert.
Der Gasturbinenbrenner wird in einem experimentellen Aufbau optimiert. Ziel der Optierung ist die Minimierung der thermoakustischen Pulsationen und der NO_x Emissionen. Beide Zielfunktionen werden durch zeitlich gemittelten Messungen erfasst, welche verrauscht sind. Die Pareto-Optimierung findet eine Approximation der Pareto-Front, welche aus Kompromisslösungen für die beiden widersprüchlichen Zielgrössen besteht.
Diese Dissertation schliesst mit einer Identifikation offener Fragen für zukünftige Forschungsarbeiten.

Acknowledgments

This thesis results from 3 1/4 interesting years as a PhD student at the Eidgenössische Technische Hochschule (ETH) in Zürich, Switzerland. My PhD project was part of a cooperation with the Alstom Power Technology Center (APT) in Dättwil, Switzerland and showed me both the academic and the industrial view on research. In principle, these two views are conflicting and need to be balanced. However, both sides gave me the necessary freedom to do efficient research and I benefited by finding on both sides open people that were interested in my project and supported me greatly. While ETH gave me inspiration on optimization algorithms, APT provided applications to optimize. The following acknowledgment is far from being complete and hopefully nobody feels offended by not being mentioned or since only some aspects are acknowledged.

First of all, I wish to thank my supervisor Prof. Petros Koumoutsakos for giving me the opportunity of doing a PhD, for his support, and for teaching me how to work efficiently. Gratitude goes also to my two co-referents Prof. Abhari from ETH and Dr. Peter Jansohn from the APT for agreeing to do this important task. Furthermore, I wish to thank Dr. Peter Jansohn for the constructive and open work atmosphere in his department. Special thanks to my supervisors from APT, Dr. Rolf Dornberger, Dr. Peter Stoll, and Dr. Gianfranco Guidati who subsequently took over this position. Dr. Rolf Dornberger motivated me to do a dissertation and set up the contact to Prof. Koumoutsakos. Dr. Peter Stoll introduced me to programming clean and object-oriented Java code. Dr. Gianfranco Guidati developed the compressor blade parameterization and promoted optimization within APT.

I would like to thank also all people within APT that promoted me. Especially the executive Dr. Tony Kaiser, who gave me the feeling of being part of APT and not an external employee. Dieter Winkler and Camille Pedretti for inspiring discussions. Dr. Bruno Schuermans for the realization of the burner optimization. Dr. Christian Oliver Paschereit for several useful discussions, as well as providing the burner test-rig.

Gratitude goes to all my colleges from the Institute of Computational Science at ETH. Especially to Dr. Sibylle Müller for discussions, cooperations, and proofreadings of my papers and thesis, Dr. Michele Milano for introducing me to self-

organizing networks, Dr. Nicol Schraudolph for teaching me machine learning and discussing Gaussian processes, Dr. Jens Walther for providing technical administration and solving many hard- and software problems that I encountered, Dr. Nikolaus Hansen for fundamental and theoretical discussions about optimization and proof-reading my thesis, and Stefan Kern for proof-reading.
Financial support is acknowledged from ETH, from the Swiss Commission for Technology and Innovation (CTI), Project 4817.1, and from APT.
I would like to thank my parents for greatly supporting all projects and ideas that I had and my brother Marcus for great understanding and help. Finally I thank my wife Stefanie and my daughter Alida. My wife, for supporting me and understanding all positive and negative aspects of doing a PhD. My daughter, for reminding me of the funny way of life and that many things are really not so important as they might appear.

Contents

List of Acronyms

ANN	Artificial Neural Networks
BFGS	Broyden-Fletcher-Goldfarb-Shannon algorithm
CFD	Computational Fluid Dynamics
CMA-ES	Evolution Strategy with Covariance Matrix Adaptation
CMEA	Constraint Method-based Evolutionary Algorithm
EA	Evolutionary Algorithm
EP	Evolutionary Programming
ES	Evolution Strategy
ESPEA	Estimate Strength Pareto Evolutionary Algorithm
FFM	Fitness Function Model
GA	Genetic Algorithm
GP	Gaussian Process
GPOP	Gaussian Process Optimization Procedure
MOEA	Multi-objective Evolutionary Algorithm
NO_x	Nitrogen Oxide
NPGA	Niched Pareto Genetic Algorithm
NSGA	Nondominated Sorting Genetic Algorithm
NT-SPEA	Noise-Tolerant Strength Pareto Evolutionary Algorithm
Q3D	Quasi Three-Dimensional
RBFN	Radial Basis Function Network
ROT-ES	Evolution Strategy with ROTating angle self-adaptation
SDM	SubDivision Method
SOM	Self-Organizing Map
SPEA	Strength Pareto Evolutionary Algorithm
VEGA	Vector Evaluated Genetic Algorithm

List of Symbols

Latin Letters

c	change rate of the evolution and cumulation path in CMA-ES
c	step length factor
c_1	lower threshold related to the dominated fraction of the archive
c_2	upper threshold related to the dominated fraction of the archive
c_{cov}	change rate of the covariance matrix in CMA-ES
c_{ij}	element of the covariance matrix
d_σ	damping parameter for the step size
d_i	length of the local search space in direction i.
e	number of training epochs
e_i	unit vector in coordinate direction i.
f	objective function
f^{L}	lower bound of a property interval
f_{M}	merit function
f^{U}	upper bound of a property interval
$\mathbf{f}$	vector of objective functions
g	generation of the evolutionary optimization
$\mathbf{g}$	gradient of the objective function $\mathbf{g} = \nabla f$
h	neighborhood kernel of a SOM
$\mathbf{k}$	vector of correlation between the known points and a new point
l	lattice direction
$l_{\alpha_{\mathrm{D}}}$	aerodynamical loss at design incidence
$l_{\alpha_{\mathrm{D}}\pm\Delta\alpha}$,	aerodynamical loss at off-design incidences
l_{e}	length of an edge of a simplex
l_i	lower constraint value for f_i
n	dimension of the decision space
n_{SOM}	lattice dimension of a SOM
m	dimension of the objective space
p_{M}	mutation probability

p_s	success rate
$\mathbf{p}_\mathrm{c}$	evolution path
$\mathbf{p}_\sigma$	cumulated path
r	Euclidean distance between two neurons in the lattice space
r_i	normalization radii for the GP
$\mathbf{s}$	search direction of a gradient-based optimization method
$\hat{t}$	predicted mean of a GP
$\mathbf{t}_N$	set of N function values $\{t_1, t_2, \ldots, t_N\}$
u_i	upper constraint value for f_i
$\mathbf{w}$	reference vector associated with a neuron in a SOM
$\mathbf{x}$	vector of decision variables
$\mathbf{x}'$	recombined vector of decision variables
$\mathbf{x}''$	mutated vector of decision variables
$\mathbf{z}$	vector of random numbers
A	archive
$\mathbf{B}$	orthonormal matrix of eigenvectors from $\mathbf{C}$
$\mathbf{C}$	covariance matrix
D	maximal distance of a nondominated solution to the closest point on the Pareto front
D_DS	performance measure for multi-objective algorithms in decision space
D_OS	performance measure for multi-objective algorithms in objective space
$\mathbf{D}$	diagonal matrix containing the square root of the eigenvalues of $\mathbf{C}$
F	objective space
H_{12}	non-dimensional shape factor of the boundary layer
$\mathbf{I}$	identity matrix
$\mathbf{L}$	lower triangular matrix of a LU decomposition
$\mathcal{L}$	logarithm of the likelihood
Ma	Mach number
M_d	average distortion measure for SOMs
M_I	mechanical integrity indicator
N	number of evaluated solutions
N_C	number of training data for GP based on the distance to the currently currently best solution.
N_D	number of training data for the SOM and GP
N_SOM	number of neurons in a SOM
N_R	number of training data for GP based on the history.
$\mathcal{N}(0, \sigma^2)$	normal distribution with zero expectation and standard deviation σ
$\mathcal{O}$	order

P	performance measure for multi-objective algorithms
P_λ	offspring population
P_μ	parent population
R	mean probability of being dominated by an archive solution
R	performance measure for multi-objective optimization
$\mathbb{R}$	space of real numbers
S	fitness of a solution
$\mathbf{U}$	upper triangular matrix of a LU decomposition
$\mathcal{U}(0,1)$	uniform distribution between 0 and 1
X	decision space
$\mathbf{X}_N$	set of N decision vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$

Greek Letters

α	learning rate for the SOM training
α	rotating angle
α	threshold for the archive in ESPEA
α	weighting coefficient for the merit function
α_D	inlet flow angle at design condition
β	mean rotation rate
β	minimal ratio of neurons per parent
γ	multiplicative factor for the recombination strength
2δ	size of the property interval
θ	all hyperparameters of the GP
$\theta_{\{1,2,3\}}$	hyperparameters of the GP
κ	lifetime of an individual
κ	re-evaluation interval for an elitist solution
κ	autocorrelation of data point
λ	number of offspring
μ	number of parents
σ	step size
σ	standard deviation
σ_t	predicted standard deviation of a GP
σ_B	bell width
σ_SOM	characteristic for the decay of the learning rate
τ_0	global learning rate
τ	individual learning rate

τ_i	resource variable
χ	expected path length
$\Delta\alpha$	inlet flow angle variation
$\Delta\beta$	deviation from the design turning angle

Introduction

Finding optimal solutions to real world design applications is usually an iterative process limited by the available resources. These limits may be the maximal number of possible computational and experimental design evaluations or the limited labor time of the design engineers. The development of optimal designs can be achieved by human designers or by exploiting automated optimization techniques. Human designers exploit their accumulated domain knowledge while automated optimization algorithms search by analyzing design evaluations resulting from a systematic parameter variation. For a large number of problems human design is the method of choice, however as the decision space becomes large and the associated processes more complex, the systematic procedures of automated optimization become an interesting alternative. In addition, automated optimization can take full advantage of today's massively parallel computer architectures and it may result in alternative designs and eventually lead to new design philosophies.
The increasing interest in automated optimization is also a result of the growing complexity of the design processes. Design objectives and constraints from multiple disciplines need to be addressed concurrently in order to reduce the design time. Often these objectives are conflicting and there exists no best solution but a set of best trade-off solutions, representing different compromises between the objectives. The set of best trade-off solutions are referred to as Pareto solutions [84]. Automated optimization addresses these multi-objective problems by searching several trade-off solutions in parallel. From the resulting set of solutions, the designer can then select the design that fits best his specific requirements for the different objectives.
The setup of an automated optimization requires three steps. The first step addresses the automation of the data flow. The different evaluation tools of a design process usually require input data in different formats. For an automated data flow, interfaces are necessary in order to convert formats and to execute the evaluation tools without user interaction. Then, a large number of designs can be evaluated with minimal user time, while performing simultaneously sensitivity analysis and optimization. In the second step, the design objectives and constraints are defined along with the free decision variables that are allowed to be modified in the

optimization process. The objectives and constraints have to be formulated as a function of the design evaluation and optimization can then be formulated as the classical mathematical problem of finding the extreme values of functions. In the third step, an optimization algorithm has to be selected.

The selection of the optimization algorithm is highly problem dependent. When the gradient of all objectives and constraints is available, gradient-based methods are mainly employed. However, in many real-world applications, gradient information is not available and approximating this information can be inaccurate as function evaluation is noisy [76]. In experimental setups, noise in the function evaluation may result from changing environmental conditions or limited measuring precision. In such cases, non-gradient based methods that use only function values to guide the search are often applied. Within theses methods, deterministic and stochastic algorithms are distinguished. Deterministic algorithms follow a fixed search pattern, while stochastic algorithm involve some random processes in generating new designs.

Evolutionary Algorithms (EAs) [5] are well-known representatives of stochastic algorithms. EAs imitate the principles of natural evolution to find an optimal solution to a problem. Natural evolution is mainly driven by the principles of fitness-based selection and recombination/mutation of genetic information. In nature, individuals that are well adapted to their environment are more likely to survive and spawn. In an engineering environment, the natural evolution is translated to function optimization, where individuals evolve in order to improve the function value. The term Evolutionary Algorithm summarizes techniques such as Genetic Algorithms [54, 49], Evolutionary and Genetic Programming [36], and Evolution Strategies [90, 101].

The widespread use of EAs for the optimization of real-world applications is attributed to their simplicity, transferability and robustness. EAs are inherently parallel algorithms that evaluate a population of individuals concurrently, thus taking full advantage of today's massively parallel computer architectures. Optimization problems in general have a wide variety of different properties such as discrete or continuous parameters, multiple objectives and constraints, noise in the objective function, or only discrete objective values. Furthermore, the problem might be combinatorial. Different EAs have been designed to address these properties [6].

In this thesis, we develop automated optimization procedures for gas turbine components. In particular, we address the optimization of compressor blades and fuel injection patterns in combustion chambers. Today, compressor blades are designed using expensive numerical simulations. Blades are optimized with respect to high thermodynamic efficiency over a wide operating range, while they must withstand

mechanical loading and avoid critical excitation. For combustion processes, numerical simulations are still rarely used, since combustion processes are difficult to handle with today's computational techniques and computer resources. The underlying chemical reactions are complex and the mixing of fuel and air has to be sufficiently resolved since mixing is mainly responsible for the flame properties. Thus, experimental test-rigs are widely used to analyze combustion processes.
For the two gas turbine design problems, this thesis focuses especially on:

- How gas turbine components can be parametrised.
- Which tools (i.e., software, test-rigs) should be selected to evaluate these components.
- Which design objectives and constraints need to be addressed and how can they be formulated in mathematical terms.

The design of gas turbine components entails multiple objectives that are conflicting. We may consider two approaches to handle multi-objective problems. First, all objectives can be aggregated and optimizing this figure of merit leads to a single trade-off solution between the objectives. Second, the population-based search of EAs can be used to converge to the set of best trade-off solutions. This approach is also denoted as Pareto optimization. The choice between the two approaches depends on the interest of the designers.
For the Pareto optimization, a wide variety of so-called Multi-Objective Evolutionary Algorithms (MOEAs) have been developed and compared [24]. However, these algorithms are often analyzed on prototypical functions. When applying them to real design problems their performance may be different or modifications to the algorithms may be a prerequisite for a successful optimization.
From our experience on the considered design problems, we address the following questions in this thesis:

- Compared to single objective EAs, are MOEAs efficient optimization algorithms and do they convergence sufficiently towards the Pareto front?
- How can the robustness of MOEAs against noise in the objective functions be improved?
- Can MOEAs be improved by adaptive mutation and recombination operators?
- Can empirical models of the objective functions improve the optimization process?

This thesis is structured in 3 parts. In Part I, a literature survey on optimization algorithms is presented. Part II introduces analysis and improvements of existing optimization algorithms. Finally, Part III describes the compressor and burner optimization.

Chapter 1 in Part I provides an overview on automated optimization. The multi-objective optimization problem is defined in mathematical terms. Various optimization algorithms are classified.

A detailed survey on the current state-of-the-art Evolution Strategies for continuous problems is given in Chapter 2. Basic selection, recombination, and mutation operators are introduced. A common feature of all algorithms is learning, although the learning concepts differ among the considered algorithms. The main goal of learning is to reduce the number of function evaluations by adapting the mutation distribution.

Chapter 3 presents state-of-the-art Evolutionary Algorithms for multi-objective problems. The multi-objective optimization problem is introduced as well as the concept of dominance. A particular focus is on algorithms that converge to the Pareto set in a single optimization run.

An alternative method to the learning of mutation distribution in Evolution Strategies is given in Chapter 4. In this chapter, an overview on algorithms that learn empirical models of the objective function is presented. Part II consists of Chapters 5 to 8 and presents advances in the field of optimization algorithms that were obtained within this thesis. Different multi-objective EAs are analyzed in Chapter 5 concerning their convergence to the Pareto front. It is proven that the convergence of some algorithms is limited, i.e., the optimization stagnates at a certain distance to the Pareto front. Furthermore, the chapter shows how adaptive mutation from single objective optimization can be transferred to multi-objective optimization.

Chapter 6 discusses the problem of noise in the objective functions for multi-objective optimization. While for single objective EAs noise has been analyzed in detail[3], there is a lack in analysis for multi-objective optimization. In multi-objective optimization, state-of-the-art algorithms are shown to be sensitive to noise and modifications for increased robustness are proposed and validated.

In Chapter 7, an adaptive recombination operators for multi-objective optimization are proposed based that base on a self-organizing map. The self-organizing map is trained on the best solutions found so far and then used as basis for recombination and mutation. In spite of the common believe that the selection operator is the most important operator in multi-objective optimization, the analysis shows a potential performance gain when using adaptive recombination operators.

A novel approach for integrating empirical models in optimization algorithms is

introduced in Chapter 8. The models are integrated into an iterative optimization procedure that is denoted as Gaussian Process Optimization Procedure (GPOP). The procedure proposes concepts for local modeling and search. Furthermore, the optimization of real-world application with several objectives and constraints is discussed. The procedure is especially useful for problems with expensive design evaluations, since the training of the model requires significant computational time.

In Part III (Chapters 9 to 11) EAs and the GPOP are applied to the design of gas turbine compressor blades and burners of the combustion chamber. A general introduction to gas turbines is given in Chapter 9 showing the main components and the use of gas turbines.

Chapter 10 presents the multi-objective optimization of an experimental combustion test-rig. The test-rig comprises a gas turbine burner under atmospheric conditions. Burners are designed mainly for two objectives, namely the reduction of emissions and the minimization of thermo-acoustic pressure waves (pulsation). The two objectives are conflicting and optimization results always in compromise solutions. Thus, a multi-objective optimization algorithm is applied. Since the objective values result from time averaged measurements, they are always subject to noise requiring a robust optimization algorithm.

In Chapter 11, compressor blades are optimized concerning all relevant design objectives and constraints of a realistic design setup. Optimizing compressor blades is a difficult task, since it represents a highly constraint problem and the objectives and constraints are very sensitive to small variations of the blade geometry. These properties make blade optimization an interesting candidate to compare single and MOEAs as well as the optimization procedure integrating empirical models.

This thesis closes with an outlook and proposals for future work in Chapter 13. Finally, conclusions are given in Chapter 12 that extend the chapters by more general considerations.

Part I

Survey on Optimization Algorithms

Chapter 1

Overview

The automated optimization of a problem requires the definition of the set of decision variables that should be optimized, a single or a set of functions that measures the quality of these decision variables, and the selection of an optimization algorithm. This chapter introduces the multi-objective optimization problem and classifies optimization approaches. Furthermore, an overview over some of the most widely used optimization algorithms for engineering problems is given.

1.1 Multi-Objective Optimization Problem

1.1.1 Introduction

Optimization can be defined as the search for the best possible solution(s) to a given problem. Real-world problems often entail the optimization of multiple objectives. If these objectives are conflicting, then no best solution exists, but a set of good compromise solutions.
Given a set of solutions to the problem, a partial ordering can be found by the principle of *dominance*: A solution is clearly better than (dominating) another solution, if it is better or equal in all objectives, but at least better in one objective. Using this principle, the set of best compromise solutions results by removing all solutions that are dominated by at least one other solution. The remaining solutions are all of equal quality (indifferent). A mutual comparison of always two solutions shows that each one is always better and worse in at least one objective. This set of indifferent solutions is referred to as the *Pareto set*, named after the work of the engineer and economist Vilfredo Pareto [84]. Starting from a Pareto solution, one objective can only be improved at the expense of at least one other objective.
Preference information of the decision maker is needed to perform a further selection. This selection process is also known as Multiple Criteria Decision Making

(MCDM).

1.1.2 Definitions

A multi-objective optimization problem can be described by a vector of decision variables $\mathbf{x}$ and the corresponding vector of objectives $\mathbf{f} = \mathbf{f}(\mathbf{x})$. Without loss of generality we restrict ourselves to the minimization of all objectives, since every maximization of a function f can be transformed into a minimization problem with:

$$\max\left(f(\mathbf{x})\right) = -\min\left(-f(\mathbf{x})\right) \tag{1.1}$$

Definition 1 *The multi-objective optimization problem is defined as the search for the set of solutions* $\mathbf{x}$*, that minimizes:*

$$\begin{aligned}\mathbf{f}(\mathbf{x}) &= (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})) \in F \\ &\textit{with } \mathbf{x} = (x_1, x_2, \ldots, x_n) \in X,\end{aligned} \tag{1.2}$$

where $X \subseteq \mathbb{R}^n$ is the n-dimensional decision space, $F \subseteq \mathbb{R}^m$ is the m-dimensional objective space. Both decision and objective space are real spaces, as they correspond to continuous variables and objectives for the proposed applications.
For solving the optimization problem in Eqn. 1.2, a quality measure for comparing different solutions is needed. A partial ordering among different solutions can be found by the *dominance criterion* as illustrated in Fig. 1.1.

Definition 2 *A solution* $\mathbf{a} \in X$ *is dominating a solution* $\mathbf{b} \in X$ *(*$\mathbf{a} \succ \mathbf{b}$*) if and only if it is superior or equal in all objectives and at least superior in one objective. This can be expressed as:*

$$\begin{aligned}\mathbf{a} \succ \mathbf{b},\ \textit{if} \quad & \forall\, i \in \{1, 2, \ldots, m\} : \ \ f_i(\mathbf{a}) \leq f_i(\mathbf{b}) \\ & \wedge \exists\, j \in \{1, 2, \ldots, m\} : \ \ f_j(\mathbf{a}) < f_j(\mathbf{b})\end{aligned} \tag{1.3}$$

Definition 3 *The solution* $\mathbf{a}$ *is indifferent to a solution* $\mathbf{c}$*, if and only if neither solution is dominating the other one.*

When no a priori preference is defined among the objectives, dominance is the only way to determine, if one solution performs better than the other [40]. Furthermore,

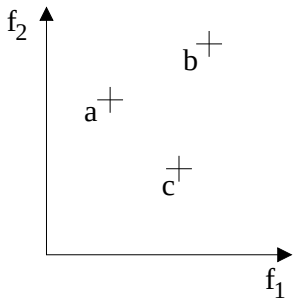

Figure 1.1: Illustration of the dominance principle for a two-objective minimization problem. The solution $\mathbf{a}$ *is dominating* $\mathbf{b}$*, since* $\mathbf{a}$ *is superior in both objective values.* $\mathbf{a}$ *is indifferent to* $\mathbf{c}$*, since in a mutual comparison, each solution is superior in one objective.*

the best solutions to a multi-objective problem are the nondominated subset among all feasible solutions in F. These solutions are denoted as the *Pareto set* [84]. Depending on the relationship between the different objectives, one or several Pareto solutions may exist.

Definition 4 *The relationship between two objectives is defined as correlated, if the corresponding Pareto set contains only one solution; it is defined as conflicting, if more than one solution is in the set.*

The relationship between objectives is illustrated in Fig. 1.2. In this definition, the relationship depends only on the number of solutions in the Pareto set. For correlated objectives, there exists only one Pareto solution. Optimizing one objective concurrently optimizes the other objective. Thus, this optimization problem is similar to a single objective problem. For conflicting objectives, there exists no best solution, but a set of Pareto solutions that represent different trade-offs between the objectives. If the optimization is converged to a Pareto solution, one objective can only be improved at the expense of at least one other objective. A more extended discussion about the relationship between objectives is given by Purshouse and Fleming [87].

1.1.3 Targets

When optimizing problems with conflicting objectives, two different targets are mainly persuit. The choice between the targets depends on the decision maker and

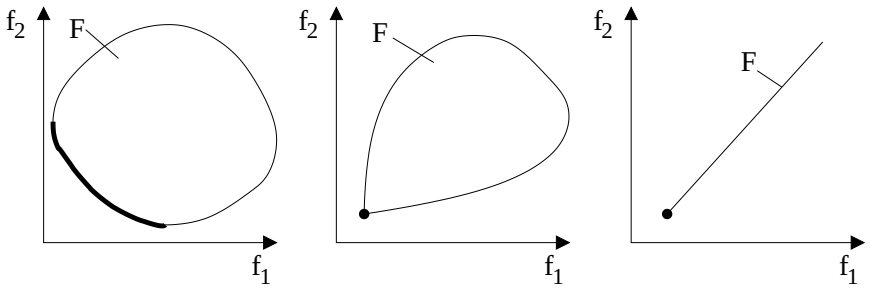

Figure 1.2: Illustration of the relationship between the objectives for a two-objective minimization problem. The Pareto set for the set of all feasible solutions F is marked in each figure buy a bold line. Two objectives are defined as conflicting, if their Pareto front consists of several solutions (left). They are correlated, if the Pareto front consists of a single solution (middle and right).

his or her knowledge or interest about the problem to optimize:

> *Search for a compromise solution:* The decision maker can specify preference relations between the objectives by, e.g., weighting or formulating bounds for the objectives. This information is used to aggregate all objectives into a single figure of merit that will be optimized. The result is a best matching compromise solution for the given information.
>
> *Pareto optimization:* If no preference for the objectives is available or the decision maker is interested in the shape of the Pareto front, then the optimization should search for an approximation of the Pareto set. This optimization is challenging twofold. On one hand, the optimization algorithm must be able to converge sufficiently fast towards the Pareto front. On the other hand, the optimization must preserve diversity in order to be converge to solutions on the whole Pareto front.

The Pareto optimization is usually the more expensive target as several Pareto solutions are searched, but provides more information to the decision maker.

1.2 Optimization Algorithms

1.2.1 Classification

Goldberg [49] classifies the optimization algorithms in three main classes as illustrated in Fig. 1.3: gradient-based, enumerative and guided random algorithms. *Gradient-based* techniques use gradient or higher order derivative information

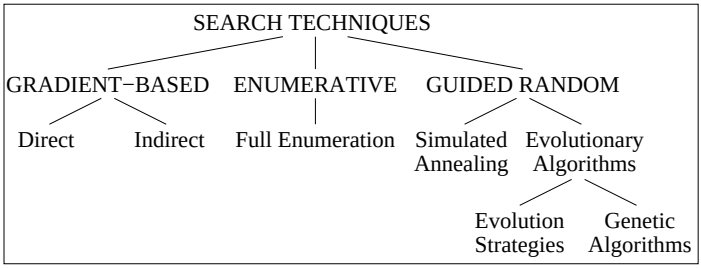

Figure 1.3: Classification of optimization algorithms

about the function to optimize. Here we distinguish direct and indirect techniques. Indirect methods compute the position of the minima by differentiating the objective function and setting the obtained gradient equations to zero:

$$\frac{\partial f}{\partial x_i} = 0, \quad i \in \{1, 2, \ldots, n\}\,, \tag{1.4}$$

while fulfilling the sufficient condition

$$\frac{\partial^2 f}{\partial x_i \partial x_j} > 0, \quad i, j \in \{1, 2, \ldots, n\}\,. \tag{1.5}$$

Indirect gradient-based methods require the mathematical equations of the objective functions. Furthermore, the functional relationship has to be such that it can be solved by hand or a computer programs supporting symbolic calculations. Thus, indirect approaches are considered for test functions, but are not considered for the real-world optimization problems in this thesis, since these problems are given either by an experimental setup or complex numerical simulations.
Direct gradient-based methods converge iteratively to the optimum. For a given starting point, the gradient is computed and used as direction for successive search

points. Here, the mathematical equations of the objective function are not required as the gradient can be approximated, e.g., by finite differences.
Enumerative methods evaluate the function to optimize at every point in the search space. Real-valued search spaces are usually discretized. Full enumeration is the most expensive technique in terms of number of function evaluations. It is only applicable to search spaces with a limited number of feasible points. However, since each point in the search space is evaluated, there is a guaranty to find the optimum. Another expression for this method is area bombing.
Guided random methods use random processes to find the optimum. They are also referred to as semi-stochastic algorithms. Many semi-stochastic methods are inspired by nature, since nature operates with random processes (e.g., for mutating genetic information, within the annealing process of metal, in molecular dynamics, or in swarm behaviors of birds). Well-known representatives of semi-stochastic methods are Evolutionary Algorithms [9] and Simulated Annealing [64].

1.2.2 Direct Gradient-Based Methods

Direct gradient-based methods use gradient information of the objective function for determining the direction of the following search points. Most of these methods are deterministic.
For a given start point $\mathbf{x}_0$, the objective function $f(\mathbf{x}_0)$ and its gradient $\mathbf{g}(\mathbf{x}_0) = \nabla f(\mathbf{x}_0)$ are calculated. The gradient is used to determine the search direction $\mathbf{s}$ for the following one-dimensional (1D) search. Often, the search direction is either the negative gradient $\mathbf{s} = -\mathbf{g}(\mathbf{x}_0)$ itself (steepest descent) or results from an iteratively approximation of second order derivative information (Hessian matrix). The latter is also referred to as quasi-Newton methods and one example is the Broyden-Fletcher-Goldfarb-Shannon (BFGS) algorithm (see, e.g., [28]). The line search minimizes the objective function along the line $\mathbf{x}_0 + \alpha\mathbf{s},\ \alpha \in \mathbb{R}^+$, determined by the starting point and the search direction. Equivalently, the line search could be written as find $\min_{\alpha\in\mathbb{R}^+} f(\mathbf{x}_0 + \alpha\mathbf{s})$. One line search method is the golden section method (see, e.g., [101]), that divides the line subsequently by a golden section around the currently best solution. The best solution from the line search is the start point for the next gradient calculation.
Gradient-based methods rely on derivative information of all objectives and all constraints for determining the search direction of the optimization. The simplest approach for obtaining derivatives is the finite differencing with forward differ-

ences:

$$g_i = \frac{f(\mathbf{x}_o + \Delta\, \mathbf{e}_i) - f(\mathbf{x}_0)}{\Delta}, \tag{1.6}$$

where g_i is the partial derivative of f in the space direction i, Δ is the length of the finite step and $\mathbf{e}_i$ is a unit vector in space direction i. However, finite differences are strongly affected by noise in the function evaluation and are very sensitive the chosen finite difference [14]. Detailed analysis of the effect of different step sizes are given in Martins *et al.* [76].
For computer programs, derivatives can also be obtained by the following three alternatives, which all require access to the source code of the programs:

- In the adjoint formulation (see, e.g., [92]), the underlying equations of f are differentiated and programmed. For complex flow problems as in the considered compressor optimization in Chapter 11, adjoint formulations are rarely available, since deriving and implementing the adjoint equations is a difficult and time-consuming task [72].

- For automatic differentiation [12], software libraries are developed that differentiate the source code and result in an extended code, that computes the objective function also its derivative. This method is usually much more resource intensive than the adjoint method, but the software for the derivatives is obtained in an automated fashion.

- The complex-step method [76] bases on a Taylor series expansion of the objective function f. Instead of using a real step h for the expansion, a purely imaginary step ih is added ($i = \sqrt{-1}$). We write the Taylor series expansion for a single variable x as:

$$f(x + ih) = f(x) + ihf'(x) - h^2 \frac{f''(x)}{2!} - ih^3 \frac{f'''(x)}{3!} + \dots \tag{1.7}$$

  Taking solely the imaginary part of Eqn. 1.7 and solving the equation for $f'(x)$ yields:

$$f'(x) = \frac{\mathrm{Im}(f(x + ih))}{h} + h^2 \frac{f'''(x)}{3!} + \dots \tag{1.8}$$

  By neglecting the third order derivative in Eqn. 1.8, the gradient $f'(x)$ is of O (h^2). The key difference in the complex-step method compared to

finite-differencing is that the derivative results from a single evaluation of the objective function and not from a difference of two evaluations thus circumventing the problem of noise resulting from such differentiation. However, in the complex step method, all real numbers in the source code must be replaced by complex numbers and thus computing a single sensitivity is about twice as expensive as computing the objective function.

Summarizing, the reliable computation of the gradient is most important for the success of gradient-based methods. The line search is less problematic since robust methods like the golden section method just compare the objective value of different solutions and thus are more robust to noise.

1.2.3 Evolutionary Algorithms

Evolutionary Algorithms (EAs) [9] are representatives of the class of stochastic optimization algorithms and do not require gradient information. They are considered as robust optimization algorithms [7], since they are able to cope a wide variety of problem features: discrete and continuous decision variables, noise in the objective function, multimodality, discontinuous objective functions. Most of these features are difficult for gradient-based methods [101]. However, EAs are also considered computationally expensive in terms of the number of evaluated solutions required for convergence.

EAs are inspired by the principles of natural evolution to find an optimal solution to a problem. Natural evolution is driven by the principles of fitness-based selection and recombination/mutation of genetic information. In nature, individuals in a population, which are well adapted to their environment, are likely to survive the natural selection process. These individuals can become parents and their offspring are likely to spread in the following generations. The genetic information of the offspring is either a copy of the genes of a single parent or results from the mating process of multiple parents (recombination) by copying gene sequences from the parents. Genetic information of offspring also includes minor modifications due to reproduction error and some random mutation. In an engineering environment, the genetic information are the decision variables, which specify the properties of a solution to the engineering optimization problem. The fitness of the solution is determined by the objective function.

According to Bäck *et al.* [7], most EAs are based on three independent but related roots: Evolution Strategies, Genetic Algorithms and Genetic Programming.

Evolution Strategies (ESs) were initially developed by Rechenberg [90] and

Schwefel [100, 101] for continuous decision variables. Each individual in the population is represented by a vector of real decision variables. In ESs, mutation is performed by adding a normally distributed random number with mean zero and a certain standard deviation to the variables. Mutation is considered the main operator and the adaptation of the standard deviation is of core importance. Furthermore, individual standard deviations and correlation information can be included for the decision variables in the mutation operator (see, e.g., [5, 52]).
Genetic Algorithms (GAs) were proposed by Holland [54]. In the most popular canonical GA, an individual is represented by a string of bits. For continuous problems, the decision variables are decoded into the binary string. Mutation and recombination is performed by flipping bits or exchanging substrings of different parents, respectively. The probability of mutating a bit is fixed to a low number (about one bit per individual) and the recombination operator is considered the key operator [9].
Evolutionary Programming (EP) was introduced by Fogel [36] and represents individuals similar to EAs in a real-valued vector. EP was initially designed for predicting output values for a sequence of inputs.
EAs will be discussed in detail for single-objective problems in Section 2 and for multi-objective problems in Section 3.

1.2.4 Related Algorithms

Simplex

The simplex method of Nelder and Mead [83] in 1965 is an example of a non-gradient, deterministic method. The method starts with $n + 1$ initial points, where n is the number of decision variables. The start points are arranged equidistant and are the vertices of a simplex. For $n = 2$, the simplex is a triangle, for $n = 3$, a tetrahedron and in general a polyhedron. Starting with an initial simplex, the vertex with the worst objective value is replaced by reflection on the midpoint of the other vertices. Depending on the resulting objective function, the simplex is expanded, contracted or shrinked. The algorithm is widely used, however, recently the algorithm has been shown to be slow for some problems and sometimes fails even on convex functions [121]. Thus various modifications have been proposed.

Simulated Annealing

Simulated Annealing (SA) is inspired by the annealing process of steel. Steel is heated to a temperature above the melting point and cooled in a controlled process. The cooling process determines the crystal structure and the crystal size, which are important for the strength of the steel. In the beginning of the annealing process, atoms are likely to change between different crystal structures and their internal energy might increase or decrease. At a later stage (lower temperatures) almost no change occurs, especially no change that would increase the internal energy, thus always the lowest energies are kept.
In SA, this annealing process is transferred to optimization. The SA algorithm operates with two solutions $\mathbf{x}, \mathbf{x}' \in X$. The algorithms starts by setting $\mathbf{x}$ to a given point in the decision space or at random. A new solution $\mathbf{x}'$ is created in the neighborhood of $\mathbf{x}$ by some random process (e.g., by creating a random vector taken from a normal distribution with $\mathbf{x}$ as mean and a small variation). If the function value of the new solution $f(\mathbf{x}')$ is equal or better than $f(\mathbf{x})$, then set $\mathbf{x} = \mathbf{x}'$. If the function value of the new solution $f(\mathbf{x}')$ is worse than $f(\mathbf{x})$, then with a certain probability set $\mathbf{x} = \mathbf{x}'$, otherwise leave $\mathbf{x}$ unchanged.
While in the real annealing process atoms may change from lower to higher energy levels, in SA, the worse of two solutions may be selected. This selection may help to overcome local minima by adding some randomness to the selection. Similar to the real annealing, the probability of changing to a higher energy level decreases over time.

Hybrid Methods

Hybrid methods combine different optimization algorithms in order to exploit their different. For example, evolutionary algorithms are often combined with direct, gradient-based methods. Evolutionary Algorithms are used in the beginning of an optimization to overcome local minima. Then, the direct, gradient-based methods are used to search the neighborhood of the best solution(s) found, since they are usually faster on smooth and unimodal problems.
Hybrid methods is a wide field of research and an overview is given in Part D3 of [6]. However, a prerequisite is that the different optimization algorithms are applicable to the given problem.

Surrogate Approach

Given that the several solutions to the optimization problem are already evaluated, a surrogate of the objective functions can be constructed. Then, the evaluation of the objective function can be replaced by surrogate evaluations. Surrogates are especially interesting for expensive objective functions, since the necessary computational effort to build the surrogate should be smaller than the expense of the objective function evaluation. They have been successfully applied to complex problems such as a helicopter blade optimization [14] and in turbo machinery design [61]. An overview on how evolutionary algorithms can be coupled with surrogates will be given in Section 4 and a new implementation will be proposed in Section 8.

Chapter 2

Evolution Strategies for Single Objective Optimization

A set of different evolution strategies is presented. These strategies will be used for the optimization of the gas turbine design problems. First the common properties of all considered strategies are presented. Then, each strategy is described in detail. A focus is on how these strategies learn in order to adapt the mutation distribution.

2.1 Introduction

In Section 1.2, Evolution Strategies (ESs) have been introduced as representatives of stochastic optimization algorithms. ESs imitate the principles of natural evolution to find an optimal solution to an optimization problem. An ES is determined by a selection, recombination, and mutation operator. For all considered ESs, these three operators are successively processed as given by the iterative loop in Fig. 2.1.

The evolutionary optimization starts by generating an initial population of individuals at a generation $g = 0$. An individual i in the population consists of a vector of decision variables $\mathbf{x}_i \in X$ where $X \subseteq \mathbb{R}^n$ is the n-dimensional decision space. The population consists of λ individuals and can be described by $P_\lambda^g = \{\mathbf{x}_i\}_{i=1,\dots,\lambda}$. In the evolutionary optimization, the population evolves over several generations g. Within each generation, the selection, recombination, and mutation operators are applied for different aims. The selection operator increases the fitness in the population by selection on average fitter individuals. Thus, selection decreases on average the diversity in the population. The recombination operator exchanges information in the population in order to spread good properties of solutions. Finally, the mutation operator increases diversity in the population by adding random variations to the decision variables.

The quality of an evolutionary algorithms depends highly on the interplay of these

operators, as some combinations of recombination and mutation operators may even lead to divergence [69]. In ES, the focus is on the mutation operator [9], which is often such that it adapts while converging towards the optimum. Adaptation is necessary for fast convergence as shown by Rechenberg [90]. For simple objective functions, Rechenberg computed the ideal mutation variance as a function of the distance of solutions to the optimum.

Selection Operators

In each generation g, a fraction of the population is selected based on their fitness. The selected set serves as parent population P_μ^{g+1} for the next generation. In ESs, the deterministic selection of the μ best individuals is most popular. For minimization problems, "better" refers to lower objective values. A selection operator that selects solutions from the current population *and* their parent population is denoted as $(\mu + \lambda)$-strategy. This operator is elitist by preserving always the best solutions obtained so far and thus avoids a deterioration of the fitness in the parent population. This ensures a constant improvement of the fitness of the parents.
In contrast, selecting the parent population solely from the current population P_λ^g is referred to as (μ, λ)-strategy. In this strategy, the fitness of the parent population increases or decreases between two generations, depending on the fitness of the current population. While on the first view, a decrease in the parent fitness seems unfavorable for finding the optimum, the (μ, λ)-strategy ensures a constant change in the parent population. This is shown to be advantageous for adaptive strategies [5] and is a necessity for noisy objective functions, in order to avoid stagnation at noisy solutions [4].
A more general formulation of the selection operator is the (μ, κ, λ)-strategy [7], which introduces a maximal lifetime $1 \leq \kappa \leq \infty$. For $\kappa = 1$ and $\kappa = \infty$, the (μ, λ) and $(\mu + \lambda)$-strategy are obtained, respectively.

Recombination Operators

In nature, evolution is a highly parallel process by using large populations of individuals. Most species recombine their genetic information by mating two parents. Mating allows to spread genetic information in the population and the offspring might benefit from combining preferable properties of their parents.
In ES, the decision variables represent the genetic information and mating is denoted as recombination. Several different recombination operators exist. Operators that combine ρ parents denoted as $(\mu/\rho \overset{+}{,} \lambda)$ strategies. For each new solution,

ρ parents are chosen at random. Special cases are $\rho = 2$ (binary recombination) and $\rho = \mu$ (global recombination).
For *global recombination*, the mean of all parents $\mathbf{x}_{j,j=1,\ldots,\mu}$ is computed:

$$x'_i = \frac{1}{\mu} \sum_{j=1}^{\mu} x_{j,i}. \tag{2.1}$$

For *binary recombination*, we will consider two operators from [9] to obtain a recombined individual $\mathbf{x}'$ from two parents $\mathbf{x}_1$ and $\mathbf{x}_2$. The first approach is discrete recombination:

$$x'_i = \begin{cases} x_{1,i} & \text{if } p < 0.5, p \sim \mathcal{U}(0,1) \\ x_{2,i} & \text{otherwise,} \end{cases} \tag{2.2}$$

where $\mathcal{U}(0,1)$ is a uniform distribution of random number between 0 and 1. In the second operator, the parents are linearly recombined by:

$$x'_i = \alpha x_{1,i} + (1-\alpha) x_{2,i}, \tag{2.3}$$

where $\alpha \in \mathcal{U}(-0.5, 1.5)$. This recombination operator inter- and extrapolates parents.

Mutation Operators

Mutation is a process of adding random changes to the genetic information. This enables a population to adapt to changing environmental conditions.
In ES, mutation is usually performed by adding a normally distributed random vector $\mathbf{z}$ [101, 5, 52] to a recombined solution $\mathbf{x}'$:

$$\mathbf{x}'' = \mathbf{x}' + \mathbf{z}, \;\; \mathbf{z} \sim \mathcal{N}(0, \mathbf{C}), \tag{2.4}$$

where $\mathcal{N}(0, \mathbf{C})$ is a multivariate normal distribution with expectation $E = 0$ and covariance matrix $\mathbf{C}$. In the simplest case, an *isotropic mutation* distribution is chosen:

$$\mathbf{z} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}) = \sigma \mathcal{N}(0, \mathbf{I}) \tag{2.5}$$

where σ is the standard deviation of the normal distribution and $\mathbf{I}$ is the identity matrix. For isotropic mutation, the mutation of all decision variables is uncorrelated and of equal standard deviation σ. The standard deviation σ is also referred

to as *strategy parameter*. The probability density function is given in Fig. 2.2. Lines of equal probability are circles for $n = 2$ variables and (hyper-)spheres for $n \geq 3$. The previous equation for isotropic mutation can also be written as:

$$z_i \sim \sigma \mathcal{N}(0,1), \quad i = 1, \ldots, n. \tag{2.6}$$

An extension of the isotropic mutation is obtained by using for each decision variable $\mathbf{x}_i$ an *individual step size* σ_i. The probability distribution of this mutation is similar to a stretching or compressing of the isotropic distribution parallel to the coordinate axis of the decision space and is illustrated in Fig. 2.2. Lines of equal probability are (hyper-)ellipses with their principal axis parallel to the coordinate axis. The resulting covariance matrix is now diagonal ($\mathbf{C} = \text{diag}\{\sigma_i^2\}$) and mutation can also be written as:

$$z_i \sim \sigma_i \, \mathcal{N}(0,1), \quad i = 1, \ldots, n. \tag{2.7}$$

Arbitrary normally distributed mutation is obtained by non-zero non-diagonal elements in the covariance matrix. The probability distribution is given in Fig. 2.2 and results from linear transformation (rotation, scaling) of an isotropic probability distribution. The mutation of the decision variables is now correlated and thus referred to as *correlated mutation*. Due to the symmetry of the covariance matrix, $\frac{n(n+1)}{2}$ elements of the covariance matrix need to be specified. The elements are denoted as strategy parameters and consists of n step sizes (diagonal elements $c_{ii} = \sigma_i^2$) and $\frac{n(n-1)}{2}$ correlation factors (non-diagonal elements $c_{ij} = c_{ji}$, $i \neq j$). Going from isotropic mutation via individual step sizes to correlated mutation increases the number of free strategy parameters from 1 via n to $\frac{n(n+1)}{2}$. These parameters can be set manually or can be learned as described in Sections 2.2 to 2.4. The choice of the mutation distribution is problem dependent and is a compromise between minimizing the learning effort and being able to adapt to mis-scaling or correlated decision variables. Adaptation of the mutation distribution has been show to be necessary for efficient optimization algorithms as constant distributions are far from being optimal [5].

2.2 One-Fifth Success Rule

The two-membered $(1 + 1)$-strategy was the first evolution strategy developed by Rechenberg in 1964 [90] and consists of a single parent and offspring. The strategy implements a selection and mutation operator, but no recombination operator,

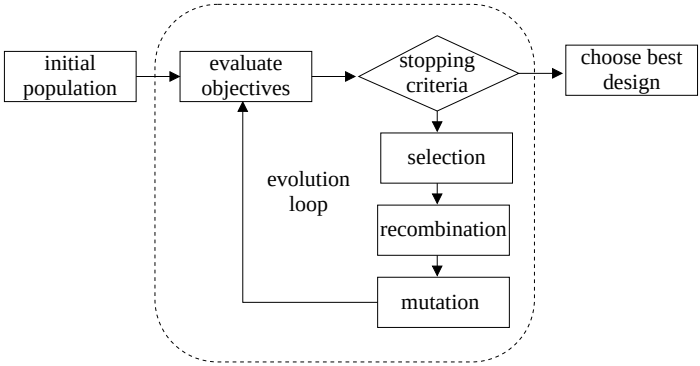

Figure 2.1: Illustration of the optimization procedure for an evolution strategy

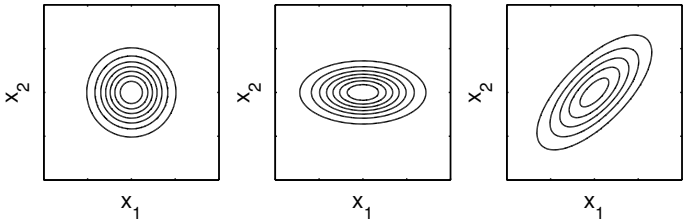

Figure 2.2: Lines of equal probability for isotropic (left), scaled (middle) and correlated (right) mutation

since just one parent exists. The mutation operator varies the parent by isotropic mutation (Eqn. 2.6) with a single step size σ. The mutation is successful, if the objective function value of the offspring is better than the value of the parent. Then, the parent is replaced by the offspring.
For two simple objective functions, Rechenberg [90] computed theoretically the ideal success rate and step size for different objective functions. He concluded that the ratio of successful mutations should be one fifth and formulated the adaptation of the step size σ as a function of the success (*1/5 success rule*):

> In an optimization, the success rate should be $1/5$. If, it is greater than $1/5$, then increase σ; if it is less, then decrease σ.

Schwefel [101] proposed an implementation of Rechenberg's rule. He stated that the ratio of success should be computed always after generating n offspring, where n is the number of decision variables. The success rate p_s should be measured over the last $10n$ computed offspring. Then, the step size is adapted following Rechenberg's rule by:

$$\sigma = \begin{cases} c\,\sigma & \text{, if } p_s < 1/5 \\ \sigma & \text{, if } p_s = 1/5 \\ \sigma/c & \text{, if } p_s > 1/5, \end{cases} \tag{2.8}$$

where c is a multiplicative factor for increasing and decreasing the step step size. Schwefel [101] showed that Rechenberg's theory is valid for $c = 0.817$ and proposed to set $0.817 \leq c \leq 1$. For $c = 1$ the step size σ is constant.
Furthermore, the success rate p_s should be decreased, if the objective function is noisy ($p_s = 1/14.8$ [90]), constraint [102] or local mimima exist [102]. Rechenberg's success rule was a major progress in the adaptation of a single step size, but different methods have to be applied for non-isotropic mutation.

2.3 Self-adaptation

The $1/5$ success rule of Rechenberg introduced a deterministic rule for adapting the step size of an isotropic mutation distribution. The rule could be regarded as an external control element and is limited to a single strategy parameter. In order to avoid external control, Rechenberg [90] proposed that the values of multiple strategy parameters could result from a "learning population". Strategy parameters such as step sizes and correlation information could be subject to random mutation, similar to the mutation of the decision variables. Now, each individual in the population carries a vector of strategy parameters in addition to its vector of decision variables. This approach assumes that individuals with well-adapted strategy parameters are more likely to generate offspring that get selected in the next population and thus these strategy parameter will be dominant. Rechenberg formulated and tested successfully how to learn individual step sizes. He also mentioned the possible learning of correlation factors for the decision variables or of parameters of the recombination operator. Rechenberg argued that this approach is more robust than the $1/5$ success rule, since it can handle discontinuities in the derivative and nonisotropic mutation distributions can be learned.
Schwefel [101] modified Rechenberg's approach and extended it to correlated mutation. In addition to the individual step sizes of Rechenberg, rotation angles were

introduced for adapting arbitrary normal distributions. He denoted his adaptation principle as *self-adaptation*. While Schwefel developed self-adaptation for mutation control in evolution strategies, today, self-adaptation is associated with a variety of different operators , e.g., with the adaptation of the mutation and recombination rates in GA [8, 5] and in differential evolution [1].
In the following we outline the basic principles of self-adaptive, correlated mutation and refer to the terminology of Schwefel and Rudolph [103]. This mutation is also referred to as *rotating angle mutation* and abbreviated in the following as ROT-ES. Correlated mutation requires the adaptation of arbitrary normal distribution. Samples from arbitrary normal distributions can be obtained by first creating a random vector $\mathbf{z}$, which is then rotated by multiplying $\mathbf{z}$ with a rotation matrix $\mathbf{T}$. For the random vector $\mathbf{z}$, n individual step sizes σ_i are required. The rotation matrix $\mathbf{T}$ requires $\frac{n(n-1)}{2}$ rotation angles α_k.
Each individual in the population is encoded by a vector of decision variables and strategy parameters $\{\mathbf{x}, \{\sigma_i\}_{i=1,\ldots,n}, \{\alpha_k\}_{k=1,\ldots,\frac{n(n-1)}{2}}\}$. In self-adaptation, the mutation of the strategy parameters is performed similarly to the mutation of the decision variables by:

$$\sigma_i'' = \sigma_i' \exp(\tau_0 z_0 + \tau z_i) \quad \text{, with } z_0, z_i \sim \mathcal{N}(0,1) \tag{2.9}$$

$$\alpha_k'' = \alpha_k' + \beta z_k \quad \text{, with } z_k \sim \mathcal{N}(0,1) \tag{2.10}$$

where τ_0, τ and β are the learning rates, and recommended values are:

$$\tau_0 = \frac{1}{\sqrt{2n}}, \quad \tau = \frac{1}{\sqrt{2\sqrt{n}}}, \quad \beta = 5^\circ \tag{2.11}$$

After mutating the strategy parameters, the decision variables are mutated with:

$$\mathbf{x}'' = \mathbf{x}' + \mathbf{T}\mathbf{z}, \text{ , with } \quad z_i \sim \sigma_i \mathcal{N}(0,1),\ i = 1, \ldots, n \tag{2.12}$$

$$\mathbf{T} = \prod_{i=1}^{n-1} \prod_{j=i+1}^{n} \mathbf{T}_{ij}$$

The matrices $\mathbf{T}_{ij}$ are rotation matrices and are identical to the identity matrix, except that the four elements with the positions $\{i,i\}$, $\{i,j\}$, $\{j,i\}$, $\{j,j\}$ of the

matrix are changed:

$$\mathbf{T}_{ij} = \begin{pmatrix} 1 & 0 & & & & & \cdots & & & & & 0 \\ 0 & 1 & & & & & & & & & & \\ & & \ddots & & & & & & & & & \\ & & & 1 & & & & & & & & \\ & & & & \cos(\alpha_k) & & & & -\sin(\alpha_k) & & & \\ & & & & & 1 & & & & & & \\ \vdots & & & & & & \ddots & & & & & \vdots \\ & & & & & & & 1 & & & & \\ & & & & \sin(\alpha_k) & & & & \cos(\alpha_k) & & & \\ & & & & & & & & & 1 & & \\ & & & & & & & & & & \ddots & \\ & & & & & & & & & & 1 & 0 \\ & 0 & & & & & \cdots & & & & 0 & 1 \end{pmatrix}, \tag{2.13}$$

with $k = 1/2(2n-i)(i+1) - 2n + j$. In literature, self-adaptive mutation is also found without correlation (i.e., without the rotation matrix $\mathbf{T}$) or isotropic mutation (i.e., all individual step sizes are replaced by a single one with $\sigma_i \equiv \sigma$).

Self-adaptive mutation is sensitive to the chosen population size, recombination operator, and selection operator [69, 52]. Typically, a (μ, λ) strategy is used with $\lambda/\mu = 7$ [7] and in particular a $(15, 100)$. However, Hansen and Ostermeier [52] recommend to set the number of parents proportional to the number of strategy parameters. Thus, the number of parents is proportional to n for individual step sizes and approximately $n^2/2$ parents are needed for correlated mutation.

No clear statement can be given about the best recombination operator in conjunction with self-adaptation. Kursawe [69] shows that self-adaptation can fail for specific recombination operators. Our preferred method is global recombination of the decision variables and individual step sizes. No recombination is applied to the rotation angles.

Schwefel's rotating angle mutation includes two main disadvantages. First, the performance of the mutation operator is dependent on the orientation of the coordinate system. Second, the rotation matrix $\mathbf{T}$ depends on the sequence of rotation matrices $\mathbf{T}_{ij}$. Thus, the rotation angles are not independent.

Bäck [5] overcomes the latter problem. His self-adaptation adapts directly all elements of the covariance matrix instead of using the rotation matrix $\mathbf{T}$. The covari-

ance matrix is constructed by the n standard deviations σ_i and the $\frac{n(n-1)}{2}$ rotation angles α_k with [1] :

$$\begin{aligned} c_{ii} &= \sigma_i^2 \\ c_{ij,\ i\neq j} &= \tanh(2\alpha_k)\sigma_i\sigma_j, \text{with } k = 1/2(2n-i)(i+1) - 2n + j \end{aligned} \tag{2.14}$$

The decision variables are now mutated with:

$$\mathbf{x}'' = \mathbf{x}' + \mathbf{z}, \text{ , with } \mathbf{z} \sim \mathcal{N}(0, \mathbf{C}), \tag{2.15}$$

Mutation now requires sampling random vectors from the covariance matrix. This is described in the next section (Section 2.4).

2.4 Covariance Matrix Adaptation

The Covariance Matrix Adaptation (CMA) describes a deterministic rule for adapting the covariance matrix $\mathbf{C}$ for correlated mutation. It was developed by Hansen and Ostermeier and we refer to their publication from 2001 [52]. The mutation vector $\mathbf{z}$ for the decision variables is computed from a global step size σ and a covariance matrix $\mathbf{C}$:

$$\mathbf{z} \sim \sigma\mathcal{N}(0, \mathbf{C}) \tag{2.16}$$

A sample of the mutation vector $\mathbf{z}$ can be obtained by first computing a vector from the uncorrelated normal distribution, which is then multiplied by the global step size and a linear transformation with:

$$\mathbf{z} \sim \sigma\mathbf{B}\mathbf{D}\mathcal{N}(0, \mathbf{I}) \text{ with } \mathbf{C} = (\mathbf{B}\mathbf{D})\,(\mathbf{B}\mathbf{D})^T, \tag{2.17}$$

where $\mathbf{B}$ is an orthonormal matrix of eigenvectors from $\mathbf{C}$, and $\mathbf{D}$ is a diagonal matrix containing the square root of the eigenvalues of $\mathbf{C}$.
Assuming the necessary information for adapting the strategy parameters σ and

[1] In Eqn. 2.14, we differ from Bäck [5], who computes the non-diagonal elements as $c_{ij,\ i\neq j} = \frac{1}{2}\tanh(2\alpha_k)\left(\sigma_i^2 - \sigma_j^2\right)$. This equation does not fulfill the mathematical limit for the non-diagonal elements of a covariance matrix with $||c_{ij}|| \leq ||\sigma_i\sigma_j||$. In Bäck's equation, c_{ij} becomes even infinite for $\alpha = 45°$.
Denoting α as a rotating angle is misleading, since it scales the correlation between the individual step sizes σ_i. Furthermore, a rotation of the covariance matrix would change the variance of $\mathbf{C}$ in the coordinate axis direction. In Bäck's approach, however, the individual step sizes are unchanged for arbitrary α.

$\mathbf{C}$ is of the same order as the strategy parameters, the scalar σ will be adapted much faster than the matrix $\mathbf{C}$. Thus, the mutation distribution can react fast to a changing optimal step size, while the adaptation to the topology of the objective function is much slower.
For CMA, a (2/2,10) selection operator is often selected as in [52, 80] and the two parents are recombined by computing the mean for each decision variable. For the adaptation of the mutation distribution, CMA uses at each generation g only the information included in the position of the recombined parent $\mathbf{x}'^{(g)}$. The motion of the recombined parent is tracked over previous generations and denoted as evolution path. In each generation, the evolution path is compared to the path that result from random selection. Comparing the two paths indicates efficient mutation distributions. If the evolution path is smaller than the random path, smaller mutations have been more efficient in the past generations and the variance of the mutation distribution is reduced. If the evolution path is larger than the random path, the variance is increased. Furthermore, the orientation of the mutation distribution is updated by adding the evolution path as a new sample for the construction of the covariance matrix.
First, we consider the adaptation of the covariance matrix. The evolution path $\mathbf{p}_\mathrm{c}$ and the covariance matrix $\mathbf{C}$ are updated by:

$$\begin{aligned} \mathbf{p}_\mathrm{c}^{(g)} &= (1-c)\,\mathbf{p}_\mathrm{c}^{(g-1)} + \\ &\quad \sqrt{c(2-c)}\frac{\sqrt{\mu}}{\sigma^{(g-1)}}\left(\mathbf{x}'^{(g)} - \mathbf{x}'^{(g-1)}\right) \end{aligned} \tag{2.18}$$

$$\mathbf{C}^{(g)} = (1-c_\mathrm{cov})\,\mathbf{C}^{(g-1)} + c_\mathrm{cov}\mathbf{p}_\mathrm{c}^{(g)}(\mathbf{p}_\mathrm{c}^{(g)})^T, \tag{2.19}$$

where c and c_cov determine the decay of information from previous generations. The parameter c is recommended as $c = \frac{4}{n+4}$ and thus for $n \gg 1$ the characteristic time for updating the evolution path is $1/c \approx n/4$. Furthermore c_cov is recommended as $c_\mathrm{cov} = \frac{2}{(n+\sqrt{2})^2}$ and the characteristic time for $\mathbf{C}$ is $1/c_\mathrm{cov} \approx n^2/2$. Similarly, the global step size is adapted by first computing a cumulated path $\mathbf{p}_\sigma$ while omitting the scaling with $\mathbf{D}$ and than comparing the path length with the expected path length $||\mathrm{E}(\mathcal{N}(0,I))|| =: \chi \approx \sqrt{n}(1 - 1/(4n) + 1/(21n^2)$ under

random selection:

$$\begin{aligned} \mathbf{p}_{\sigma}^{(g)} &= (1-c)\mathbf{p}_{\sigma}^{(g-1)} + \\ &\quad \sqrt{c(2-c)}\frac{\sqrt{\mu}}{\sigma^{(g-1)}}\mathbf{B}^{(g-1)}(\mathbf{D}^{(g-1)})^{-1}(\mathbf{B}^{(g-1)})^{-1}(x'^{(g)} - x'^{(g-1)}) \\ \sigma^{(g)} &= \sigma^{(g-1)}\exp\left(\frac{1}{d_{\sigma}}\frac{||\mathbf{p}_{\sigma}^{(g)}|| - \chi}{\chi}\right). \end{aligned} \tag{2.20}$$

The parameter $d_{\sigma} = \sqrt{n}/4$ ($\approx 1/c$) controls the adaptation speed of σ. At the beginning of the optimization ($g = 0$) all the covariance matrix is set unitary ($\mathbf{C}^{(0)} = \mathbf{I}$) and the evolution paths are set to zero ($p_{\mathrm{c}}^{(0)} = p_{\sigma}^{(0)} = 0$).

Chapter 3

Multi-objective Evolutionary Algorithms

Evolutionary Algorithms are *the* standard tool for multi-objective optimization. Their parallel search leads to an approximation of the Pareto front in a single optimization run. This is a major advantage compared to traditional optimization algorithms like gradient-based methods that converge to a single Pareto solution. Furthermore, traditional methods require an aggregation of all objectives to a single figure of merit. This is difficult if the shape of the Pareto front is unknown before optimization.
This chapter starts with the history of multi-objective evolutionary algorithms and gives an overview on state-of-the-art algorithms. Different test problems are introduced and performance measures for comparing different algorithms are added.

3.1 Introduction

Evolutionary algorithms can exploit the population-based feature and converge in parallel to the Pareto front. While optimizing, different solutions in the population converge to different areas of the Pareto front, and thus an approximation of the Pareto front can be obtained in a single optimization run. The research interest has increased over the past twenty years on the development and application of evolutionary algorithms for Pareto optimization. Several promising methods have been proposed and compared by several researchers [125, 116, 20]. An exhaustive list of references can be found on the web page of Coello [19]. Two books are devoted to multi-objective optimization [24, 21]. The various multi-objective evolutionary algorithms are usually distinguished by their fitness assignment operators, while mutation and recombination operators are usually adopted from standard single-objective algorithms.
The first evolutionary algorithm that was designed to converge to multiple Pareto solutions is the Vector Evaluated GA (VEGA) in 1985 [98]. The name of the algorithm results from the optimization of a vector of objectives instead of a scalar

in single objective optimization. For m objectives, the algorithm divides the current population into m fraction, and the selection operator selects solutions in each fraction according to a different objective. A drawback of this algorithm is that it focuses on the ends of the Pareto front as in each selection always one objective is preferred and never compromise solutions.

A more efficient approach was introduced by Goldberg [49] in 1989. He introduced the principle of dominance as selection criterion, but did not implement any algorithm. He also suggested to use fitness sharing [50] for promoting the diversity among solutions in order to prevent the population from converging to a single point on the Pareto front. Different implementations of Goldberg's suggestions were published between 1993 and 1995, e.g., Multi-Objective GA (MOGA) [37], Niched Pareto-GA (NPGA) [56] and Nondominated Sorting GA (NSGA) [109]. Between 1995 and 2000, various evolutionary algorithms have been proposed and compared in literature [38, 55, 125, 20, 117].

In multi-objective optimization, the current set of nondominated solutions contains a large amount of information about the problem to optimize. This information is important for determing the fitness of a solution. As a small example, assume only two solutions are given. In single objective optimization, one solution is better than the other or both are equal. Thus, it is clear, which solution is preferable. In multi-objective optimization, it is very likely that the two solutions are indifferent, i.e., non of the two solutions is dominating the other one. So, which one should be preferred? To determine the preference between two solutions, the current set of nondominated solutions can be very helpful by, e.g., specifying the number of solutions that dominate each of the two solutions. This measure could be chosen to determine the preference.

Elitism, a technique of preserving always the best solutions obtained so far, prevents the algorithm from losing this information. In multi-objective optimization, elitism is performed by preserving the nondominated solutions in an archive [116]. The parents of the next generation are selected from the current population and the archive. The convergence comparison of Zitzler *et al.* [125] in 1999 showed that elitism is beneficial for all considered algorithms. Some researchers state elitism as a necessity for multi-objective optimization [116].

Today, elitist multi-objective algorithms, which base on the dominance criterion and implement some kind of niching criterion are most popular [117]. Two representatives are SPEA2 [123] and NSGA-II [26], which were the most applied algorithms on the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO2003) [39]. However, there are still alternative approaches that might be more suited for some applications as show in some empirical com-

parisons [88].
In this chapter, we focus on multi-objective optimization problems where the decision maker is interested in obtaining an approximation of the Pareto front. The Pareto front gives valuable information about the trade-off between different objectives. Then, the decision maker selects usually one or several solutions from the approximated Pareto front, which will be used, e.g., for building prototypes.

3.2 Selection Operators

We consider multi-objective evolutionary algorithms, performing a population-based search in order to approximate the Pareto set. This approximation can be obtained in a single or in several optimization runs. The various multi-objective evolutionary algorithms mainly employ a selection operator [125, 13]. In the following we classify selection operators in 3 classes as proposed by Horn [55]. Algorithms of the first class approximate the Pareto front by a number of independent runs, with each run containing a different aggregation of the objectives. Algorithms of the second and third class approximate the Pareto front in a single run by a cooperative population search. These classes are distinguished by the use of dominance in the selection operator.
From the Pareto definition, two targets have to be considered when developing a selection operator for Pareto optimization. On one hand, the resulting algorithm must be able to converge sufficiently fast towards the Pareto front, while on the other hand, it must preserve diversity among its population in order to be able to spread over the whole Pareto front.

3.2.1 Independent Sampling

An approximation of the Pareto front can be obtained by performing several independent runs with different aggregation of the objectives by, e.g., a weighted sum or a constraint approach [88]. This leads to a discrete approximation of the Pareto front, with each optimization run converging to a different point of the Pareto front. One implementation of independent sampling is the Constraint Method-based Evolutionary Algorithm (CMEA) [88], which will be discussed in detail.

Constraint Method-based Evolutionary Algorithm

Ranjithan *et al.*[88] proposed to use a Constraint Method-based Evolutionary Algorithm (CMEA). One objective f_h is selected for optimization, while all other

objectives $f_{i,i\neq h}$ are treated as constraints. Thus, the multi-objective optimization problem is formulated as:

$$\text{find } \min f_h, \text{ while } f_i < u_i^t \; \forall \; i = 1, \ldots, m \; ; i \neq h, \tag{3.1}$$

where $u_i^t, \; t = 1, \ldots, k$ are k different constraint values for each objective f_i. The constraint values should be set such that they are within the function values of the Pareto front. According to Horn [55] bounds of the Pareto front can be found by optimizing each objective f_i in a separate optimization run without setting any constraints. The best solution of each of the m optimization runs is chosen. The bounds of the Pareto front are equal to the minimal and maximal objective value of the m solutions for each objective i and are written as $f_i^{\min}$ and $f_i^{\max}$, respectively. Finally, the upper constraints u_i^t are set such that they divide the interval between the bounds of each objective f_i in k intervals $t = 1, \ldots, k$ of equal width:

$$u_i^t \quad = \quad f_i^{\min} + \frac{t}{k}(f_i^{\max} - f_i^{\min}). \tag{3.2}$$

For each possible combination of one constraint value per objective, an optimization run is performed and the Pareto front is approximated.
Knowledge of previous runs can be exploited by using the best solution(s) obtained so far as initial solution(s) for the next run [88].

3.2.2 Cooperative Population Searches with Dominance Criterion

Algorithms from the class of cooperative population searches find an approximation of the Pareto front in a single optimization run. Here, the principle of dominance is chosen as the selection criterion. In addition, a fitness sharing techniques or other density estimation techniques are used in order to preserve diversity in the population. According to Van Veldhuizen and Lamont [117] this class of fitness assignment is most widely used. Recent performance comparisons [16] [88] show however that other classes of optimization approaches can also lead to comparable results. Two of the most prominent representatives are the Nondominated Sorting Genetic Algorithm (NSGA) of Srinivas and Deb [109] with successors NSGA-II [26] and NSGA-IIc [27] and the Strength Pareto Evolutionary Algorithm (SPEA) of Zitzler and Thiele [125] with its successor SPEA2 [124].

Nondominated Sorting Genetic Algorithm I & II

NSGA assigns fitness by a nondominated sorting procedure in conjunction with a fitness sharing technique as described by Goldberg [49]. The sorting starts by as-

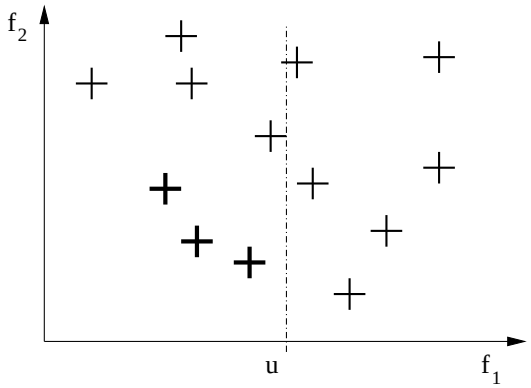

Figure 3.1: Selection by constraint approach for 2 objectives: For f_1 a hard upper bound constraint is set at $f_1 = u$ [dash-dotted lines]. From all solutions [+ symbols], the solutions that do not violate the constraint are selected and from this subset, the μ best solutions [bold + symbols] with respect to f_2 are selected (here $\mu = 2$).

signing rank 0 to the nondominated solutions of the population and removing them from the population. Then, the nondominated solutions of the remaining population are assigned (the next higher) rank 1. This is repeated until all solutions are sorted to a certain rank. An example for a 2-objective problem is given in Fig. 3.2 Within all solutions of a certain rank fitness sharing is computed in the objective space. Fitness sharing promotes solutions in sparse areas.
Selection is performed by a binary tournament. Always two solutions are taken from the current population. If the rank of the two solutions differs, the one with the lower rank is chosen. Otherwise the one with the lower sharing value is chosen in order to promote diversity. The chosen solution is copied into the parent population of the next generation.
The studies of Zitzler and Thiele [125] have illustrated that elitism improves the performance of multi-objective evolutionary algorithms on noise-free test problems as it avoids the loss of information by storing the nondominated solutions. Elitism was implemented into the second version of NSGA (NSGA-II) [26]. In addition the sharing technique was replaced by a crowding distance. The crowding distance of a solution is equal to the smallest edge of the largest hypercube that can be constructed around the solution without including any other solution.

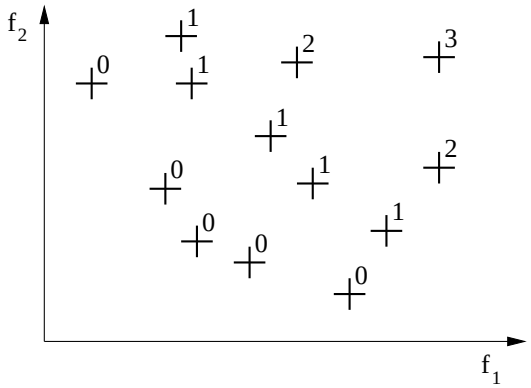

Figure 3.2: Nondominated sorting procedure: The sorting starts by assigning rank 0 to the nondominated solutions of the population and removing them from the population. Then, the nondominated solutions of the remaining population are assigned (the next higher) rank 1.

In NSGA-II, the selection process starts by unifying the current population with its parent population. Then, solutions are selected starting with rank 0. The rank is increased until the number of selected solutions exceeds the given parent population size. Finally, among the solutions with the highest rank, always the solution with the smallest crowding distance is removed until the target size is reached.

Strength Pareto Evolutionary Algorithm 1 & 2

The Strength Pareto Evolutionary Algorithm (SPEA) of Zitzler and Thiele [125] is a well-established Pareto-optimization algorithm. The advantages and drawbacks of the algorithm have been extensively discussed [125, 122]. The algorithm entails a fitness assignment and selection mechanism based on the concept of elitism. SPEA uses the nondominated solutions for the fitness assignment as illustrated for a 2-objective problem in Fig. 3.3. First, the fitness of each nondominated solution is computed as the fraction of the population that it dominates. The fitness of a dominated individual is equal to one plus the fitness of each nondominated solution by which it is dominated. This fitness assignment guarantees that the fitness of nondominated solutions is always lower than the fitness of the dominated solutions. In addition, it promotes solutions in sparse areas by assigning smaller fitness

values.

In SPEA, elitism is performed by storing the nondominated solutions in an

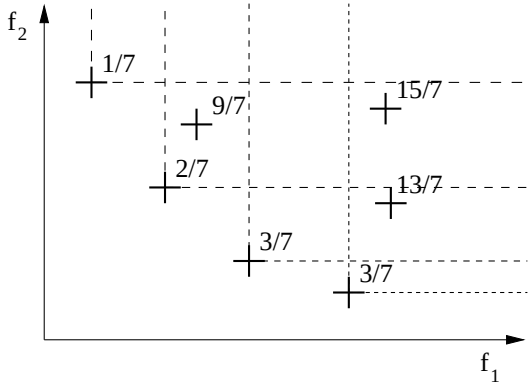

Figure 3.3: Fitness assignment by SPEA for 2 objectives (low fitness is preferred in the selection operator): For each nondominated solution, the fitness is equal the fraction of dominated solutions in the population (here: population size is 7). The fitness of a dominated solution is equal to one plus the fitness of each nondominated solution by which it is dominated. For each nondominated solution, the dominated area is marked with dashed lines.

archive. The archive is updated by always adding a copy of the current population to the archive and removing the dominated. In order to preserve diversity in the archive and to keep its size limited, a clustering algorithm is used. Clustering removes solutions in dense areas as measured in the objective space.

In the selection process, a copy of the current population and the archive is unified in a pot. Always two individuals are taken from the pot and compete in a binary tournament such that the individual with the lower fitness wins and is copied into a new parent population. The next population is generated by applying recombination and mutation operators to the parent population.

In SPEA2 [124], SPEA was extended by changing the fitness assignment and the clustering algorithm in order to improve the algorithm especially for correlated objectives and increase the diversity in the archive, respectively.

3.2.3 Cooperative Population Searches without Dominance Criterion

This class of optimization approaches generates an approximation of the Pareto front in a single optimization run, but does not use the dominance criterion within the selection operator. Moreover, several (local) selections are performed from one population with different aggregation of the objectives. This might be beneficial compared to Independent Sampling, since information can be exchanged in the optimization run by individuals being selected by several different aggregations. One example is the Subdivision Method (SDM) [16], an optimization approach with some similarities to the CMEA.

Subdivision Method

In the objective space, the SDM performs several local (μ, κ, λ) selections (see Section 2.1) and then unifies all selected solutions to the parent population. This is illustrated in Fig. 3.4. Similar to the CMEA one objective f_h is selected for optimization, while all other objectives $f_{i,i\neq h}$ are treated as constraints. However, a lower and upper constraint value is set:

$$\min f_h, \text{ while } l_i^t \leq f_i \leq u_i^t, \; \forall\, i = 1, \ldots, m \,;\, i \neq h. \tag{3.3}$$

The constraints are obtained by first computing the nondominated front, created by the current population and their parent population. For the nondominated front, the minimal and maximal function values $f_i^{\min}$ and $f_i^{\max}$, respectively, are computed. Then, the lower and upper constraints l_i^t and u_i^t are set such that they divide the nondominated front along each objective axis of f_i in k intervals $t = 1, \ldots, k$ of equal width:

$$\begin{aligned} l_i^t &= f_i^{\min} + \frac{t-1}{k}(f_i^{\max} - f_i^{\min}) \\ u_i^t &= f_i^{\min} + \frac{t}{k}(f_i^{\max} - f_i^{\min}). \end{aligned} \tag{3.4}$$

Thus, the constraints change along the optimization process as the current nondominated front changes. For each possible combination of choosing one interval $[l_i^t, u_i^t]$ for each of the objectives $f_{i,i\neq h}$, a separate selection is performed with respect to f_h. The constraints are hard, i.e., a solution that violates any constraint is not considered. This process is repeated until each objective is chosen once as a selection criterion, in order to avoid any preference between the objectives. In

total $m \cdot k^{m-1}$ local selections are performed.

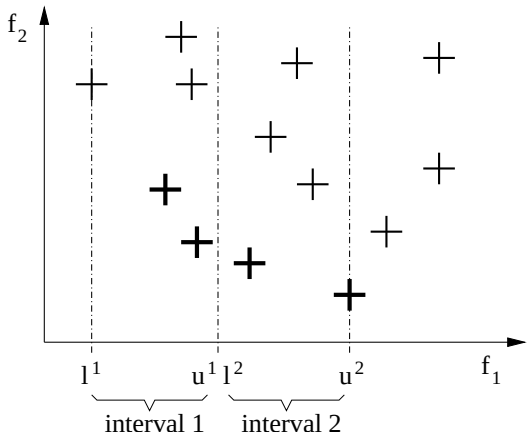

Figure 3.4: Selection by SDM for 2 objectives: The objective space is divided along f_1 into two intervals by specifying a hard lower and upper constraint value l and u for f_1, respectively [dash-dotted lines]. From all solutions [+ symbols] in an interval, always the μ best solutions [bold + symbols] with respect to f_2 are selected. Then the procedure is repeated by dividing the space along f_2 and considering f_1 as selection criterion.

3.3 Recombination and Mutation Operators

We apply simple recombination and mutation operators, as those described for the evolution strategies in Chapter 2. These operators will assist as a basis for performance comparisons of multi-objective evolutionary algorithms in the following chapters. The settings result from various optimization runs.
Each individual in the optimization process is recombined by binary recombination as described in Section 2.1. Each individual is either recombined by uniform recombination, intermediate recombination, or it is a direct copy of a parent. Each of these recombination operators is applied with a probability of one third. For the mutation operator, a simple method with constant normally distributed mutation is

implemented. Each decision variable x_i' is mutated by:

$$x_i'' = x_i' + \begin{cases} N(0,\sigma^2), & \text{, if } p_i < p_{\mathrm{M}},\ p_i \in \mathrm{U}(0,1) \\ 0 & \text{, otherwise} \end{cases}, \tag{3.5}$$

where $N(0,\sigma^2)$ is a normally distributed random number with zero mean and standard deviation σ. p_{M} is the probability of mutating a decision variable and is similar to the mutation probability of bits in the binary string of GA. The mutation probability is set such that on average one to two variables are mutated ($p_{\mathrm{M}} \approx 1.5/n$). p_i is evaluated for every decision variable x_i' separately.

3.4 Test Problems

A wide variety of noise-free test problems for multi-objective optimization can be found in the literature. A number of problems have been summarized in the review articles of van Veldhuizen and Lamont [115] and Deb [23]. From these test problems, 3 different problems for real-valued decision variables are selected. Furthermore, the objective functions of all test problems are unimodal as multimodal function would complicate the comparison of different algorithms.

Test Problem DEB

From Deb [23], a two-objective minimization problem for an arbitrary number of decision variables $x_{1,\ldots,n}$ is chosen:

$$\mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ \frac{1}{x_1} g(\mathbf{x}) \end{bmatrix}, \quad g(\mathbf{x}) = 1 + \sum_{j=2}^{n} x_j^2 \tag{3.6}$$

with $x_1 \in [0.5; 2]$ and $x_{2,\ldots,n} \in [-2.0; 2]$. The Pareto front is given by:

$$x_1 \in [0.5; 2]\,,\ x_{2\ldots n} = 0 \tag{3.7}$$

Test Problems ZDT1,2,6

Zitzler *et al.* propose 6 two-objective test functions with a similar structure to the test problems proposed by Deb [23]. We choose 3 among the 6 functions, where the Pareto front of the first function is convex (ZDT1), the front of the second

function is concave (ZDT2), and density of the Pareto optimal solutions of the third function is non-uniform (ZDT6):

$$\text{ZDT1}: \quad \mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ 1 - \sqrt{\frac{f_1}{g(\mathbf{x})}} \end{bmatrix}, \; g(\mathbf{x}) = 1 + 9\sum_{j=2}^{n} x_j^2 \quad (3.8)$$

$$\text{ZDT2}: \quad \mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} x_1 \\ 1 - \left(\frac{f_1}{g(\mathbf{x})}\right)^2 \end{bmatrix}, \; g(\mathbf{x}) = 1 + 9\sum_{j=2}^{n} x_j^2 \quad (3.9)$$

$$\text{ZDT6}: \quad \mathbf{f} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} 1 - \exp(-4x_1)\sin^6(6\pi x_1) \\ 1 - \left(\frac{f_1}{g(\mathbf{x})}\right)^2 \end{bmatrix},$$

$$g(\mathbf{x}) = 1 + 9\left(\sum_{j=2}^{n} x_j^2\right)^{0.25}, \quad (3.10)$$

with $x_i \in [0;1]$. The Pareto front is obtained by varying the first decision variable while setting all other variables to zero:

$$x_1 \in [0;1], \; x_{2\ldots n} = 0 \quad (3.11)$$

Test Problem FF

As a second test function, the 2-objective problem of Fonseca and Fleming[40] is considered:

$$f_{1,2} = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i \pm \sqrt{\frac{1}{n}}\right)^2\right) \quad (3.12)$$

The Pareto front is given by:

$$x_{1\ldots n} = t, \;\; \text{with} \;\; -\sqrt{\frac{1}{n}} \leq t \leq \sqrt{\frac{1}{n}} \quad (3.13)$$

Test Problem SPH-m

For analyzing the scaling of the optimization algorithms over the number of objectives, a multi-objective generalization of the sphere model [71] is considered,

which is also referred to as *SPH-m* [123], where m denotes the number of objectives:

$$f_i = (1 - x_i)^2 + \sum_{j=1, j\neq i}^{n} x_j^2, \;\; i = 1, \ldots, m, \;\; m \leq n \tag{3.14}$$

with $x_{1,\ldots,n} \in [-b; b]$, $b > 1$ and i is the index of the objective.. A major difference of this test function compared to test functions DEB and FF is that for solutions far from the Pareto front, the objectives are correlated. The analytical Pareto front of test function SPH-m is convex. Thus, it can be computed by performing a weighted-sum aggregation of all objectives into one function. The derivation of this function with respect to all variables x_j leads to n equations. An elimination of the weighting factors from this set of equations leads to the Pareto front. For two objectives the Pareto front is given by:

$$x_1 + x_2 = 1, \; x_{3,\ldots,n} = 0 \wedge x_{\{1,2\}} \geq 0, \tag{3.15}$$

and for 3 objectives by:

$$x_1 + x_2 + x_3 = 1, \; x_{4,\ldots,n} = 0 \wedge x_{\{1,2,3\}} \geq 0. \tag{3.16}$$

In the decision space, the Pareto front of the 2- and 3-objective problem describes either a straight line or an equilateral triangle, respectively. For setting $b = 1000$, the range of each decision variable is about 1000 times larger than the length of the Pareto front. Thus, in the beginning of the optimization, the problem is to locate the Pareto front in the search space.

3.5 Performance Measures

In order to compare different optimization algorithms on the test functions, performance measures are necessary. In multi-objective optimization, the definition of the quality of an optimization run usually considers two different aspects. The quality is dependent on the *convergence* of the optimization run as well as the *distribution* of the final nondominated set along the Pareto front. The convergence is defined as the ability of the algorithm to decrease the distance of the current solutions from the Pareto front. The quality of the distribution depends on the uniformity of the nondominated solutions along the Pareto front. This is different from single objective optimization where convergence is sufficient, since there exists a single global optimum.

In literature several performance measures are proposed. Van Veldhuizen and Lamont [117] present an overview with performance measures in the decision and objective space. Since we will compare different aspects of multi-objective algorithms and also problems that contain noise in their objective functions, several performance measures are necessary. We define performance measures in the decision and objective space.

Performance Measure P

We are interested in a performance measure that shows how well each point of the Pareto front is approximated by the evaluated solutions of the optimization run. This measure addresses both the convergence of the optimization run and the distribution of the nondominated front. It should also directly relate to the mean distance of the approximation to the Pareto front.
Thus, we define the performance measure P as the mean distance of all Pareto solutions to their closest evaluated solution of the optimization run. The distance is measured in decision space. Since this measure cannot be computed for real problems with infinite Pareto solutions, P is approximated by computing the distance just for 10 uniformly distributed points $\mathbf{x}_k^*, k = 1, \ldots, 10$ along the Pareto front. Here we define uniform as equidistant in the design space. To each point $\mathbf{x}_k^*$ the closest evaluated solution $\mathbf{x}_i$ of the optimization run is searched and the distance is computed. The mean of the resulting 10 distances is taken as performance measure P:

$$P = \sqrt{\frac{1}{10}\sum_{k=1}^{10} \min_{i=1,\ldots,N} (||\mathbf{x}_i - \mathbf{x}_k||_2)^2}, \tag{3.17}$$

where N is the number of evaluated solutions. For the considered test functions, the ten uniformly distributed points are:

DEB
$$x^*_{k,1} = \frac{1}{2} + \frac{1}{6}(k-1) \tag{3.18}$$
$$x^*_{k,\{2,\ldots,n\}} = 0$$

FF
$$x^*_{k,\{1,\ldots,n\}} = \frac{1}{9\sqrt{n}}(2k-11) \tag{3.19}$$

SPH-2
$$x^*_{k,1} = \frac{1}{9}k \tag{3.20}$$
$$x^*_{k,2} = 1 - x^*_{k,1}$$
$$x^*_{k,\{3,\ldots,n\}} = 0$$

SPH-3
$$x^*_{k,\{1,2,3\}} \in [0, \frac{1}{3}, \frac{2}{3}, 1], \tag{3.21}$$
$$\text{such that } x^*_{k,1} + x^*_{k,2} + x^*_{k,3} = 1$$
$$x^*_{k,\{4,\ldots,n\}} = 0$$

The performance measure P is to be minimized and small values of P are only obtained if both the optimization converged close to the Pareto front and the solutions are well distributed along the Pareto front. For a single objective problem, P simplifies to the distance of the closest solution to the minimum.

Performance Measure D_{DS} and D_{OS}

For decoupling the convergence and distribution, we define an additional performance measure D, which measures only the convergence of the optimization run. The measure simply computes the distance of each solution in the current population to its closest point on the Pareto front and then takes the root mean square of all distances. This performance measure can be measured in decision space D_{DS} and objective space D_{OS}.

Chapter 4

Optimization Using Fitness Function Models

We present an overview of evolutionary algorithms that use empirical models of the fitness function to accelerate convergence. A fitness function model represents an inexpensive surrogate of the expensive fitness function to optimize.
We distinguish between evolution control and the surrogate approach. In evolution control, models are used to pre-evaluate solutions of an evolutionary algorithm in order to indicate promising solutions. In the surrogate approach, the optimum is searched directly on the model. The resulting optimum is a promising candidate for evaluation on the fitness function. We conclude this chapter with an overview on different models.

4.1 Introduction

The cost of optimizing expensive problems is dominated by the number of fitness function evaluations required to reach an acceptable solution. For evolutionary algorithms, various approaches exist to reduce this cost by exploiting knowledge of the history of evaluated points. This knowledge can for instance be used to adapt the recombination and mutation operators in order to sample offspring in a promising areas. Thus the Covariance Matrix Adaptation (CMA) algorithm [52, 51] uses the path of successful mutations to build up a covariance matrix; the new population is then sampled with this covariance.

Knowledge of past evaluations can also be used to build an empirical model that approximates the fitness function to optimize. The approximation is then used to predict promising new solutions at a smaller evaluation cost than the original problem. The prediction quality generally improves with a growing number of evaluated points in the optimization process. Such models are also referred to as surrogates [113, 15], response surfaces (especially for polynomial approaches), or metamodels [34, 74]. A prerequisite for using them is that the expense of model construction and prediction is lower than evaluating the fitness function; they are

thus used primarily for expensive optimization problems.

4.2 Models in Evolutionary Algorithms

There are two main ways to integrate models into an evolutionary optimization. In the first, a fraction of individuals is evaluated on the fitness function itself, the remainder merely on its model. Jin *et al.* [61] refer to individuals that are evaluated on the fitness function as *controlled*, and call this technique *evolution control*. In the second approach, the optimum of the model is determined, then evaluated on the fitness function. The new evaluation is used to update the model, and the process is repeated with the improved model. This *surrogate approach* evaluates only predicted optima on the fitness function, otherwise using the model as a surrogate.

4.2.1 Evolution Control

In evolution control [61], a *controlled* fraction of individuals are evaluated on the fitness function, the remainder only on the model. Assuming perfect approximation of the fitness function by the model, and computational cost dominated by the fitness function evaluation, this produces a relative reduction in cost equal to the fraction of uncontrolled individuals. Various implementations can be distinguished according to their selection of controlled individuals. Jin *et al.* [61] define two main classes of control rule: in *individual-based* evolution control a fraction of each population is controlled, while in *generation-based* evolution control the entire population is either controlled or uncontrolled.

In individual-based evolution control it is still an open question which of the individuals should be controlled, and the fraction of controlled individuals varies between 10% [34] and 50% [60]. Jin *et al.* [60] state that when randomly controlling individuals, about 50% of the offspring need to be controlled. When pre-evaluating all solutions on the model, then evaluating the best individuals on the fitness function, Giotis *et al.* [48] and Jin *et al.* [60] control about 40% of the population. Emmerich *et al.* [34] show that for simple problems such as, e.g., a symmetric quadratic function, controlling the best 10% of pre-evaluated individuals is sufficient.

For more complex problems such as multimodal or correlated functions, however, their algorithm easily gets stuck. They address this problem by using a merit function as proposed by Torczon and Trosset [113]. Merit functions are a weighted

sum of the model prediction and a negative density measure. The density measure promotes unexplored regions, i.e., areas where no points have yet been evaluated. Thus, merit functions balance the goal of finding promising solutions (exploitation) with improving the model by obtaining information about new regions (exploration), thus decreasing the risk of premature convergence [113, 81].
Beltagy and Keane [33] employ Gaussian process models [73], which provide an uncertainty measure (in terms of a standard deviation) along with the predicted fitness function value. They only control individuals whose predicted standard deviation exceeds a certain limit, assuming the model prediction to be accurate otherwise. The limit is decreased linearly over the course of the evolution.
In generation-based evolution control, the entire population is evaluated on either the model or the fitness function; this allows for better parallelization. Again, various rules have been developed to determine which generations to control. Ratle [89] controls the first generation of his GA; subsequent generations are evaluated only on the model until the model predictions do not improve for a given number of generations. The next generation is again controlled. Compared to the original GA, this approach accelerated convergence for uni- and multimodal functions. Jin *et al.* [59] control generations until the error between model prediction and fitness function drops below a certain threshold. The population then remains uncontrolled for a given number of generations before control resumes.
For both control methods several questions remain open. First, there is disagreement about which and how many individuals of a population need to be controlled. Then it is not clear whether all or just a fraction of evaluated points should be used for model construction, so as to perform a global [33], respectively local (i.e., recent) approximation [34]. The model's complexity must also be appropriate for the amount of data used in order to avoid overfitting the data.
Finally, inexact model predictions may mislead the selection operator to propagate inferior individuals. This may be especially detrimental to optimization algorithms that are themselves adaptive. We hypothesize:

> The more information from the population is exploited by the evolutionary algorithm (e.g., to adapt the mutation distribution), the higher the fraction of controlled individuals has to be in order to provide sufficient information for the adaptation process.

In other words, evolution control is based on the assumption that the evolutionary algorithm needs very little information, which can be provided by evaluating (controlling) just a fraction of the population. However, this argumentation holds only for those inefficient algorithms that indeed use little information — for "smarter"

algorithms such as CMA [52, 51] that pull much more information out of a population, we expect that virtually all individuals must be controlled.
The success of evolution control is thus highly dependent on the fraction of controlled individuals, which is difficult to determine as it depends on both the fitness function complexity and the optimization algorithm. It is always a compromise between avoiding the computational cost of fitness function evaluation and the danger of a poor model misleading the optimization [59].

4.2.2 Surrogate Approach

In the surrogate approach, a fitness function model is constructed for an initial training set of evaluated points. An optimization algorithm then searches for the optimum of the model's fitness prediction. The predicted optimum constitutes an ideal candidate for an improved solution to the problem, and is therefore evaluated on the fitness function. The result of the evaluation is added to the model's training data, facilitating an improved approximation of the fitness function by the model. The procedure then iterates by searching for the optimum of the improved model. This surrogate approach has been discussed by Torczon *et al.* [113]; similar methods can be found in [47, 89, 44, 81].
Similar to evolution control, the surrogate approach is in danger of getting stuck in a local minimum unless points in unexplored regions of the search space are added to the model's training set. Thus, merit functions (or similar techniques) are also in use here [113, 81]. Open issues again include the question of whether global or local modeling should be preferred. For local models, the search for the optimum must be limited to the region that is well-approximated by the model. The potential reduction in computational cost is higher for the surrogate approach than for the evolution control, especially once enough data is available to allow for construction of a model that is accurate near the true optimum. Since the surrogate approach proceeds sequentially from one predicted optimum to the next, however, it is more difficult to parallelize.

4.2.3 Empirical Models

A wide variety of empirical models are used in the literature as fitness function models for an optimization procedure. The most prominent among them are polynomial models [108], artificial neural networks [60], radial basis function networks [45, 81], and Gaussian processes [89, 61, 33, 34].

Polynomial Models

Polynomial fitness function models — also referred to as *response surfaces* — can easily be fitted to data with a least squares approach (see, e.g., [108]). The order of the polynomial is important: quadratic or cubic polynomials are mainly used, with quadratic polynomials best suited for continuous, unimodal problems. Since higher-order polynomial fits tend to oscillate too much, multimodal shapes are better approximated by splines, i.e., piecewise polynomial fits with continuity constraints at the boundaries.

Artificial Neural Networks

Artificial Neural Networks (ANNs) [53] consist of a large number of highly interconnected processing units, each aggregating information from a large number of connected peers. Given a sufficient number of units, an ANN can approximate any function. An ANN can be trained by adapting the weights that specify the amplification of signals between connected units. Among the most popular training algorithms for ANNs is *error back-propagation* [93].
A major disadvantage of ANNs is that the resulting weights of the trained network are difficult to interpret. Another problem is the difficulty of finding an appropriate network topology and size. In order to avoid both underfitting (i.e., bad approximation of the training data) resulting from too small networks, and overfitting (i.e., bad generalization) due to too large networks, often several networks of different size and/or topology must be trained [10], and their performance compared on a separate set of test data to estimate their generalization properties.

Radial Basis Functions Networks

The output of a Radial Basis Function Network (RBFN) [53] is a weighted sum of radial basis functions, each characterized by its center $\mu \in \mathbb{R}^n$ in the design space, and a function which declines with increasing distance from the center. With Gaussian radial basis functions, for instance, the output $\hat{y}$ of an RBFN is given by

$$\hat{y}(\mathbf{x}) = \sum_{i=1}^{N_R} w_i \exp\left(-\frac{\left\|\mu_i - \mathbf{x}\right\|^2}{r^2}\right), \tag{4.1}$$

where N_R is the number of radial basis functions, w_i their weights, μ_i their centers, and r their rate of decay. Often each given data point is used as the center of a

radial basis function. The weights are obtained by a least squares fit similar to the polynomial approach [53, 81]. A major difficulty in RBFNs is to set the decay parameter r, which has a large impact on the approximation. Only limited theory and some empirical formulæ exist to address this problem. In [81], the decay parameter is set as:

$$r = d_{\max} \left(n \, N_R\right)^{-1/n} , \tag{4.2}$$

where $d_{\max}$ is the maximal distance between the data.

Gaussian Processes

Gaussian processes (GPs) [73] specify a probabilistic model over a given set of data points, constructed such that the likelihood of the function value given the decision variable values is maximized for all data points. This model can then be extended to predict the mean and standard deviation of the function value at new data points. GPs have a small number of *hyperparameters* which can be set by the user, or optimized via a maximum likelihood approach.
Like ANNs, GPs can approximate any function. Their main advantage over ANNs is their simplicity: no network size or topology must be chosen. In contrast to the weights of an ANN, the hyperparameters of a GP have intrinsic meaning — specifying, e.g., typical length scales — and can therefore be set using prior knowledge of the problem, such as noise levels, location of discontinuities, *etc*. One drawback of GPs is their computational cost: for N data points, it takes $\mathcal{O}(N^3)$ steps to construct the GP, $\mathcal{O}(N)$ to predict the mean function value at a new point, and $\mathcal{O}(N^2)$ to predict the standard deviation.

Part II

Advances in Evolutionary Algorithms

Chapter 5

Convergence Limits of Multi-objective Selection Operators

Evolutionary Algorithms are a standard tool for multi-objective optimization that are able to approximate the Pareto front by a set of nondominated solutions in a single optimization run. However, the convergence of different algorithms can be limited. Here we quantify convergence as the decrease in distance in the objective space of the current nondominated solutions to the Pareto front.
Two necessary prerequisites for convergence are defined. First, the selection operator of the algorithm must be able to operate efficiently in any distance of the Pareto front. Second, the mutation strength must adapt to the decreasing distance to the Pareto front in order to result in an efficient algorithm. The mutation strength is the mean deviation of the offspring from the parents.
We show that the convergence is limited for various selection operators based on the dominance criterion and we compute an estimate for their convergence limit. The algorithm uses self-adaptive mutation and different algorithms are analyzed on a 2- and 3-objective test function.

5.1 Introduction

In Pareto optimization, recent research has focused on multi-objective selection operators and in particular fitness assignment techniques. Various selection operators are compared in the literature [125, 88, 124] and according to Van Veldhuizen and Lamont [117], the dominance criterion in combination with niching techniques is one of the most efficient techniques for the fitness assignment. This group of algorithms is referred to by Horn [55] as “Cooperative Population Searches with Dominance Criterion” and two prominent representatives are SPEA [125] and NSGA-II [26].
While these algorithms perform well in a number of test problems, we observe a

stagnation in the convergence of these algorithms at a certain distance from the Pareto front. The distance is only dependent on the selection operator and varies not so much for different mutation or recombination operators. In this chapter, we estimate this stagnation distance by deriving an analytical solution for a simplified Pareto front. The distance can increase nonlinearly with the number of objectives as analyzed by Khare *et al.* [63]. This raises the question, which selection operators are able to converge to the Pareto front and in addition, which operators converge efficiently?

Two alternatives to the dominance criterion are the Constraint Method-based Evolutionary Algorithm (CMEA) [88] and Subdivision Method (SDM) [16]. These algorithms perform selection by optimizing one objective, while the other objectives are treated as constraints. They are able to converge to the Pareto front for certain test cases.

In conjunction with the selection operator, the mutation and recombination operators are important for an efficient convergence and should adapt to the decreasing distance to the Pareto front while converging. In recent years, some efforts have been made in order to apply self-adaptation in multi-objective optimization. Kursawe [68] and Laumanns *et al.* [71] developed two implementations of self-adaptation [101]. Kursawe's algorithm performs selection based on a randomly chosen objective. In his work each individual contains a separate vector of design variables and step sizes for each objective (poliploid individuals). Laumanns *et al.* assign a single step size to each individual, which yields isotropic mutation distributions. Sbalzarini et al. [97] use mutative step size adaptation with a constant adaptation factor.

In the following, different multi-objective algorithms, as proposed by Horn [55], and described in Section 3.2 are analyzed theoretically in terms of their ability to converge to the Pareto front. Then, the implementation of self-adaptive mutation is discussed. Finally, the performance of the different algorithms is analyzed on a 2- and 3-objective test function. In the performance comparison, the number of resulting nondominated solutions of the different algorithms is allowed to be small. This is consistent with algorithms for real-world applications as analyzing these solutions is often an expensive process. The quality of the resulting solutions is more important and thus the focus is on the ability of these algorithms to converge with arbitrary precision to the Pareto front.

5.2 Analysis of Selection Operators

We analyze different selection operators with respect to their general ability to converge to the Pareto front. This requires that the selection operator selects efficiently independently of the distance from the Pareto front. As a necessary but not sufficient criterion for convergence an efficient selection operator must select, on average, solutions that are closer to the Pareto front than the average solution in the population. Some of the algorithms do not fulfill this criterion.

5.2.1 Independent Sampling

Independent Sampling generates an approximation of the Pareto front by performing several independent optimization runs with different aggregation of the objectives. We analyze the Constraint Method-based Evolutionary Algorithm (CMEA) [88] (see Subsection 3.2.1) as a representative of independent sampling. In CMEA, one objective f_h is selected for optimization, while all other objectives $f_{i,i\neq h}$ are treated as constraints. The constraint method allows full ordering among all feasible solutions (if we assume that each objective vector $\mathbf{f}$ occurs only once): In a mutual comparison of two solutions one is always clearly better. If neither of the two solution violates any constraint, the solution with the lower objective value for f_h is preferred. If one or both solutions violate at least one constraint, the solution with the smaller sum of all constraint violations is preferred. The best solution is a single point located at the intersection of all constraints with the Pareto front. Thus, independent sampling is able to operate at any distance from the Pareto front by full ordering of the solutions.

5.2.2 Cooperative Population Searches with Dominance Criterion

Cooperate population searches often use the dominance criterion in conjunction with a niching technique. While the dominance criterion promotes the convergence towards the Pareto front, the niching technique promotes diversity in the population and thus a uniform approximation of the Pareto front. A prerequisite for convergence is that at least one individual dominates another individual in the population. Otherwise, all individuals are indifferent and it is not clear, which individual is closer to the Pareto front. For SPEA1/2 or NSGA-I/II the dominating individual is an archive solution or a nondominated individual in the population, respectively. We discuss this convergence prerequisite using a 2-objective and 3-objective example.

In Fig. 5.1, a simple 2-objective minimization problem is given with the Pareto front being a straight line between $\mathbf{f} = \{1, 0\}$ and $\mathbf{f} = \{0, 1\}$. SPEA1/2 or NSGA-I/II operate with a limited number of parents μ. SPEA1/2 contains in addition an archive, which is set to the same size as the parent popuation. Without loss of generality we consider μ to be small and equal to $\mu = 5$. For the ideal case, that all archive solutions are uniformly distributed along the Pareto front, the nondominated area is at minimum [35]. In the figure, the bounds of the nondominated area are given by the Pareto front and the dashed lines. This area contains indifferent solutions, i.e., solutions of the same quality as the considered archive solutions, although these solutions may have a certain distance to the Pareto front.
For the limited population size μ, a selection operator may select a new solution within this area over one of the archive solutions. The resulting archive is then clearly worse than the previous one. Here, the selection operator based on dominance fails.
For the considered ideal case in 2D, the maximal distance of a nondominated solution to its closest point on the Pareto front can be calculated as:

$$D = \frac{1}{\sqrt{2}(\mu - 1)}. \tag{5.1}$$

Thus, the minimal number of archive solutions $\mu_{\min}$ in order to dominate a solution in a distance D from the Pareto front scales with:

$$\mu_{\min} \sim \frac{1}{D} \tag{5.2}$$

In Fig. 5.2 (left), a 3-objective minimization problem is given with the Pareto front being an equilateral triangle in unit space. First, we consider an archive size of $\mu = 3$, with each archive solution in one corner of the Pareto front. The resulting nondominated volume is a tetrahedron with the 3 archive solutions and the point $\mathbf{f} = \{1, 1, 1\}$ as vertices. The maximal distance of a nondominated solution to its closest point on the Pareto front is equal to the height of the tetrahedron which is equal to $D = \frac{2}{3}\sqrt{3}$. In order to reduce this distance, we use resembling tetrahedra of smaller size. For tetrahedra of half of the size of the initial tetrahedron 6 nondominated solutions are needed and the nondominated volume is given by the 4 tetrahedra as shown in Fig. 5.2 (right). The height of each tetrahedron is then equal to $D = \frac{1}{3}\sqrt{3}$. For reducing the nondominated tetrahedra by a factor of k, $\mu = \sum_{i=1}^{k+1} i = \frac{1}{2}(k+1)(k+2)$ solutions are needed. Thus the maximal distance

D can be computed as:

$$D = \frac{2}{\sqrt{3}\left(\sqrt{2\mu + \frac{1}{4}} - \frac{3}{2}\right)}. \tag{5.3}$$

The minimal number of archive solutions $\mu_{\min}$, necessary to dominate a solution in distance D from the Pareto front, scales for 3 objective with:

$$\mu_{\min} \sim \frac{1}{D^2} \tag{5.4}$$

For two simple Pareto fronts, we have shown in Eqns. 5.1 and 5.4 that the convergence of selection operators based on dominance is limited and scales with the population or archive size. Similar results can be expected for differently shaped Pareto fronts. For algorithms like SPEA1/2 [124] and NSGA-I/-II [26] the scaling is experimentally validated in Subsection 5.3.2.

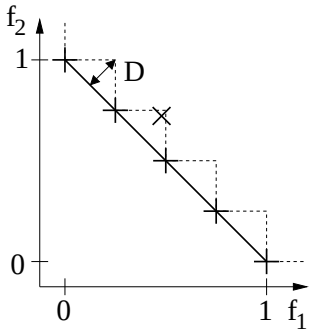

Figure 5.1: Disadvantage of the dominance as selection criterion: for a limited number of archive solutions [+ symbols], which may be on the Pareto front [solid line], the objective space cannot be completely dominated, e.g., the solution [x symbol] is not dominated, even though it is still far from the Pareto front.

5.2.3 Cooperative Population Searches without Dominance Criterion

This class of optimization approaches converges towards the Pareto front in a single optimization run, but does not use the dominance criterion in the selection

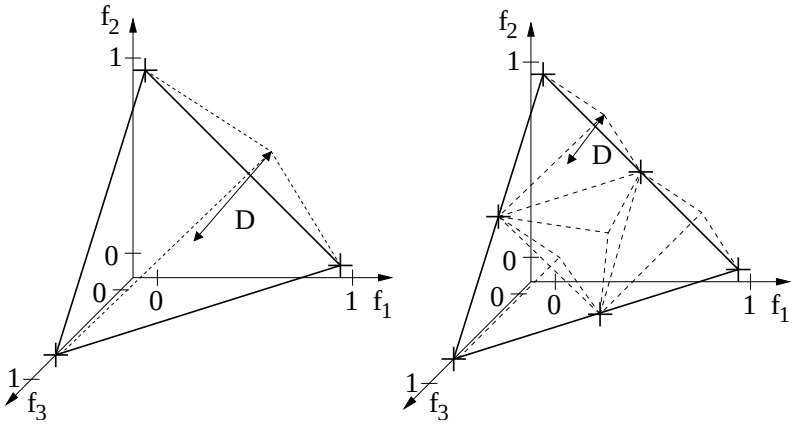

Figure 5.2: Disadvantage of the dominance as selection criterion: for a limited number of archive solutions [+ symbols], which may be on the Pareto front [solid line], the objective space cannot be completely dominated. The size of the nondominated tetrahedra [dashed lines] is dependent on the number of archive solutions.

operator. One representative is the Subdivision Method (SDM) [16], as introduced in Subsection 3.2.3. Similar to the CMEA, SDM performs selection based on a constraint approach. However, several selections with different constraint values are done in one population. The best solution for each selection is a single point located at the intersection of the constraints with the Pareto front. Thus, similar to CMEA, SDM is also able to operate at any distance from the Pareto front by full ordering of the solutions.

5.3 Experimental Analysis

We experimentally analyze the performance of the 3 different classes of multi-objective algorithms and theoretically discussed in the previous section. For simplicity, just one representative of each class is considered in order to focus on the general convergence properties and not on the convergence speed.

5.3.1 Combining Self-adaptive Mutation with Multi-objective Selection Operators

In Section 5.2, an efficient selection operator was introduced as a prerequisite for converging towards the Pareto front. A further prerequisite is also an efficient recombination and mutation operator. While converging, the mutation strength has to be adapted in order to take into account the decreasing distance from the Pareto front.
We discuss the integration of self-adaptive mutation as described in Section 2.3 into the various selection operators. The integration is not obvious since self-adaptive mutation was introduced for single objective optimization [101] with the goal to converge to a single (global) optimum. For Pareto optimization, the question arises how self-adaptation will perform when converging to a Pareto front instead to a single point.

Independent Sampling

In independent sampling, an approximation of the Pareto front is obtained by performing several optimization runs with aggregating differently all objectives into a single figure of merit. Independent sampling is an ideal candidate for applying self-adaptative mutation as the multi-objective problem is transformed to a set of single objective problems. Thus, self-adaptive mutation of single objective optimization is directly applicable.
Some knowledge of previous runs can be exploited by using the best solution(s) obtained so far as initial solution(s) for the next optimization run [88]. The CMEA of Subsection 3.2.1 is used as representative of independent sampling for the experimental analysis.

Cooperative Population Searches with Dominance Criterion

Cooperative population searches that base on the dominance criterion converge in a single run towards the Pareto front. Applying self-adaptation in form of correlated mutation has been shown to be difficult for this class of selection operators for various reasons:

- The usually elitistic selection scheme is not beneficial for self-adaptation, since it might inhibit the adaptation of the strategy parameters by preserving parents with miss-adapted strategy parameters over several generations. These parents are unlikely to produce superior offspring.

- For self-adaptive mutation, recombination of the individual step sizes is considered beneficial as it levels out the noise in the adaptation. This class of selection operators, however, promotes a uniform distribution of solutions along the Pareto front. It is an open question, which of these solutions should be recombined. The different solutions converge to different parts of the Pareto front and might have different efficient strategy parameters for the mutation.

- Adapting the mutation distribution is only necessary, if the initial mutation distribution does not fit to the function topology or if the distance of the intial population to the optimum (Pareto front) has to be decreased over several orders of magnitude. Adaptation for this class of selection operators is questionable since the convergence is limited as described in the previous Section 5.2.

- In correlated mutation, lines of equal probability of the mutation distribution are ellipses that can be arbitrarily rotated. The analysis of Sadnik and Glas [94] show that for CMA-ES, the ellipses orient such that the main principal axis are in the decision space parallel to the Pareto front, while towards the Pareto front the mutation strength decays. This promotes generating further nondominated solutions. However, it does not promote the convergence to the Pareto front. Thus, using correlated mutation becomes inefficient.

Nevertheless, we combine self-adaptation with this class of selection operators in order to analyze if any adaptation occurs. The mutation is performed after intermediately recombining two parents.

Cooperative Population Searches without Dominance Criterion

Self-adaptation can be applied to SDM by the following procedure: The selection process can be considered as performing several local selections in a “subdivided” objective space with the constraints being responsible for the subdivision. Each local selection selects a set of individuals according to the given constraints. Each selected set may have a different mean distance to the Pareto front, resulting in different efficient strategy parameters. Thus, recombination as described in Section 2.3 is always performed within each selected set. Finally, self-adaptive mutation is applied to the recombined individuals. Similar to Independent Sampling, the SDM converges towards a fixed set of points on the Pareto front. Thus, self-adaptive mutation is applicable to SDM.

5.3.2 Experimental Results

The different selection operators are analysed on the test function SPH-m for $m = 2$ and $m = 3$ objectives and $n = 10$ decision variables. The initial population of each optimization run is sampled from a uniform distribution within $\mathbf{x}_i = [-10, 10]$. The main interest is on the convergence of the optimization process to the Pareto front. This is is addressed by plotting the mean distance D_{OS} from Section 3.5 of the parent population (SDM, CMEA) and the archive solutions (SPEA2) over the number of function evaluations. The distribution of the solutions is of minor importance and is analyzed by plotting the final parent population and archive of the optimization run in the objective space. The experimental results are given separately for the 3 considered optimization approaches. For each approach and test function, 10^6 function evaluations are performed.

Independent Sampling

The CMEA is analyzed as a representative of Independent Sampling. For 2- and 3-objective test functions, the Pareto front is approximated by performing independent optimization runs with a $(15, 3, 100)$ strategy, implementing correlated mutation. All initial step sizes are set to 1 and $17 \cdot 10^3$ solutions are evaluated for each run, leading in total to about 10^6 evaluated solutions for all six optimization runs.

First, the 2-objective case is considered. The optimization starts with 2 single objective optimization runs for f_1 and f_2 from random initial solutions. Then, 4 constraint optimization runs are performed, where f_2 is optimized and f_1 is treated as constraint. The best solution of the two single objective optimization runs have f_1 values of approximately 0 and 2, respectively. Thus, the constraints on f_1 for the 4 remaining runs are set uniformly between the two values as 0.4, 0.8, 1.2, and 1.6.

For the 3-objective case, 3 single objective optimization runs are started for f_1, f_2 and f_3. Then 3 constraint optimization runs are started, where f_3 is optimized and f_1 and f_2 are constrained. The constraints are set to $\{f_1, f_2\} = \{1, 2\}, \{2, 1\}, \{1, 1\}$, respectively. For the constraint optimization runs, some knowledge of previous runs is exploited by using the best solutions obtained so far as initial solutions for the next run [88]. We consider a hard constraint: If solutions violate a constraint, they are not considered for selection, given a sufficient number of solutions do not violate any constraint. Otherwise, the solutions that do not violate the constraint are selected first and then solutions are added dependent

on their constraint violation. This constraint ensures a convergence to a point on the Pareto front. A soft penalty may converge to a point in the neighborhood of the Pareto front.

Fig. 5.3 shows the performance measure for all 6 optimization runs, which represents the mean distance of the parent population at each generation to the Pareto front. The figure shows large differences in the convergence between the different runs. While the single objective optimizations converge linearly, the convergence of the constraint optimizations slows down at a distance of about $D_{\mathrm{OS}} = 10^{-3}$ to the Pareto front. Fig. 5.4 addresses this aspect for the 2-objective problem and shows the contour lines for f_1 and f_2 and the Pareto front in the (x_1,x_2) space. Each contour line of f_1 represents a different constraint setting. The optimum for each constraint optimization is located in the intersection of a f_1 contour line with the Pareto front. In the vicinity of the optimum, the topology is badly scaled and oriented such that it is difficult to optimize with the given self-adaptation scheme (compare Hansen and Ostermeier [52], Fig. 2).

For the 2-objective problem, Fig. 5.9 (left) shows the final parent population of each optimization run and the Pareto front. While the solutions are equally spaced in f_1 axis, the distribution is nonuniform along the f_2 axis, with the highest density on areas that are nearly parallel to the f_1 axis. This results from treating f_1 and f_2 differently as constraint and objective, respectively.

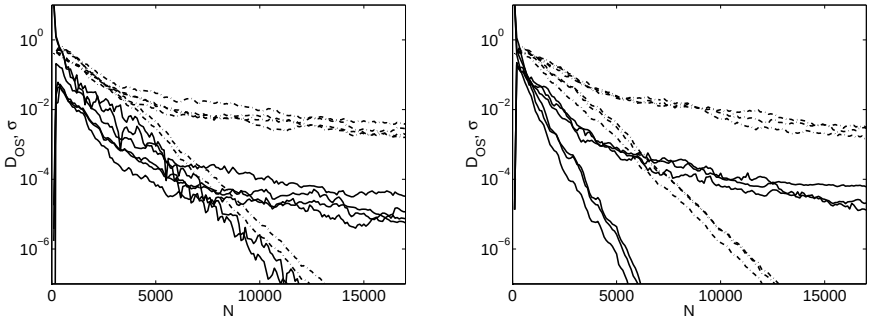

Figure 5.3: Convergence of CMEA with self-adaptive mutation for the 2-objective (left) and 3-objective (right) sphere problem SPH-m with $n = 10$ decision variables. The mean distance D_{OS} of the current parent population to the Pareto front is shown [solid line] as well as the mean standard deviation σ of the mutation distribution [dash-dotted line].

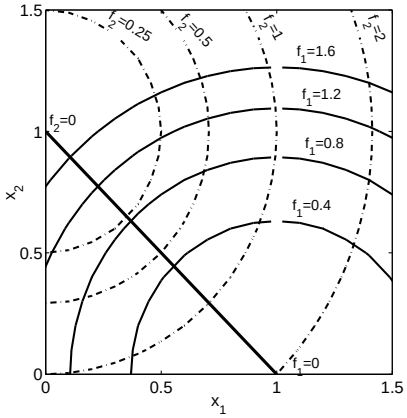

Figure 5.4: Convergence difficulty in the CMEA for optimizing f_2 while setting f_1 as constraint. The optimum for a specific constraint value for f_1 is located at the intersection of the Pareto front [bold solid line], with the contour line of f_1 [solid line]. In the vicinity of the optimum, the topology is badly scaled and equals a long narrow valley with the main expansion parallel to the line given by the equation $x_1 = x_2$.

Cooperative Population Searches with Dominance Criterion

For SPEA, different archive sizes of $\mu = 10$, 30, 100, and 300 are analyzed for the 2- and 3-objective sphere problem SPH-m. The number of parents and offspring is set equal to the archive size, and in total 10^6 solutions are evaluated. In order to analyze just the effect of the archive, a normally distributed mutation with zero mean and constant standard deviation between 0.0005 and 0.002 is used with a mutation probability of 15% per variable. Discrete and intermediate recombination of 2 parents is considered with 33% probability each. These settings are taken from Section 3.3 and lead to the best results in terms of the final convergence of the algorithm to the Pareto front. The actual convergence speed of SPEA is of minor interest.

Fig. 5.5 shows the convergence measure D_{OS}, which represents the mean distance of the archive solutions to the Pareto front, measured in objective space. It can clearly be seen that the convergence stagnates at a certain distance D_{OS} from the Pareto front. This distance decreases with increasing archive size. Comparing the 2- and 3-objective optimization results, D_{OS} is significantly larger for the 3-

objective problem than for the 2-objective problem. In addition, increasing the archive size leads to a smaller relative decrease in D_{OS} for the 3-objective problem.
This underlines the previously derived estimated distance in Eqns. 5.1 and 5.4: In order to obtain the same mean distance D_{OS} from the Pareto front, the necessary number of archive solutions is larger for 3 objectives than for 2 objectives. From Fig. 5.5, the mean of D_{OS} is computed for the stagnation period between $N = 5{\cdot}10^5$ and $N = 10^6$ and compared to the estimated distance D from Eqns. 5.1 and 5.4. Since the Pareto front of the test functions is in each objective direction twice as large in size as the unit Pareto front from the estimation, the estimated value D is multiplied by a factor of 2. The result is given in Fig. 5.7. The experimental results agree with the two equations. The slope of D is equal to D_{OS} for the 2 and 3 objective problem. However, there is a constant offset between the two curves. This offset may have different reasons. The estimated distance was computed for some ideal assumptions (a straight Pareto front, uniformly distributed archive solutions, etc.). In addition, the clustering procedure that limits the archive size of SPEA may have some effect. This effect is analyzed by replacing the clustering procedure by randomly deleting solutions from the archive. The results are also added to Fig. 5.7. While for the 2-objective problem, the resulting curve differs from the clustering, the difference is minor for the 3-objective problem.
The distribution of the final archive solutions is shown for the 2-objective problem with an archive size of 300 in Fig. 5.9. The archive approximates the Pareto front uniformly by a large number of nondominated solutions.
Using self-adaptive mutation leads to worse results than for the constant step size. The results are given in Fig. 5.5 and show the mean distance D_{OS} as well as the mean step size of the archive population. The step size decreases as the archive converges towards the Pareto front and stagnates as the distance of the archive stagnates. For the case of the 2-objective problems with large populations, the performance is especially worse than for the constant step size. Then, the adapation process is slower due to the large population size and the consequently lower number of generations in the optimization run. The results show that self-adpation reduces the step size, however, at a very moderate speed.

Cooperative Population Searches without Dominance Criterion

The SDM is now analyzed on the multi-objective sphere. For each local selection a $(8, 3, 56)$ strategy is used. The convergence measure D_{OS} for the SDM is set to the mean distance of all selected parents to the Pareto front, measured in objective

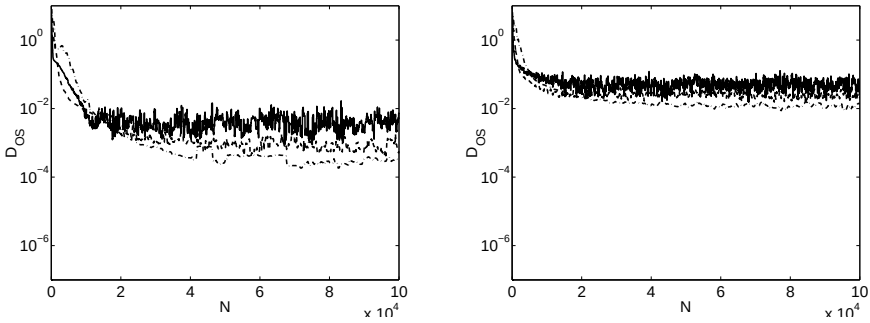

Figure 5.5: Convergence of SPEA with an optimized mutation step size for the 2-objective (left) and 3-objective (right) sphere problem. The mean distance D_{OS} of the current archive to the Pareto front is given for archive sizes of 30 [solid line], 100 [dashed line], and 300 [dash-dotted line].

space. For the 2-objective problem, 3 intervals are chosen along each objective axis, leading in total to 6 local selections and a maximal number of 48 parents and 336 offspring. For the 3-objective problem, each objective axis is divided into 2 intervals, leading to a total number of 12 local selections and a maximal number of 96 parents and 772 offspring. The bounds of the intervals are obtained from the current nondominated front of the optimization run and thus no user specification is needed. Similar to CMEA, a global intermediate recombination of the parents from each local selection is performed for the variables and step sizes. No recombination is applied to the rotation angles.

In total 10^6 solutions are evaluated and the convergence measure is plotted in Fig. 5.8. In addition, the mean step size of the mutation distribution is given. The convergence speed decreases at a value of about 10^{-3} for the same reason as for the CMEA: The convergence becomes difficult due to the constraint optimization. Here, the convergence speed for the 3-objective problem is slower than for the 2-objective problem. This result is mainly due to the double number of local selections. The parents of the final population are plotted in Fig. 5.9 and are uniformly distributed along the Pareto front.

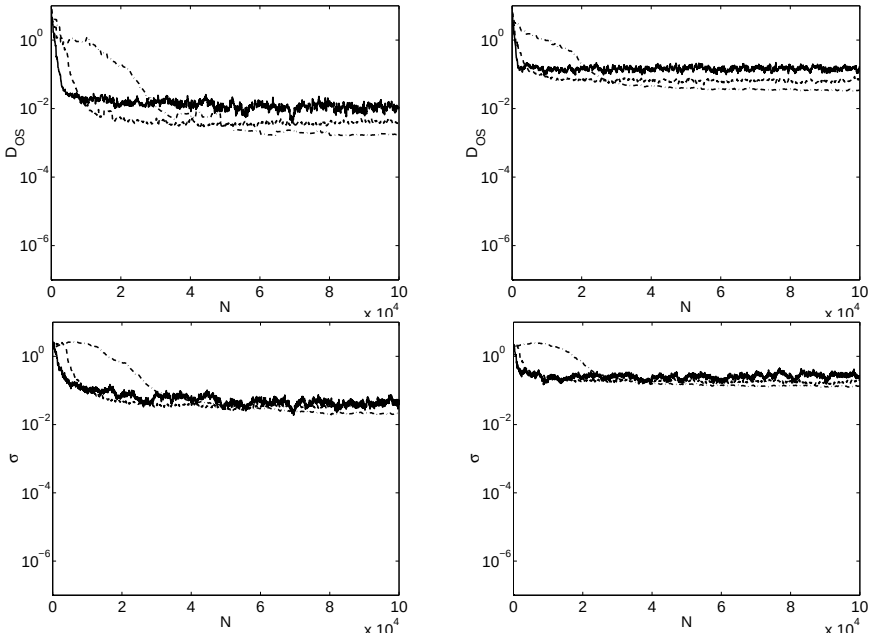

Figure 5.6: Convergence of SPEA with self-adaptive mutation for the 2-objective (left) and 3-objective (right) sphere problem. The mean distance D_{OS} of the current archive to the Pareto front is given (upper two figures) as well as the mean standard deviation σ of the mutation (lower two figures) for archive sizes of 30 [solid line], 100 [dashed line] and 300 [dash-dotted line].

5.4 Conclusions

Evolutionary Algorithms for multi-objective optimization should implement efficient techniques in order to improve convergence towards the Pareto front, while approximating it uniformly. We studied three different classes of multi-objective algorithms by comparing one representative of each class. The question was which of these algorithms is able to converge to the Pareto front with an arbitrary precision.

We found that cooperative population searches like SPEA that use the dominance criterion in the fitness assignment cannot approximate the Pareto front with arbitrary precision. For a 2-objective optimization problem, the necessary number

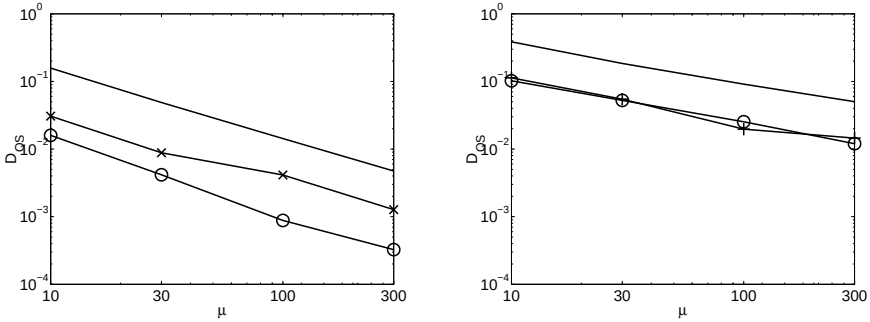

Figure 5.7: Comparison between theoretical and experimental results for SPEA on the 2-objective (left) and 3-objective (right) sphere problem SPH-m with $n = 10$ decision variables. For archive sizes of $\mu = 30$, 100, and 300 the mean value of D_{OS} is plotted for limiting the archive size as described in SPEA2 [o] or by randomly deleting solutions [x]. Furthermore, the theoretical estimate D [no symbol] from Eqns. 5.1 and 5.4 is added.

of archive scales inversely with the final distance of solutions to the Pareto front. For 3 objectives, convergence becomes even more difficult, since the necessary number of archive solutions scales inversely with the distance squared. This result holds for other algorithms using the dominance criterion and a limited population (e.g., NSGA-II, SPEA2).

The algorithms CMEA and SDM do not use dominance. In these algorithms, one objective is selected for optimization, while the other objectives are treated as constraints. Both algorithms converge to a fixed number of discrete points on the Pareto front and can reach theoretically arbitrary precision. In the experimental comparison, the convergence velocity decreased over the optimization run, since the imposed constraints modify the optimization problem such that the problem becomes similar to a correlated and misscaled function. CMEA finds one optimal point in each optimization run and thus needs to be run for several times in order to find an approximation of the Pareto front. In contrast, SDM finds an approximate Pareto front in a single optimization run. It was shown that self-adaptation can easily be applied to CMEA and SDM and that both algorithms converge successfully to the Pareto front. As an additional result of this study, the comparison shows that they clearly outperform SPEA in terms of the final distance to the Pareto front as show in Table 5.1. Comparing CMEA and SDM in terms of convergence speed, CMEA is faster on the considered test function, although it is

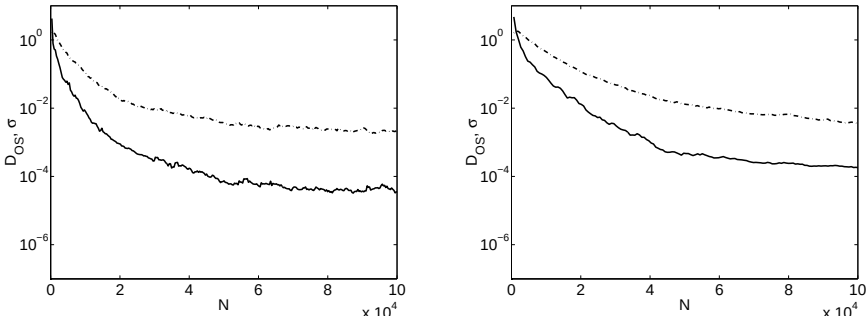

Figure 5.8: Convergence of SDM with self-adaptive mutation for the 2-objective (left) and 3-objective (right) sphere problem SPH-m with $n = 10$ decision variables. The mean distance D_{OS} of the current parent population to the Pareto front is given [solid line] as well as the mean standard deviation σ of the mutation [dash-dotted line].

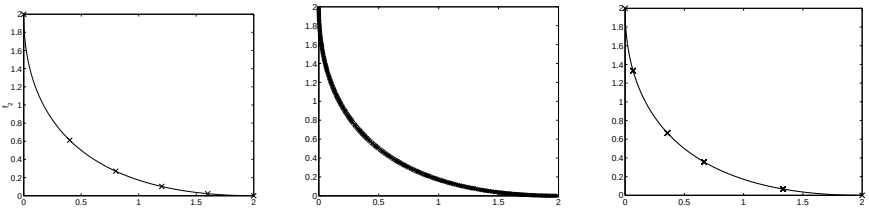

Figure 5.9: Location of the best solution from each independent run of CMEA (left), the final archive of SPEA (middle) and final parent population of SDM (right) is given [x] together with the Pareto front [bold line].

not known if this generalizes to other functions and to other adaptation schemes. For CMEA, one has to decide before optimization, which objective is optimized subject to the other objectives, which are then treated as constraints. This is in contrast to SDM that gives no a-priori preference to any of the objectives.
For the considered test functions, SPEA results in a larger number of nondominated solutions than CMEA and SDM. However, a small number of converged solutions is often sufficient, especially since in real-world applications the analysis of these solutions is often expensive. Thus, algorithms like CMEA and SDM are interesting alternatives to the well established algorithms based on the dominance criterion.

Table 5.1: Comparision of the final mean distance D_{OS} of CMEA, SPEA, and SDM to the Pareto front after 10^6 evaluated solutions.

Algorithm	D_{OS}	
	2 obj.	3 obj.
CMEA	8.01e-06	1.55e-05
SDM	3.25e-05	1.81e-04
SPEA-2, archive limited by k-nearest neighbor cluster algorithm		
$\mu = 10$	1.60e-02	1.02e-01
$\mu = 30$	4.16e-03	5.24e-02
$\mu = 100$	8.80e-04	2.53e-02
$\mu = 300$	3.25e-04	1.20e-02
SPEA-2, archive limited by random deletion		
$\mu = 10$	3.06e-02	1.13e-01
$\mu = 30$	8.82e-03	5.47e-02
$\mu = 100$	4.13e-03	1.97e-02
$\mu = 300$	1.27e-03	1.45e-02

Some difficulty in converging has been found for CMEA and SDM. CMEA and SDM transfer the multi-objective sphere problem into a constraint optimization problem that may impose additional optimization difficulties. While the optimization run of a single objective of the sphere problem converges linearly (see Fig. 5.3a, solid lines), the convergence speed of the constraint optimization problem decreases over the number of function evaluations (see Fig. 5.3a, dash-dotted lines). In general the question arises if objectives could be aggregated by a different method, leading to a linear convergence also in the constraint case. In addition, Hansen and Ostermeier [52] stated that self-adaptation does not efficiently adapt to arbitrary correlation distributions and thus methods like the Evolution Strategy with Covariance Matrix Adaptation (CMA-ES) [52] might perform better on the constraint problem.

Chapter 6

Multi-objective Evolutionary Algorithm for Noisy Objective Functions

---

This chapter introduces a multi-objective evolutionary algorithm capable of handling noisy problems with a particular emphasis on robustness against unexpected measurements (outliers). The algorithm extends the Strength Pareto Evolutionary Algorithm (SPEA) of Zitzler and Thiele [125] by re-evaluating archive solutions and extending the update of the archive. Furthermore, the new concept of a domination dependent re-evaluation interval is presented. Several tests on prototypical functions underline the improvements in convergence speed and robustness of the extended algorithm for noisy objective functions and outliers in the evaluated population.

6.1 Introduction

Although the number of applications in the field of multi-objective (Pareto) optimization is increasing, problems with noisy objective functions are rarely considered, even though noise is present in almost every real-world application. As evolutionary algorithms do not require gradient information, they are already inherently robust against small amounts of noise, a feature which is sufficient for many problems. In several experiments, however, large-amplitude noise is induced from various sources, such as unsteady operating conditions, limited measurement precision, and time averaging in restricted sampling time. In addition, measurements may fail, leading to erroneous outliers, characterized by nonphysical objective values. Standard multi-objective evolutionary algorithms cannot handle these difficulties, and there is a need to extend their basic components to overcome these difficulties.

While for single objective optimization, several studies of noisy objective functions have already been performed [90, 79, 4], for multi-objective optimization,

limited results are available in literature. Averaging the parent population, a remedy for noisy single objective problems, is not useful in this case since a diverse population is desired to converge toward the Pareto front. Two recent publications [112, 57] adapt the Pareto ranking scheme [49] to noisy solutions by defining probabilities of dominance between them. Both methods assume either a uniform or normal distribution of the noise and can benefit from a priori knowledge of its magnitude.
Arnold and Beyer [3] analyzed elitist optimization algorithms like the simplex method or $(\mu + \lambda)$ evolution strategies on noisy functions. In elitist algorithms, solutions may survive an infinite time. They showed that these algorithms can easily be trapped in noisy solutions, if the noise level is significantly large. Furthermore, they demonstrated that the robustness against noise can be improved by re-evaluating solutions in certain intervals. We transfer this concept to multi-objective optimization and apply it in particular to SPEA, an elitist optimization algorithm. We extend the concept by assigning to each solution a dominance-dependent re-evaluation interval. The interval is inversely proportional to the number of solutions that it dominates. Thus, solutions that are very dominant are assigned the shortest interval and are re-evaluated first in order to limit their impact on the overall population. In addition, an extended update mechanism for the archive is defined. These principles are applied to SPEA and denoted as the noise-tolerant SPEA (NT-SPEA).
In this chapter, a survey on modifications for SPEA with respect to robustness against noise is presented. All algorithms are analyzed on noisy and noise-free test functions. The analyzed noise reflects the characteristics of the intended application in Chapter 10.

6.2 Noise and Noise-tolerant Multi-Objective Evolutionary Algorithms

This section describes two different types of noise, which can be present in real-world applications. Furthermore, different modifications for SPEA are presented in order to make the algorithm more robust against noise.

6.2.1 Definition of Noise in Applications

In experiments and industrial configurations, we can always detect different results for repeated measurements of the same operating point. The differences are

attributed to noise and unobserved factors in the setup.
Noise may occur in various areas in the experiment: The setting of the operating conditions is within a limited precision. In the realization, the operating condition may vary over time and finally measurement errors occur. It is up to the careful setup by the experimenter to keep the noise within a limited range. We define this noise, which is present in all measured experiments, as *experimental noise*. It is often modeled by a normal distribution specified by mean and standard deviation. In addition, during an automated optimization cycle, an experimental measurement may fail completely, producing *outliers*, i.e., arbitrary nonphysical results. This occurs very rarely, but may have large impact on the automated process optimization if not recognized and captured by some penalty function. Outliers cannot be described by a statistical model with given mean and deviation, but are best modeled by a probability of occurrence.
Noise and outliers influence the multi-objective optimization process by misleading the selection operation. Hence unrealistic inferior solutions may dominate superior ones, thus delaying or completely misleading the convergence to an unrealistic Pareto front.

6.2.2 Non-Elitist Strength Pareto Evolutionary Algorithm

The presence of noise affects the fitness assigned to an individual. This may cause inferior solutions to occasionally win in the selection process. Multi-objective evolutionary algorithms, which implement elitism, would then select these solutions into the archive, thus misleading the entire optimization run by participating in the selection process. More importantly, these solutions may dominate other solutions in the archive, and in the worst case all other solutions in the archive are then removed. In order to avoid this, a first and simple modification of *original SPEA* of Zitzler and Thiele [125] is proposed. We define a *non-elitist* SPEA algorithm. In each generation, the archive is filled with the nondominated solutions of the current population. Nondominated solutions from previous generations are not considered.

6.2.3 Statistical Strength Pareto Evolutionary Algorithm

Re-evaluating a solution several times and taking the mean as a statistical estimate can decrease the level of noise in an objective function. Implementing this approach into SPEA is simple and is in the following referred to as *statistical SPEA*. The disadvantage of this concept is the increased evaluation cost per solution. For

the performance comparison in the next section, 7 evaluations are averaged per solution.

6.2.4 Estimate Strength Pareto Evolutionary Algorithm

The Estimate Strength Pareto Evolutionary Algorithm (ESPEA) of Teich [112] modifies the SPEA algorithm by introducing a *probability of dominance*. It is assumed that each objective value f cannot be computed exactly, but can be bounded within a *property interval* $[f^{\mathrm{L}}, f^{\mathrm{U}}]$, where f^{L} and f^{U} are the lower and upper bound of the interval, respectively. Teich assumes that the probability of the function value is uniform in the interval. These assumptions lead to the new definition of a probability of dominance. If two solutions with overlapping property intervals are compared, the dominance has to be assigned by a probability. Teich computed the probability for minimizing an arbitrary number of m objectives. If two solutions a and b with the property intervals $[a_i^{\mathrm{L}}, a_i^{\mathrm{U}}]$ and $[b_i^{\mathrm{L}}, b_i^{\mathrm{U}}]$, $i = 1, \ldots, m$, respectively, are compared, the probability that a dominates b is given by

$$p(a \succ b) = \prod_{i=1}^{m} \begin{cases} 0 & \text{, if } a_i^{\mathrm{L}} > b_i^{\mathrm{U}}, \\ 1 & \text{, if } a_i^{\mathrm{U}} < b_i^{\mathrm{L}}, \\ \frac{1}{a_i^{\mathrm{U}} - a_i^{\mathrm{L}}} \int_{y=min\{a_i^{\mathrm{L}}, b_i^{\mathrm{L}}\}}^{b_i^{\mathrm{L}}} dy & \\ + \int_{y=\max\left\{a_i^{\mathrm{L}}, b_i^{\mathrm{L}}\right\}}^{\min\left\{a_i^{\mathrm{U}}, b_i^{\mathrm{U}}\right\}} \frac{b_i^{\mathrm{U}} - y}{b_I^{\mathrm{U}} - b_i^{\mathrm{L}}} dy & \text{, otherwise.} \end{cases} \tag{6.1}$$

Three different dominance relations are distinguished in the equation. First, solution a does not dominate b ($p(a \succ b) = 0$) if at least one lower bound of the property intervals a_i^{L} is larger than the corresponding the upper bound b_i^{U}. Second, the solution a dominates b ($p(a \succ b) = 1$), if the upper bound of all the property interval a_i^{U} are smaller than the lower bounds b_i^{L} for all objectives. In the third case, a dominates b with a certain probability $p(a \succ b) \in]0, 1[$, if for all objectives i the lower bound a_i^{L} is smaller than b_i^{U} and at least one bound a_i^{U} is larger than b_i^{L}.

Assuming that the values for a and b, obtained by test functions or real applications, are in the middle of the property intervals and both intervals are of size 2δ, the interval bounds can be computed as $a_i^{\mathrm{L}} = a_i - \delta$, $a_i^{\mathrm{U}} = a_i + \delta$, $b_i^{\mathrm{L}} = b_i - \delta$

and $b_i^{\mathrm{U}} = b_i + \delta$ and Eqn. 6.1 can be rewritten as

$$p(a \succ b) = \prod_{i=1}^{m} \begin{cases} 0 & \text{, if } a_i > (b_i + 2\delta), \\ 1 & \text{, if } a_i < (b_i - 2\delta), \\ \frac{1}{2\delta}(b_i - a_i + \delta) & \\ +\frac{1}{8\delta^2}\mathrm{sgn}(a_i - b_i)(a_i - b_i)^2 & \text{, otherwise,} \end{cases} \tag{6.2}$$

where sgn is the signum function. Since for the principle of a probability of dominance, the distinction between nondominated and dominated solutions is fuzzy, the archive update needs to be modified. Here, we differ slightly from Teich's update of the archive. First, the current population P_λ is added to the archive A generating an extended archive A'. For each solution a in A', the mean probability R of being dominated by a solution b in A' is computed by:

$$R(a) = \frac{1}{N-1} \sum_{b \in \{A'\}: b \neq a} p(b \succ a), \tag{6.3}$$

where N is the number of solutions in A'.
Then, all solutions with $R(a) > \alpha$ are removed from the archive. The parameter α scales the fuzziness of the archive. For increasing α, more solutions remain in the archive and the archive changes to a more fuzzy nondominated front.
This approach corresponds with the results of Arnold and Beyer [4]. They computed the progress rates of the (μ, λ) evolution strategy for noisy single objective problems and found that selecting a set of μ parents out of λ individuals leads to a higher convergence speed than just selecting the best individual. This observation is in contrast to the noise-free case, where selecting the best solution leads to the highest convergence speed [90]. The (μ, λ) strategy harmonizes with the fuzzy nondominated front.
For better comparison, we use the standard clustering algorithm of SPEA to keep the archive size limited. This is valid, since the core aspect of the ESPEA is the concept of a dominance probability and not the clustering. The fitness is assigned in two steps. First, the fitness S of each archive solution a is computed as:

$$S(a) = \frac{1}{N+1} \sum_{b \in \{P \cup A\}} p(b \succ a), \tag{6.4}$$

where N is the number of solutions in the unified set of archive A and population P_λ. The fitness of a solution in the population is equal to one plus the fitness

of the archive solutions, by which it is dominated with a probability larger than a threshold α. ESPEA contains two additional strategy parameters, which are the threshold α and the size of the property intervals δ. A drawback is that one needs to set the interval size before optimizing, such that the interval reflects the size of the noise in the objective function. In addition, if the noise varies for the different objectives, separate property interval sizes have to be chosen for the different objectives.

6.2.5 Noise-tolerant Strength Pareto Evolutionary Algorithm

We propose two modifications for SPEA and denote the resulting algorithm as the Noise-tolerant Strength Pareto Evolutionary Algorithm (NT-SPEA). Arnold and Beyer [3] showed that elitist algorithms can easily be trapped in noisy solutions, if the noise level is significantly large. To overcome this problem, we described in Subsection 6.2.2 a non-elitist SPEA that selects individuals by a (μ, λ) scheme. One disadvantage of this algorithm is that noise reduces the selection pressure [79], suggesting that elitism, which increases the selection pressure by conserving elite solutions, should be used to compensate. To successfully use elitism in a noisy environment, further modifications are needed to ensure fast convergence while maintaining robustness to noise.
Arnold and Beyer [3] proposed to keep the elitist algorithm, but re-evaluate solutions in certain intervals and assign always the re-evaluated objectives to the solutions. We transfer this proposal to multi-objective optimization and extend it by the following modifications applicable to SPEA:

1. *Re-evaluation of archive solutions with dominance-dependent intervals:* Following the idea of Arnold and Beyer [2, 3], we re-evaluate solutions in the archive of SPEA in certain intervals. Re-evaluation allows solutions to stay in the archive for an infinite time, although, their function values will change due to the noise in the evaluation. We extend this idea by assigning a dominance-dependent re-evaluation interval κ to each individual instead of a fixed interval. The interval is variable and related to the dominance of a solution. It is shortened, if the solution dominates a major part of the archive. This limits the impact of a solution and safeguards against outliers.

2. *Extended update of the archive:* SPEA updates the archive in every generation by adding the current population to the archive and then removing all dominated solutions. We extend the update to *all* solutions with non-expired

re-evaluation intervals. This hinders loss of information, since solutions that were removed by clustering or domination may reenter the archive.

With these modifications, NT-SPEA uses the advantage of an archive as convergence accelerator, but it reduces the risk induced by outliers.
The dominance-dependent re-evaluation interval of an individual is assigned according to Fig. 6.1. The interval is measured in generations. For dominating less than a fraction c_1 of the archive A, the maximal interval length $\kappa = \kappa_{\max}$ is assigned to the individual. For dominating more than a fraction c_2 of A, the minimal interval length of $\kappa = 1$ is assigned. In between these two fractions, the interval length is interpolated in discrete steps of one generation. In mathematical form, the interval length is computed as:

$$\kappa(c) = \begin{cases} \kappa_{\max} & \text{, if } c < c_1, \\ 1 & \text{, if } c > c_2, \\ \text{rnd}\left(\kappa_{\max} - \frac{c-c_1}{c_2-c_1}(\kappa_{\max} - 1)\right) & \text{, otherwise,} \end{cases} \tag{6.5}$$

where c is the fraction of the archive that is dominated by a solution in the population and rnd rounds the result to the next integer. The dominance-dependent re-evaluation interval κ reduces the impact of a solution. An individual that dominates a large fraction of the archive has a high chance of being selected in the selection process, but is assigned the shortest interval length.
The re-evaluation allows nondominated solutions to stay in the selection process as long as the re-evaluation leads to a nondominated solution. In the case of an outlier, it is not likely, that the re-evaluated solution is again an outlier with good objective values and hence it would be deleted from the archive. On the other hand, solutions with good design variable settings are likely to be nondominated again, assuming that the effect of noise is limited.
The extended update considers the nondominated solutions among all solutions with non-expired re-evaluation intervals for the update of the archive. Since the assigned intervals differs between the solutions, the set of nondominated solutions changes. Dominated solutions become nondominated ones dependent on the re-evaluation result of their dominator. This is especially important if a noisy solution or an outlier dominates a large fraction of the archive. The dominated solutions are then removed from the archive. The noisy solution or outlier is assigned a short re-evaluation interval. After the re-evaluation, the removed nondominated ones may be re-selected to the archive. With the original update of SPEA, their information is lost. After the update of the archive, the clustering algorithm of SPEA is used in order to get a limited number of uniformly distributed archive solutions. In SPEA,

solutions of the population and archive participate in the selection process. With these two modifications, the noise-tolerant SPEA is given by:

Algorithm NT-SPEA

1. **begin**
2. Generate an initial population P_λ of random individuals and an empty archive A.
3. Evaluate the individuals in P_λ.
4. **while** termination criterion is not fulfilled **do**
5. Assign interval length: Compute for each individual in P_λ the fraction of the archive A that it dominates. The interval length κ of an individual is inverse proportional to the fraction (see Fig. 6.1).
6. Update A: Remove all solutions from A and refill it with all solutions, whose re-evaluation interval is not expired. Then remove all dominated solutions. Limit the size of A by clustering.
7. Fitness assignment: Assign fitness to the individuals in P_λ and A.
8. Selection: Use tournament selection for selecting the parent population P_μ from $P_\lambda \cup A$.
9. New population: Generate a new population P_λ by recombining and mutating individuals from P_μ.
10. Evaluate the individuals in P_λ.
11. Re-evaluation: re-evaluate the solutions from A depending on their re-evaluation interval.
12. **end while**
13. **end**

6.3 Performance Comparison

6.3.1 Generation of Noisy Test Functions

From Section 3.4, the two-objective test function DEB and the three-objective test function SPH-3 are chosen as noise-free functions for the performance comparison. These two functions will be used as a basis to generate noisy test functions

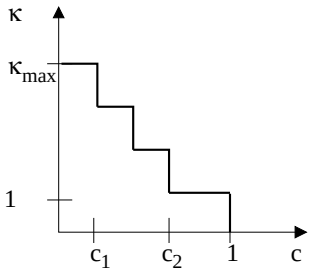

Figure 6.1: Dependence of the re-evaluation interval κ of an individual on the fraction c of the archive that it dominates. κ decreases from a maximal value $\kappa_{\max}$, if the individual dominates more than the fraction c_1 until it reaches a minimal value of $\kappa = 1$ at c_2 .

and for comparing the performance of the algorithms to the noise free case.
For the *experimental noise*, we assume that the noise can be modeled by a normal distribution with zero mean and standard deviation σ_N and a noisy test function is generated by adding noise to each objective function f_i with:

$$f_i = f_i + z_i, \;\; z_i \sim \mathcal{N}(0, \sigma_N^2), \tag{6.6}$$

where z_i is a random number, taken from a normally distribution $\mathcal{N}(0, \sigma_N^2)$ with zero mean and standard deviation σ_N. The standard deviation is set to $\sigma_N = 0.8$, which is about half the maximal difference in the objective function values for the Pareto solutions. The random number is computed separately for each objective and individual in the evolution.
The second type of noise that was introduced in Subsection 6.2.1 refers to the random occurrence of *outliers*. For the modeling in a test function, we define a probability p_o for the occurrence of an outlier. Since we consider the minimization, reducing an objective value has a stronger influence on the optimization process than increasing the value by giving a solution with reduced values a higher chance to survive. Therefore, we reduce the objective value by dividing it by a factor of 10, if an outlier occurs. The large factor is chosen in order to produce a significant change in the objective value. We write test functions with outliers by:

$$f_i = \begin{cases} \frac{1}{10} f_i & \text{, if } p < p_o, \;\; p \sim \mathcal{U}(0,1) \\ f_i & \text{, otherwise} \end{cases} , \tag{6.7}$$

where $\mathcal{U}(0,1)$ is a uniform distribution of random numbers in the interval $[0,1]$. The probability of an outlier is small and set to $p_o = 0.01$.
Using the two test functions DEB and SPH-3 with no noise, normally distributed noise, and outliers results in a total number of 6 test functions.

6.3.2 Experimental Results

In the following, the performance of the algorithms introduced in Section 6.2 are numerically analyzed on the 6 test functions. For all optimization algorithms, a parent and child population of $\mu = \lambda = 60$ is used, with an archive size of 20 for the two-objective test functions and a size of 50 for the three-objective test functions. The recombination and mutation operators of Section 3.3 are used. The number of design variables n is set to $n = 7$. This number is equal to the number of design variables of the burner optimization problem, which is addressed in Chapter 10. For the mutation operator, the standard deviation σ is set to 10% of the interval size in which the variable is defined, and a mutation probability p_{M} of 20%. The result of 100 optimization runs is averaged for each test function.
The quality of a multi-objective optimization algorithm depends on the convergence speed of the algorithm as well as on the uniformity of the approximation of the Pareto front. To compare the performance of the different algorithms, the performance measure P of Section 3.5 is chosen. P is defined as an approximation of the mean distance in decision space of the Pareto front solutions to the respective closest evaluated solution of the optimization run.
Some of the analyzed algorithms contain heuristic parameters. No heuristic parameters have to be set for SPEA, the non-elitist SPEA and the statistical SPEA. For ESPEA, no parameter settings are given by the author [112]. Thus, a performance analysis is made for all combinations of a threshold $\alpha \in [0.008, 0.01, 0.015, 0.02, 0.04, 0.07, 0.1, 0.2, 0.5]$ and a property interval size of $(a_i^{\mathrm{U}} - a_i^{\mathrm{L}}) = 2\delta \in [0, 0.2, 0.4, 1.0, 2.0, 3.0, 4.0]$. On average, the best results of ESPEA on all test problems is obtained with $\alpha = 0.04$ and $\delta = 0.2$. Since the parameters of ESPEA were optimized for the considered test functions, the general performance of the algorithm might be worse. For NT-SPEA, the fractions c_1 and c_2 are set to 0.1 and 0.3, respectively and a maximal interval size $\kappa_{\max} = 4$ is used. A discussion of these settings is introduced in the next section.
The results for all test functions are given in Fig. 6.2. The performance measure P is plotted in a logarithmic scale over the number of evaluated solutions N. The measure P reflects the mean distance in decision space of each Pareto solution to its closest approximation by the set of evaluated solutions.

First, the two-objective and noise-free test function DEB is considered. At the beginning of the optimization run, P drops rapidly and levels off at the end of the run. The optimization levels off, since a limited population and archive size cannot exactly approximate the Pareto front and the convergence speed decreases at a certain level as shown in Chapter 5.
The performance of the different algorithms varies significantly. The slowest convergence is observed for the statistical SPEA. The statistical SPEA computes for each solution the mean of 7 evaluations. For a certain number of evaluated solutions, the statistical SPEA proceeds compared to SPEA just by $1/7$ of the number of generations.
The second slowest is the non-elitist SPEA, due to the lack of an archive for storing the nondominated solutions. ESPEA shows better performance since the algorithm contains an archive. In contrast to the original SPEA, the archive of ESPEA can contain dominated solutions. Increasing α or the property interval size raises this fraction of dominated solutions and decreases the selection pressure. The best performance can be found for NT-SPEA and the original SPEA. In contrast to the ESPEA, the archive of both NT-SPEA and the original SPEA contain just nondominated solutions and thus the selection pressure is higher. NT-SPEA re-evaluates solutions, although this is not necessary for a noise-free test function. However, since the fraction of re-evaluated solutions is small, this disadvantage is small and the algorithm performs well even on noise-free test problem.
As second test function, normally distributed noise is added to the test function DEB. The standard deviation of the noise is set to $\sigma_N = 0.8$ and is about the same magnitude as the difference in objective values of points on the Pareto front. The convergence behavior of the different algorithms is also illustrated in Fig. 6.2. The convergence speed for the noisy test function is drastically reduced compared to the noise-free test function 1 and the convergence levels off at a higher value of P. Excluding the statistical SPEA, the difference in performance between the algorithms is smaller compared to the noise free case. Here, elitism in the form of the original SPEA is a disadvantage. The non-elitist SPEA performs superior to the original SPEA. ESPEA converges about equally fast as the non-elitist SPEA. NT-SPEA converges best, due to the compromise between using an archive and limiting the risk of getting stuck in noisy solutions by a limited and dominance-dependent re-evaluation interval of solutions.
For analyzing the effect of outliers, an error probability of $p_o = 1\%$ per objective is defined for test function DEB. Since the test function has two objectives, the probability that at least one objective contains an error is therefore about 2%. In other words, about one individual in the population of 60 individuals contains an

error and is thus an outlier.
The results of the numerical analysis are given in Fig. 6.2. Again, NT-SPEA performs best and the non-elitist SPEA performs better than the original SPEA. Analysis of the convergence of the original SPEA shows that the algorithm gets stuck in the outliers. Outliers occur with a small probability and it is unlikely that they are removed from the archive. This explains why the non-elitist SPEA performs significantly better than the original one. ESPEA shows no advantage for this test function, compared to the original SPEA.
The performance of the NT-SPEA is superior to all other algorithms. It avoids getting stuck in outliers. The shortest re-evaluation interval is assigned to outliers that dominate a large part of the Pareto front. Since they are re-evaluated first and the probability that an error occurs again is low, they will be removed from the archive. This allows solutions with larger re-evaluation interval than the outliers to reenter the archive after the outlier is removed.
Now the three-objective test function SPH-3 is analyzed. Compared to the two-objective test function DEB, obtaining a solution of the same quality in terms of the performance measure P requires noticeably more evaluations. However, The relative performance between algorithms is consistent with the two-objective test function DEB. The best agreement is found for the noise free case. SPEA and NT-SPEA perform clearly best on this function.
NT-SPEA, ESPEA, and the non-elitist SPEA show about equal convergence on SPH-3 with normally distributed noise. The differences are within the sampling tolerance. Slightly inferior convergence is obtained with the original SPEA, demonstrating again the disadvantage of elitism in form of an archive of nondominated solutions, which may be kept an infinite time.
In the last performance comparison, outliers are added to the test function SPH-3 with an error probability of 1% per objective. For this three-objective problem, the probability that an individual contains an error in at least one objective is about 3%, thus about 2 of the 60 individuals in a population are outliers. Similar to the test function DEB with outliers, NT-SPEA performs best, but here the original SPEA performs slightly superior than the non-elitist SPEA.
Summarizing the results from the 6 test functions, we found that elitism, implemented by the archive of the original SPEA, is a convergence accelerator for noise-free problems. For noisy problems it is a disadvantage and the non-elitist SPEA performs better on average.
The relative behavior of the different algorithms shows similar tendencies for 2 and 3 objectives. For 3 objectives, however, the differences are smaller.
The statistical SPEA has the drawback of multiple function evaluations per solu-

tion and converges, except for the test functions with outliers, slower than all other algorithms. Again, the differences are smaller for 3 objectives.
Setting the parameters α and δ of ESPEA is very problem dependent and leads to large performance differences. The best convergence for the noise-free test function DEB is obtained for $\alpha = 0.008$ and $\delta = 0$, a setting which leads to an algorithm and convergence similar to the original SPEA. For test function DEB with noise, increasing α to 0.04, but keeping a property interval $\delta = 0$ leads to the best result. Increasing α introduces dominated solutions to the archive. A positive effect of a property interval $\delta > 0$ could not be found for the noisy functions. For test function DEB with outliers, the ideal settings are $\alpha = 0.2$ and $\delta = 1.5$. These settings differ tremendously from the previous two settings, especially in the property interval, but the performance on this test function is still poor. In addition, compared to the other algorithms, ESPEA performs better for 2 objectives than for 3 objectives.
For NT-SPEA, a marginal problem dependence is found for the parameters c_1, c_2 and $\kappa_{\max}$. This will be analyzed in more detail in the next section.
Comparing the mean behavior of the algorithms over all test functions, NT-SPEA performs clearly best. One possibility for a mean performance analysis for all 6 test functions is obtained by summing the minimal value of P over all test function. NT-SPEA clearly results in the smallest value with $\sum_{i=1}^{6} \min(P_i) = 1.75$. NT-SPEA is followed by the original SPEA (1.97), ESPEA (2.02) and the non-elitist SPEA (2.17) and finally the statistical SPEA (3.21).

6.3.3 Discussion of the Heuristic Parameters c_1, c_2 and $\kappa_{\max}$ in NT-SPEA

The NT-SPEA algorithm, as described in Subsection 6.2.5, includes the heuristic parameters c_1, c_2 and $\kappa_{\max}$. Such parameters are often set by experimental analysis on different test functions. We proposed to set the parameters as $c_1 = 0.1$, $c_2 = 0.3$ and $\kappa_{\max} = 4$. The guiding concepts behind the settings are the following: The value for the maximal interval length $\kappa_{\max}$ is a trade-off between noise-free and noisy test functions. For noise-free functions, re-evaluating does not lead to new information, since the re-evaluated solution equals the original one. Thus, a larger maximal interval length (and also increased values for c_1 and c_2) is preferable for avoiding the re-evaluation of solutions.
In contrast, for noisy problems, it is reasonable to limit the interval for re-evaluating a solution in the archive, in order to avoid that the entire optimization process is misled by noisy archive solutions. Here, we store a solution in the

archive for at most 4 generations. The time has to be short enough to avoid that the optimization is misled by very noisy archive solutions (outliers). In addition, the time has to be larger than one generation, since dominated solutions should be able to re-enter the archive after their dominator (an outlier), is removed due to its re-evaluation. We assume that a solution, which dominates less than 10% of the archive ($= c_1$), should be assigned the maximal interval length $\kappa = \kappa_{\max}$, while a solution, which dominates more than 30% ($= c_2$) should be re-evaluated already in the next generation.

The following parameter analysis underlines that the parameter settings are robust and their influence on the algorithm performance is minor over a large parameter range. The performance analysis of Subsection 6.3.2 is repeated with all possible combinations of $c_1, c_2 \in [0.05, 0.1, 0.15, 0.2, 0.3, 0.5]$ and $\kappa_{\max} \in [2, 4, 8]$, such that the constraint $c_1 < c_2$ is fulfilled. For all combinations and all test functions, the performance measure P was computed as the mean of 100 independent runs. Table 6.1 contains the obtained performance measures $\min(P)$ and $\max(P)$ for the best and worst parameter combination, respectively and the referring heuristic parameters for all test functions.

For the noise-free test functions DEB and SPH-3, all settings performed almost identically and also better than the non-elitist SPEA, the statistical SPEA and ES-PEA. Re-evaluation is not necessary, since the original and re-evaluated solutions are identical. Thus, re-evaluating many solutions will decrease the performance. Besides influencing the number of re-evaluated solutions, the maximal interval length $\kappa_{\max}$ has a second effect. Since the archive is updated with all solutions with non-expired interval, solutions may re-enter the archive after they were removed by clustering. This appears to have a negative effect on the noise-free function, since one setup with $\kappa_{\max} = 8$ performed worst.

Differences in the performance are also small for the test functions DEB and SPH-3 with experimental (normally distributed) noise. However, the best results are obtained for $\kappa_{\max} = 4$, and setting $\kappa_{\max} = 2$ appears too short. In general, on these two test functions the differences between the analyzed SPEA implementations were also the smallest.

Adding a small percentage of outliers to the test functions seems to have a major effect on the performance of the different algorithms. Since SPEA performs poorly for test function DEB with outliers, the setting of the NT-SPEA algorithm, which is closest to SPEA, performs also worst. Due to the large maximal interval length $\kappa_{\max} = 8$ together with the large values $c_1 = 0.2$, $c_2 = 0.5$, the algorithm is in danger of getting stuck in outliers with a long maximal interval length, thus misleading the optimization. Similarly for test function SPH-3 with outliers, the

test function	result	Heuristic Parameters		
		c_1	c_2	$\kappa_{\max}$
DEB	min(P)=0.099	0.10	0.20	2
	max(P)=0.113	0.15	0.30	8
DEB with exp. noise	min(P)=0.804	0.10	0.20	4
	max(P)=0.835	0.05	0.10	2
DEB with outliers	min(P)=0.346	0.10	0.20	4
	max(P)=0.661	0.20	0.50	8
SPH-3	min(P)=0.174	0.05	0.15	4
	max(P)=0.218	0.10	0.20	2
SPH-3 with exp. noise	min(P)=0.345	0.10	0.20	4
	max(P)=0.449	0.10	0.15	2
SPH-3 with outliers	min(P)=0.583	0.10	0.30	8
	max(P)=0.669	0.10	0.15	2

Table 6.1: Sensitivity analysis of NT-SPEA on the heuristic parameters c_1, c_2 and $\kappa_{\max}$. NT-SPEA shows small performance variation over a wide range of parameter settings.

setting that is very close to the non-elitist SPEA (i.e., $\kappa_{\max} = 2$, c_1, and c_2 are small) performed worst.
Summarizing the results of for all test functions, the heuristic parameters c_1, c_2 and $\kappa_{\max}$ can be set general enough in order to perform well on noise-free and noisy problems, as well as problems with a rare occurrence of outliers. Varying the settings over a large range has a minor effect on the performance. Besides the better performance, this is a major advantage to ESPEA, since ESPEA is very sensitive the heuristic parameters.

6.4 Conclusions

A novel noise-tolerant multi-objective evolutionary algorithm (NT-SPEA) was introduced with increased robustness for applications prone to noise and outliers. The algorithm transfers the concepts of re-evaluating solutions [3] to multi-objective optimization and extends the concept by introducing a dominance-dependent re-evaluation interval. In addition, an extended update mechanism is defined for the archive. These concepts have been applied to SPEA and can also

be transferred to other elitist multi-objective algorithms.
A convergence comparison for various implementations of SPEA has been performed on noisy and noise-free test functions. In general, a decrease in convergence is observed when noise is introduced. The concept of elitism is analyzed in the presence of noise. In the absence of noise, elitism can be used as a convergence accelerator. However, for different types of noise, elitism can imply a significant disadvantage, since the optimization can get misled by outliers.
The NT-SPEA overcomes the problem by re-evaluating archive solutions in dominance-dependent intervals and extending the update of the archive. For the noise-free test problems, NT-SPEA shows similar convergence to the original SPEA, which converges best. This is a major advantage compared to a non-elitist and a statistical implementation of SPEA and the ESPEA.
While NT-SPEA performs equal or superior to the best of the other implementations for problems with normally distributed noise, it clearly outperforms all algorithms for problems with outliers. The discussion of the heuristic parameters in NT-SPEA shows that they have minor influence on the performance for a wide parameter range. A further advantage, which is not discussed in the paper, is that NT-SPEA can handle moving optima over time or changing environmental conditions. The algorithm re-evaluates solutions in certain intervals and thus, the objective values change as the environmental conditions vary.

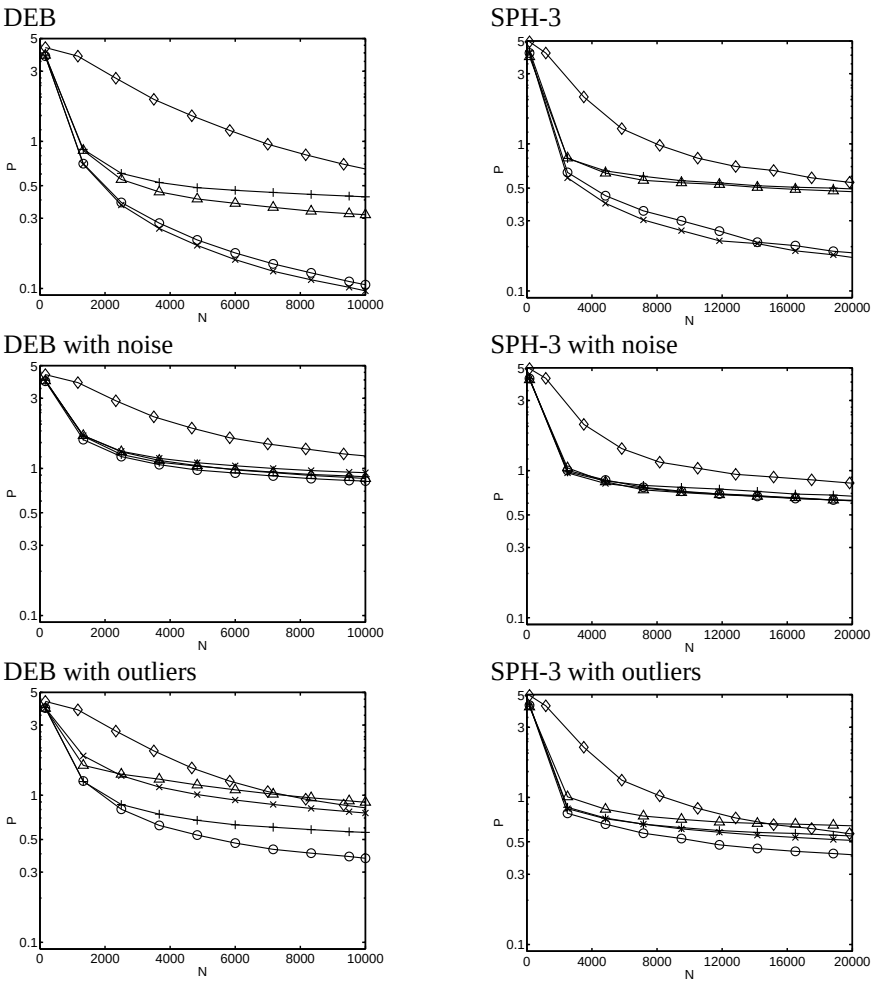

Figure 6.2: Convergence of the NT-SPEA [circular symbol] on test functions DEB and SPH-3 with 2 and 3 objectives, respectively. NT-SPEA is compared with the original SPEA [cross symbol], the non-elitist SPEA [plus symbol], the statistical SPEA [diamond] and ESPEA [triangle]. In addition to the noise-free original test functions, normally distributed noise and outliers are considered.

Chapter 7

Growing Self-Organizing Maps for Multi-Objective Optimization

---

It is well known that selection operators from multi-objective optimization algorithms differ from single objective ones. However, it is far from clear if and how variation operators such as mutation and recombination operators should differ. While the goal in single objective optimization is to converge to the (global) optimum, the goals of multi-objective optimization are to converge to the Pareto front and to spread the individuals along the front. Using density estimators in the fitness assignment of the selection operator promotes the spreading, but should spreading be also promoted by the variation operators?
We analyze standard recombination operators from single objective optimization for Pareto optimization by providing theoretical reasoning and test problem results. Furthermore, we introduce a recombination operator based on Self-Organizing Maps (SOMs) as an operator designed for multi-objective evolutionary algorithms. SOMs are a subclass of neural networks that allow a mapping of a high dimensional input space to a regular lattice of neurons. In the context of multi-objective evolutionary algorithms, the parent population is mapped onto a lattice of neurons, and recombination is performed within the lattice.
In the beginning of the optimization process, the number of nondominated solutions in the parent population is usually small and increases as soon as the Pareto front is located and solutions start to spread along the Pareto front. Simultaneously, solutions in the parent population start to differ as they converge to various areas of the Pareto front. The SOM accounts for this effect by growing the lattice size proportionally to the number of nondominated solutions thereby enabling the network to learn the locally varying properties of parent solutions at the time they occur in the optimization. The resulting algorithm leads to a fast and uniform approximation of the Pareto front. Advantages and disadvantages of the proposed algorithm are discussed.

7.1 Introduction

In Pareto optimization, recent research has focused on the development of multi-objective selection operators and in particular on fitness assignment techniques [125, 13]. Standard selection operators for multi-objective algorithms such as SPEA2 [124] and NSGA-II [26] do not describe variation operators such as mutation or recombination operators. The variation operators are often directly applied from single-objective algorithms [71]. To compare the performance on continuous problems, polynomial distributed mutation and simulated binary crossover (SBX) [25] are often used [26, 125]. Both variation operators do not adapt their parameters in the evolutionary search, thus they do not exploit any knowledge from the evolution.

However, in single objective optimization, the key components are the variation operators with a focus either on the mutation operator (Evolution Strategies) or the recombination operator (Genetic Algorithms) [9]. In Evolution Strategies (ES), the mutation operator is adaptive and exploits knowledge obtained during optimization to adapt the mutation distribution. This is especially important if large search spaces are considered and a close approximation of the optimum is desired. Limited effort has been made in the recent years in order to apply adaptation in multi-objective optimization. Most of these methods are inspired by single objective algorithms. Abbass [1] implemented a Pareto optimization algorithm with recombination and mutation based on the differential evolution [110]. He used self-adaptation in order to find the appropriate crossover and mutation rates (probabilities). Sbalzarini *et al.* [97] use a simple self-adaptation scheme for mutating step sizes. Each individual in the population is assigned an individual step size for each decision variable. From one generation to the next, these step sizes are mutated by either increasing or decreasing them by 50% or keeping them constant, each with a probability of $1/3$. Kursawe [68] and Laumanns [71] developed two further implementations of self-adaptation, closer to Schwefel's original implementation [101]. Kursawe performs selection based on a randomly chosen objective. Each individual contains a separate vector of decision variables and step sizes for each objective (poliploid individuals). Laumanns *et al.* assign a single step size to each individual, which yields an isotropic mutation distribution.

In Pareto optimization, recombination operators are also an open issue. Recombination properties such as mating restrictions are debatable [116]. In the beginning of the optimization, mating between all solutions seems promising in order to enforce the exchange of information within the population. However, at the final convergence of the population to the Pareto front, mating restrictions seem reason-

able, since solutions are well adapted to the local topology and thus are too different to recombine. Furthermore, recombination in multi-objective optimization differs from single objective optimization. In single objective Evolution Strategies, recombination is often performed by averaging several parents as , e.g., in [51]. This usually increases the convergence speed to the single optimum [52]. Such a recombination seems not suitable for multi-objective optimization, as it would destroy diversity. A diverse parent population is required in order to spread individuals along the Pareto front.

Recombination is a fundamental operator in any parallel search process, as recombination spreads information (like decision variable values and strategy parameters) among offspring might benefit from combining preferable properties of their parents. In single objective optimization, this leads to convergence acceleration. Costa and Olivaira [22] show that the performance of a multi-objective evolutionary algorithm improves when adding discrete or intermediate recombination. However, it is still an open question if standard recombination operators are efficient in multi-objective optimization or if special operators need to be developed.

This Chapteraddresses this question by analyzing standard recombination operators and by proposing a new operator based on Self-Organizing Maps (SOMs) [120, 65]. A SOM is a neural net consisting of a set of neurons and a lattice structure that allows approximating and interpolating a probability distribution or a discrete set of points. Recently, Milano *et al.* [78] introduced SOMS as an adaptation method for single objective evolutionary algorithms. SOMs are trained to approximate the area around the currently best solution. The map is used to sample new solutions with a higher sampling probability in this area.

In this Chapter, we provide an extension to multi-objective evolutionary algorithms. SOMs are formulated as an adaptive recombination operator. The SOM is trained on the current parent population of the evolution. Since the SOM provides a (lower dimensional) interpolation of the parent population, choosing a random point within the volume covered by the SOM represents an intermediate recombination of the parent population. Extrapolation is possible by using multiplicative factors to increase the lattice size. Furthermore the operator is improved by a growing lattice structures [43]. Growing lattices avoid the problem of defining appropriate lattice structures before the optimization run. Instead, the structure evolves while training the network on the current data. This increases also the quality of the mapping and reduces the risk of topological defects [43]. In addition, the size of the network can be adapted to the usually growing number of nondominated solutions. The focus is on recombination, thus we implement self-adaptive mutation [101] as a standard mutation operator.

First, definitions and objectives in multi-objective optimization are briefly outlined. Then, different recombination operators are introduced. In a first step, these operators are theoretically compared. In a second step, analyses on test problems are performed.

7.2 Multi-Objective Optimization as Two Step Process

In multi-objective optimization, conflicting objectives are usually considered such that optimization run results in an approximation of the Pareto front by a set of nondominated solutions. Zitzler *et al.* [124] discussed the convergence on the multi-objective sphere function (denoted as SPH-m) as a two step process. The first step (i.e., the beginning of the optimization) is similar to a single objective optimization. Compared to the search space, the Pareto front is small and thus converging towards the Pareto front is similar to converging to a single optimum. This step is described as the problem of locating "the region of the Pareto optimal set". The second step begins when the optimization converged already into the region of the Pareto optimal set and now, in addition to converging, the aim is also to spread the population along the Pareto optimal set such that a uniform approximation is obtained.

7.3 Recombination Operators

In nature, evolution is a highly parallel process using large populations. Most species recombine their genetic information by mating two parents. Mating allows spreading genetic information in the population and the offspring might benefit from combining preferable properties of their parents.
In Evolutionary Algorithms, the decision variables and strategy parameters represent the genetic information and mating is also denoted as recombination or cross-over. Recombination is necessary especially if mutation schemes like self-adaptation [90, 101] are considered. In these schemes, the adaptation of the strategy parameters is imperfect. Recombination is considered a correction mechanism and performed by averaging the strategy parameters [7]. However, it is not advisable to average all parents, as the collective performance of a diverse population can be superior to aggregating all information into a "super-individual"[7].
In the following, we consider a parent population $\mathbf{x}_{j,j=1,\ldots,\mu}$ of size μ. We denote a recombined child of the parents as x'. Several recombination operators exist in literature and differ mainly in the recombination method and the number of parents

ρ recombined. For each child the ρ parents are randomly chosen. Special cases are $\rho = 2$ (binary recombination) and $\rho = \mu$ (global recombination).

7.3.1 Standard Recombination Operators from Single Objective Evolutionary Algorithms

We briefly outline 4 different recombination operators for recombining the decision variables. For the strategy parameters, we follow the proposal of Bäck *et al.* [7] and compute the mean of all ρ parents of the child. Denoting a strategy parameter as σ, recombination is performed by:

$$\sigma' = \frac{1}{\rho} \sum_{j=1}^{\rho} \sigma_j. \tag{7.1}$$

The first 3 recombination operators are a global and two binary recombination operators as given in [9]. For *global intermediate recombination*, the mean of all parents in the parent population is computed:

$$x'_i = \frac{1}{\mu} \sum_{j=1}^{\mu} x_{j,i}. \tag{7.2}$$

For binary recombination, a child $\mathbf{x}'$ is created from two parents $\mathbf{x}_a$ and $\mathbf{x}_b$. The first approach is *discrete recombination* and each decision variable of the child is randomly chosen from one the two parents with:

$$x'_i = \begin{cases} x_{a,i} & \text{if } p_i < 0.5, p_i \sim \mathcal{U}(0,1) \\ x_{b,i} & \text{otherwise,} \end{cases} \tag{7.3}$$

where $\mathcal{U}(0,1)$ is a uniform distribution of random number between 0 and 1. For *intermediate recombination*, the decision variables of the child are a linearly interpolated from the two parents by:

$$x'_i = \alpha x_{a,i} + (1 - \alpha) x_{b,i}, \tag{7.4}$$

where $\alpha \sim \mathcal{U}(0,1)$. Deb and Agrawal [25] propose a recombination operator that is similar to single point crossover of binary strings. The operator is denoted as *simulated binary crossover* (SBX) and is used frequently in the performance analysis of multi-objective selection operators as in Zitzler *et al.* [124] or Deb

et al. [26]. Here always two parents $\mathbf{x}_a$ and $\mathbf{x}_b$ are selected at random and two children $\mathbf{x}'_a$ and $\mathbf{x}'_b$ are generated by:

$$x'_{a,i} = \beta x_{a,i} + (1 - \beta_i) x_{b,i} \tag{7.5}$$

$$x'_{b,i} = \beta x_{b,i} + (1 - \beta_i) x_{a,i}, \tag{7.6}$$

where each β_i is taken from a probability distribution: With equal probability the value of β is set to one, or is taken from a polynomial probability distribution with the highest probability density at a value of one. We use the typical parameter value of $\eta_c = 20$ that controls the probability distribution.

7.3.2 Recombination Operator based on Self-Organizing Maps

Growing Self-Organizing Maps

Self-Organizing Maps (SOM), as proposed by Willshaw and von der Malsburg [120] and Kohonen [65], define a mapping of a set of data points $\mathbf{x}_{j,\ j=1,\ldots,N_D}$ from a highly dimensional input space $\mathbf{x}_j \in \mathbb{R}^n$ onto a regular lattice of neurons of usually lower dimension n_{SOM}. The mapping preserves neighborhood information of the input data by mapping neighboring input data onto the same or onto neighboring neurons. It also preserves relative density differences in the input data by assigning larger fractions of the network to areas of higher density. However, it is important to mention that the relative density differences for the neurons are lower than for the data. SOMs have been extensively applied in the field of complex data analysis and processing, e.g., to data compression [104] or to discrete approximation of continuous probability distributions [41].
The SOM consists of an n_{SOM}-dimensional quadrilateral lattice of N_{SOM} neurons. We define neighboring neurons as those that are directly connected in the lattice. The distance between neighboring neurons in the lattice space is of unit length. Fig. 7.1a illustrates a SOM with $N_{\text{SOM}} = 25$ neurons and a two-dimensional lattice ($n_{\text{SOM}} = 2$) in the lattice space. A reference vector $\mathbf{w}_i \in \mathbb{R}^n$ in the input space is associated with each neuron i. For the considered SOM, Fig. 7.1b shows a possible configuration of the reference vectors together with the corresponding lattice in the input space. The response of the network to an input $\mathbf{x}_j \in \mathbb{R}^n$ is defined as the best matching neuron c, measured in the input space:

$$c(\mathbf{x}_j) = \arg\min_i \{||\mathbf{x}_j - \mathbf{w}_i||_2\} \tag{7.7}$$

The SOM is trained on the set of input data $\{\mathbf{x}_j\}_{j=1}^{N_D}$ by unsupervised learning. To each point $\mathbf{x}_j$ the response c is computed and the reference vector $\mathbf{w}_i$ of each

neuron is updated so as to become closer to the input $\mathbf{x}_j$ by the update rule [65]:

$$\mathbf{w}_i^{\text{new}} = \mathbf{w}_i^{\text{old}} + h(c,i) \cdot (\mathbf{x}_j - \mathbf{w}_i^{\text{old}}), \quad i = 1, \ldots, N_{\text{SOM}}, \tag{7.8}$$

where $h(c,i)$ is the so-called *neighborhood kernel*, defined such that $h(c,c) = 1$, $h(c,i) \geq 0,\ i = 1, \ldots, N_{\text{SOM}}$ and given by:

$$h(c,i) = \alpha \exp\left(-\frac{r(c,i)^2}{2\sigma_{\text{SOM}}^2}\right) \tag{7.9}$$

where α is the learning rate, $r(c,i)$ is the Euclidean distance between node c and node i in the lattice space and σ_{SOM} defines the decay of the learning rate over the lattice distance. Note that the distance between two neighboring neurons in the lattice space is always of unit length. Choosing each of the N_D data points once for training is referred to as one *training epoch* e.

The quality of the mapping of the SOM depends on the structure of the lattice, which could be defined before training. This is a critical issue, given unknown data sets. One possibility to overcome this problem is proposed by Fritzke [43], who introduced a *growing grid* structure. The grid is initialized as a small network equal to a hypercube with one neuron in each vertice of the cube and the edges are the connections between neighboring neurons. For example, the initial network for a one, two and three-dimensional lattice is a line with two neurons, a square with 4 neurons or a cube with 8 neurons, respectively. In the learning process, the grid grows by adding neurons until a certain quality of the mapping is reached.

To each neuron i in the network, a resource variable τ_i is assigned and initially set to zero. In the training the network, τ_i is increased by one, if neuron i is the best matching neuron for an input. In certain intervals, the network size is increased. The neuron with the largest resource variable τ_i is selected. Then, from its direct neighbors, the neuron with the largest τ value is selected. Connecting the two selected neurons specifies a lattice direction $l \in [1, \ldots, n_{\text{SOM}}]$. In this direction, the number of neurons is increased by one. The resulting additional neurons are obtained by inserting a layer of neurons between the two selected neurons by an intermediate interpolation as illustrated in Fig. 7.2 for a two-dimensional network. Then, all τ_i values are reset to zero and the training is continued. When training a SOM, there is always a certain probability that topological defects arise. One possible defect is a partial folding of the SOM in the input space. The quality of the network can be measured by the *average distortion measure* M_{d} [66]:

$$M_{\text{d}} = \frac{1}{N_D} \sum_{j=1}^{N_D} \sum_{i=1}^{N_{\text{SOM}}} h(c(\mathbf{x}_j), i) \cdot \|\mathbf{x}_j - \mathbf{w}_i\|_2 \tag{7.10}$$

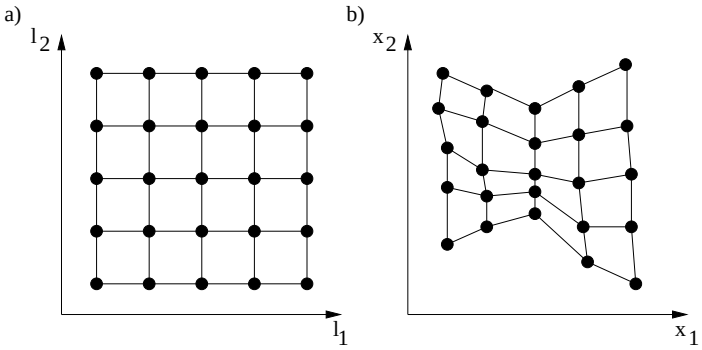

Figure 7.1: (a) SOM with $N_{SOM} = 25$ neurons [circles] and $n_{SOM} = 2$ dimensional quadrilateral lattice [thin lines]. (b) Reference vectors of the SOM plotted in an $n = 2$ dimensional input space.

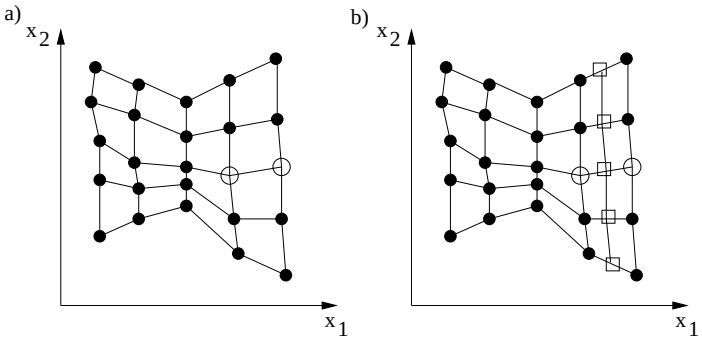

Figure 7.2: Growing SOM for an $n_{SOM} = 2$ dimensional lattice and an $n = 2$ dimensional input space. (a) Among all neurons [filled circles], the neuron with the highest resource value τ [empty circle] and its neighbor with the highest τ among all neighbors [empty circle] are searched. (b) These two neurons specify a lattice direction, in which an additional layer of neurons [empty squares] is inserted.

Using a growing grid already reduces the risk of these defects. A further reduction can be obtained by training several networks and choosing the network with the lowest distortion measure M_{d}.

Using the Self-organizing Map Interpolation as Recombination Operator

We describe a recombination operator that uses a SOM to interpolate the parent population. In each generation of the evolution the SOM is trained on the parent population. The input space of the SOM is equal to the n design variables of the optimization problem plus, if existing, additional strategy parameters such as mutation step sizes. The SOM renders itself easily as a recombination operator, as it defines a lower dimensional interpolation of the parent population. The SOM, when viewed as a recombination operator, chooses randomly a simplex of neighboring neurons in the lattice. A simplex is a geometrical body in an m dimensional lattice space, defined by $m+1$ vertices. The vertices $\mathbf{v}_{k,k=1,\ldots,m+1}$ of the simplex are the reference vectors of the chosen neurons. Within the simplex a uniformly distributed random point x' is generated by computing random numbers until the following equation is fulfilled:

$$\begin{aligned} \mathbf{x}' &= \mathbf{v}_j + \gamma \sum_{k=1,k\neq j}^{m+1} r_k \, (\mathbf{v}_k - \mathbf{v}_j) \,, \\ &\text{with } r_k \sim \text{U(0,1) such that } \sum_{k=1,k\neq j}^{m+1} r_k \leq 1 \end{aligned} \tag{7.11}$$

where $\mathbf{v}_j$ is a randomly chosen vertex of the simplex, $\mathrm{U}(0,1)$ is a uniform distribution of random numbers within the interval $[0,1]$. For $\gamma = 1$, Eq. 7.11 generates points uniformly inside the simplex; $\gamma > 1$ extends the uniform distribution to areas outside the simplex. Fig. 7.3 illustrates the recombination process. The dimension of the lattice m can be chosen according to the dimension of the Pareto front. For a 2-objective problem, the Pareto front consists of a continuous or disrupted line, thus the dimension of the SOM is set to one. Similarly for a 3-objective problem, the Pareto front is a surface and the dimension of the SOM is set to 2.
When training a SOM, there is always the risk of having topological defects as described in Section 7.3.2, which are hard to remove from a trained network. This risk can be significantly reduced by training a second SOM in certain intervals and the one with the lower distortion measure M_d is chosen. Tests showed that an interval of 4 generations is sufficient.
All settings of the SOM-recombination operator are given in Table 7.1.

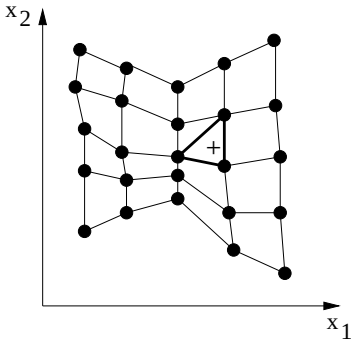

Figure 7.3: Recombination in a SOM for a 2 dimensional lattice and a 2 dimensional input space. A random simplex of neighboring neurons is created [bold line]. Within the simplex a uniformly distributed random point [plus symbol] is generated.

Adapting the Number of Neurons in the Self-Organizing Map

The recombination operator based on SOM implements a mechanism that adapts the number of neurons in the SOM as the optimization converges closer to the Pareto optimal set. The SOM is initialized with a small number of neurons (typically 2 nodes per lattice dimension). In the optimization, the number of neurons is set proportional to the number of nondominated solutions in the parent population, which usually grows as the parent population gets close to the Pareto front and the parents start to spread along the Pareto optimal set. Growing the SOM is required as a response of the map to the growing amount of information in the spreading parent population.
In each generation, the SOM is trained on the parent population with a certain number of epochs e. If the number of neurons N_{SOM} is smaller than a predefined fraction β of the nondominated set in the parent population μ_{nondom}, the network grows as described in Section 7.3.2. After each growing step, the training is repeated until $N_{\text{SOM}} \geq \beta \cdot \mu_{\text{nondom}}$.

7.4 Theoretical Comparison of the Recombination Operators

The introduced recombination operators are now compared theoretically concerning the following items:

Table 7.1: Parameter setting for the SOM recombination operator. Parameters marked with $(^*)$ are taken from Fritzke (1995).

Parameter	Symbol	Value
learning rate*	α	0.1
no. of training epochs*	e	30
initial number of neurons per lattice direction*	$N_{\text{SOM }0}$	2
bell width	σ_{B}	0.6
number of neurons per parent	β	0.3
scaling factor for recombination	γ	3

- *mating restrictions*: Mating restrictions promote the recombination of parents with a small distance to each other. The distance can either be measured in objective space or decision space. On one hand mating restriction may hinder the transfer of information from in the population, as too different parents are not recombined. One the other hand, parents may adapt to the local properties of the Pareto optimal set and mating these parents may generate a large fraction of low quality children [40] such that mating restrictions are advisable.

- *independence on the coordinate system*: Independence is given, if the location of the recombined children is independent to coordinate system rotation.

- *Location of the recombined child*: A recombination operator may generate the children in certain points, lines or within a volume with the position described by a probability distribution.

- *diversity of the offspring compared to the parents*: The recombination operator may create children with an increased or decreased diversity when compared to their parents. As a measure, the difference in size between the smallest hypercube around the children and parents can be compared.

7.4.1 Global Recombination

In global recombination, the recombined child is computed as the mean of all parents. Thus, no mating restriction exists. This type of recombination has been shown to be advantageous for some single objective optimization algorithms [52]

as averaging reduces the magnitude of random variation and thus improves tracking the population movement for the adaptation of the mutation distribution. However, global recombination is disadvantageous for multi-objective optimization. Global recombination removes all variation in the recombined population and hinders the population from spreading along the Pareto front as illustrated in Fig. 7.4. The position of the recombined child is independent to rotation of the coordinate system and as all children are in one point, no preferred recombination direction exists.

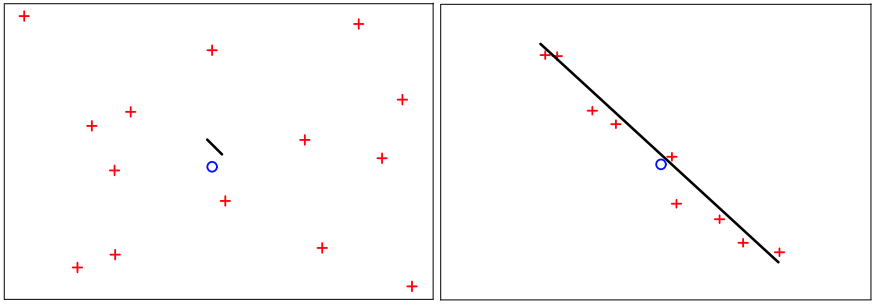

Figure 7.4: Position of the child [circle] generated by global intermediate recombination for parents [plus symbol] far (left) and close (right) to the Pareto front [solid line].

7.4.2 Discrete Recombination

In discrete recombination, always two parents are selected. A child is generated by selecting each decision variable randomly from the first or the second parent. This recombination operator contains no mating restrictions. In case of two decision variables as illustrated in Fig. 7.5, the children of two parents are located in one of four possible positions. If the coordinate system is rotated, the location of the children is different.

In Fig. 7.5, a simple Pareto front with the shape of a straight line is given. Two different parent sets are given. Consider the two parents with the number 3 and 4. The distance of these parents to the Pareto front is much smaller than the distance between the parents. When recombining these parents, it is likely that the children are much farer from the Pareto front then their parents, as the two of the four possible positions for the child are far from the Pareto front. For higher dimensions this

probability increases. Thus, it might be beneficial to introduce mating restrictions, which only recombines neighboring parents.
The children are always generated within the smallest hypercube that comprises all parents as the minimal and maximal decision variable value of the children is within the bounds of the parent values.

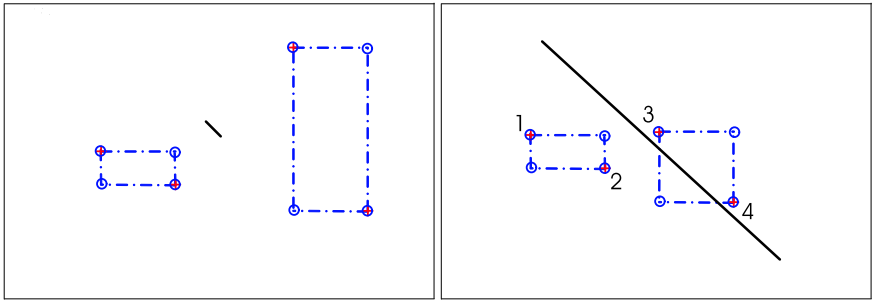

Figure 7.5: All possible positions of a child [circle] generated by uniform recombination of two pairs of parents [plus symbol] far (left) and close (right) to the Pareto front [solid line].

7.4.3 Intermediate Recombination

Intermediate recombination always selects two parents and generates a child by linearly interpolating the parents. As a consequence the decision variable values of the child are always within the bounds of the parent values and thus this interpolation does not promote spreading individuals along the Pareto front. The linear interpolation is invariant to coordinate system transformation.
Interpolating two parents that are close to the Pareto front (see Fig. 7.6, right) is likely to generate a child close to the Pareto optimal set, if the shape of the set is close to linear between the parents. If the Pareto front is curved, the children might be generated in a position far from the Pareto front and mating restrictions might be advantageous.

7.4.4 Simulated Binary Recombination

Simulated binary recombination selects two parents and generates children from a polynomial shaped probability distribution. The highest probability is at the parent

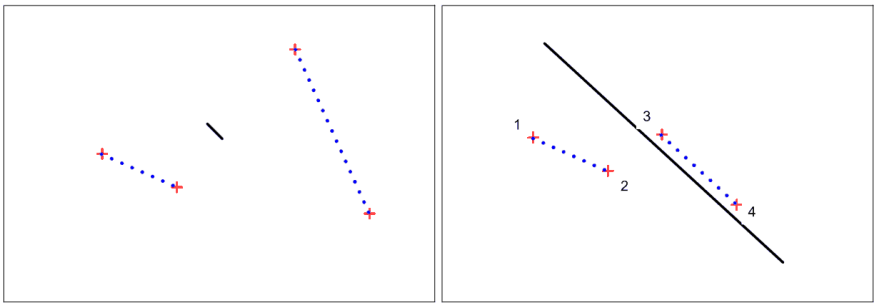

Figure 7.6: All possible positions of a child [dashed line] generated by intermediate recombination of two pairs of parents [plus symbol] far (left) and close (right) to the Pareto front [solid line].

position and decays with increasing distance to the parent as shown in Fig. 7.7. When setting the control parameter η_c for the probability distribution to the typical value of $\eta_c = 20$ [26], the probability decays fast with increasing distance from the parent. In the figure, probability of generating an offspring decays between two contour lines by a factor of 10. The distance between the children can be smaller or larger than the distance between their parents. Thus, the operator allows to spread the population as well as to generate children that interpolate their parents. The operator is constructed such that on average, the distance between the children is equal to the distance of their parents. The probability distribution changes when rotating the coordinate system.

7.4.5 SOM Recombination Operator

The SOM recombination operator trains a SOM on the parent population as shown in Fig. 7.8. The SOM is then used to generate children by constructing simplexes of neighboring neurons in the SOM lattice. A child is obtained by generating a random point within the simplex. The simplex is scaled by a factor larger than 1 in order to generate also points outside the volume covered by the SOM as the SOM always interpolates the parents and never extrapolates them.
In the figure, the lattice of the SOM is one-dimensional and thus a simplex is a line. Children are located within the volume of the decision space covered by all possible simplexes. For the one-dimensional lattice, the volume is equal to a set

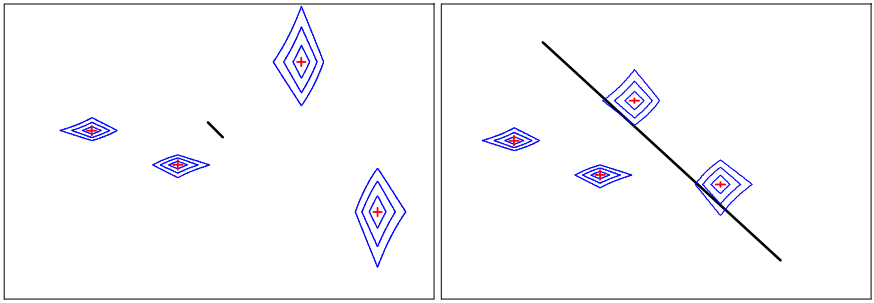

Figure 7.7: Lines of equal probability [thin line] for generating a child by simulated binary recombination of two pairs of parents [plus symbol] far (left) and close (right) to the Pareto front [solid line]. The highest probability is at the parent position and each line shows a decay of the probability by a factor of 10.

of lines; for a two-dimensional lattice, the volume is a set of surfaces. The spread of the children can be larger or smaller than the spread of the parent population. As the training of the SOM is invariant to coordinate system rotation, the SOM recombination operator is invariant, too.

The number of neurons in the SOM is set proportional to the number of nondominated solutions in the parent population. This proportionality is motivated by the two step definition of the Pareto optimization in Section 7.2 and leads from a unrestricted to a restricted mating process.

In Fig. 7.8, the recombination process is illustrated. The left half of the figure illustrates a parent population far from the Pareto front (first step). As the number of nondominated is assumed to be small, the number of neurons is small, too. The distance between neighboring neurons is in the same order of magnitude as the spread of the parent population. Thus, recombining can be considered as unrestricted.

In the right half of Fig. 7.8, the parent population is close to the Pareto front. The value of the decision variables and efficient strategy parameters is assumed to vary along the Pareto optimal set. Compared to the first step, the number of neurons in the SOM lattice grew as the number of nondominated solutions increased. This increase in neurons allows modeling the local variation between the parents and recombination is now restricted.

The neurons of the SOM can be considered as the mean of the surrounding fraction

of parents. A simplex created by neighboring neurons allows sharing information between these neurons. All possible simplexes and thus the possible location of all children are illustrated by the dotted lines. If the parent population is spread around the Pareto optimal set, the dotted lines are quite parallel to the Pareto front. If the Pareto optimal set is locally linear, children are likely to be generated close to the current approximation of the Pareto front.
The major differences of the SOM recombination operator compared to the standard operators are:

- while converging to the Pareto front, the operator adapts from unrestricted mating to restricted mating without specifying mating restrictions such as niche radii explicitly.
- While most recombination operators recombine two parents, the neurons of the SOM interpolate a local subset of parents. Creating a simplex of neighboring neurons supports the transfer of information.
- The SOM interpolation can adapt to straight or curved Pareto optimal sets.
- The scaling parameter γ allows to control the trade-off between a more inter- or extrapolating recombination.

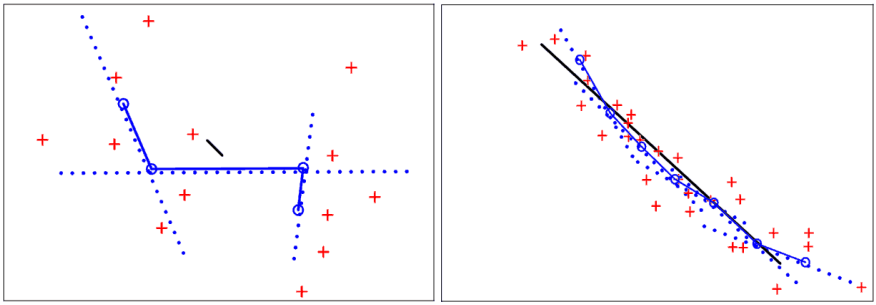

Figure 7.8: All possible positions of a child [dashed lines] generated by SOM recombination for parents [plus symbol] far (left) and close (right) to the Pareto front [solid line]. The neurons of the SOM [circles] and the lattice [solid line] are also given.

7.5 Performance Comparison on Test Problems

Comparing the performance of the different recombination operators requires a set of test problems and performance measures. We consider all 6 test problems, introduced in Section 3.4.
Performing a multi-objective optimization implies usually two goals [24]. The first goal is to obtain solutions of high quality or equivalent solutions that are close to the Pareto front. The second goal is to obtain an approximation of the entire Pareto front for illustrating the possible trade-offs between the objectives.
Here, two performance measures will be considered, which are presented in Section 3.5. The first measure is the *Generational Distance* (D_{DS}) [115], which measures the mean distance of the current parent population to the Pareto front. For each individual in the parent population, the distance to its closest Pareto solution is computed in decision space. This measure is computed at all generations of the evolutionary search and illustrates the progress of the algorithm in increasing the quality of the solutions. No information about the diversity of the individuals is provided. The second measure is the *Generational Approximation* (P) that directly relates to both the quality of the currently evaluated solutions as well as the spread of these solutions along the Pareto optimal set. To a set of uniformly distributed Pareto points the closest of all currently evaluated solutions of the optimization run is identified and the distance is computed. The root mean square of the resulting distances is taken as performance measure P.
Furthermore a mutation and selection operator has to be added. As mutation operator, isotropic, self-adaptive mutation [90, 101] is used. NSGA-II [26] is chosen as selection operator, however, it is extended by the concept of a growing parent population is introduced.
Setting a fixed parent population size is difficult, if we consider the convergence as the two step process, defined in Section 7.2. In the first step, the optimization is similar to a single objective optimization. For single-objective problems with self-adaptive mutation, parent populations of $\mu = 15$ are recommended [7]. In contrast, for approximating the Pareto front in the second step, population sizes of $\mu = 60$ $\mu = 300$ are often used.
In order to solve this conflict, we implement a growing parent population. We set a minimal parent population size of $\mu = 15$ and an upper limit of $\mu = 100$. Inbetween these limits, the number of parents is set equal to to the number of nondominated solutions in the unified set of parents from the previous generation and the current population. This proportionality is chosen as typically the number of nondominated solutions starts to increase when the population is starting to

spread along the Pareto front. Dynamic population sizes have also been analyzed by [111], where the population size was set such that a certain density of solutions along the nondominated front is obtained.
As recombination operator, we consider:

1. no recombination (NOX)
2. global intermediate recombination (GIX)
3. binary intermediate recombination (BIX)
4. uniform recombination (UNX)
5. simulated binary recombination (SBX)
6. mixed recombination (MIX), which recombines each child randomly by either binary intermediate, uniform and no recombination.
7. SOM recombination (SOMX)

Test Problem SPH-1

The SPH-1 problem is the single objective sphere function and is used as representative for problems with fully correlated objectives. For these problems all objectives can be combined into a single figure of merit and the global optimum is the only existing Pareto solution. The problem is also similar to locating a small Pareto front in a large search space and is used to analyze the capability of the self-adaptive mutation in combination with the recombination operators to converge towards the Pareto front over several orders of magnitude. The multi-objective problem difficulty of spreading along the Pareto front is not contained in this problem. The optimization results are given in Fig. 7.1.
Although NSGA-II is designed for multiple objectives, it is also capable of assigning fitness for single objective problems. Then, the algorithm equals the $(\mu + \lambda)$-selection by selecting μ best solutions in an elitist fashion. In the single objective case, the performance measures D_{DS} and P simplify to the distance in decision space of the currently best solution to the optimum.
For all considered recombination operators the logarithmic distance D_{DS} to the optimum decreases linearly over the number of evaluations N. The plot shows a clear difference in convergence speed between the different algorithms. The slowest convergence can be found for NOX and SBX. BIX, UNX, and MIX perform about equally and better than the previous two operators.
The SOMX is implemented with a 1, 2-, and 3-imensional lattice. All 3 lattice dimensions significantly outperform all implementations except GIX. The results show that the lattice dimension of the SOMX is of minor importance for the

sphere function, however it seems that a 3-dimensional lattice is preferable.

Test Problem SPH-2 and SPH-3

The convergence of the different recombination operators on SPH-2 and SPH-3 is similar to the two step process as defined in Section 7.2. The beginning of the optimization (performance measure D_{DS}, P $\gg 1$) is characterized by the problem of locating the Pareto front and the convergence is similar to SPH-1.
However, in the vicinity of the Pareto front (D_{DS}, P ≈ 1), the convergence speed decreases as the population starts spreading along the Pareto optimal set. Two reasons can be found for the decrease. First, the convergence problem becomes more costly, as different fractions of the population converge to different areas along the Pareto optimal set. Second and this is the dominant effect, since the parent population is limited, the convergence of NSGA-II is limited as shown in [17]. The P performance measure reflects how uniform the population spreads along the Pareto front.
For these 2- and 3-objective problems, the lattice dimension of the SOM is set to one and two, respectively. For SPH-2, Fig. 7.9a shows the initial SOM and parent population in the (x_1, x_2) space and objective space. The number of neurons increases as soon as the number the number of nondominated solutions in the parent population rises. Fig. 7.9b contains the final generation. The figure shows that the SOM aligns along the entire Pareto front, and the final parent population is uniformly distributed along the Pareto front in decision space as well as objective space. Since recombination is performed by neighboring neurons in the lattice, recombination becomes more and more local as the lattice grows.
For SPH-3, all parameters of the SOMX are equal to SPH-2, except that the lattice dimension of the SOM is now $n_{\mathrm{SOM}} = 2$. The initial and final population for the SOMX on SPH-3 is given in Fig. 7.10. The figure shows that the SOM approximates the Pareto front well and even though the SOM lattice is quadrilateral and has to fit into the triangular Pareto optimal set in the decision space.
The relative performance of the different recombination operators on SOM-2 and SOM-3 is measured in D_{DS} very similar to SPH-1. However, GIX, which performed best on SPH-1, shows now a worse performance and SOMX converges faster than all other implementations. While D_{DS} measures the mean distance of the population to the Pareto optimal set and thus the convergence, DX measures the spread of the solutions along the Pareto optimal set. For SPH-2, GIX, SBX or NOX clearly performs worse than the other operators. For SPH-3, the different

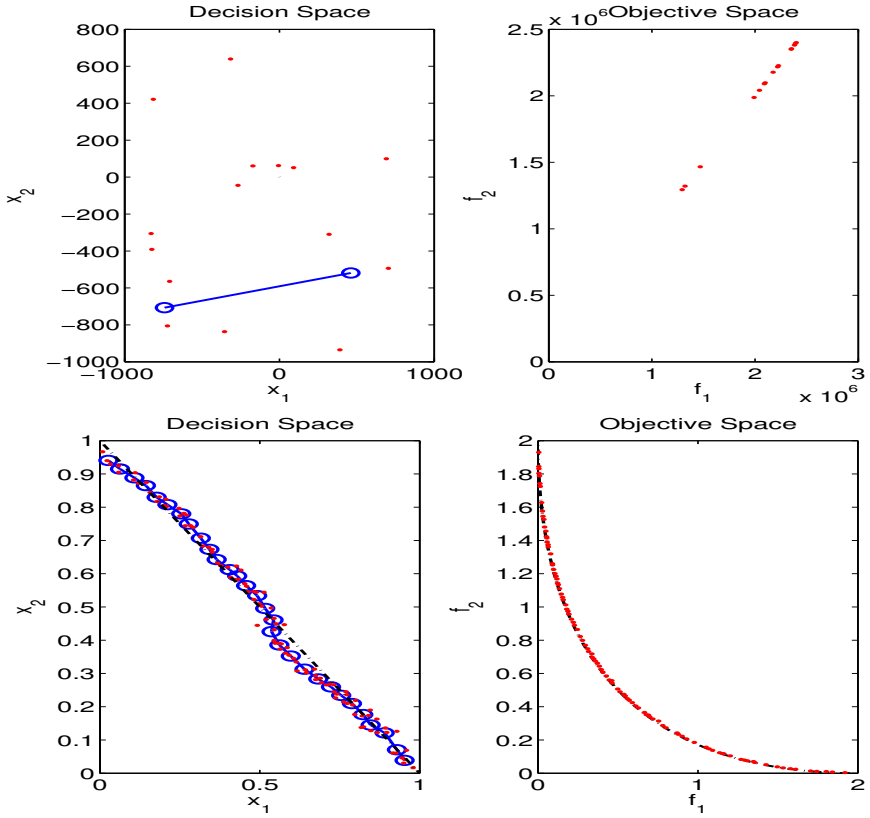

Figure 7.9: Initial (top) and final (bottom) generation of SOMX for the 2-objective sphere problem (SPH-2) with 10 decision variables. The figures contain the SOM [connected circles], the Pareto front [dash-dotted line] and the parent population [dots] in a 2-dim subspace of the decision space and in the objective space. In the initial generation, the neurons of the SOM are randomly placed and in the final generation, the SOM aligns in the decision space with the Pareto front.

recombination operators perform about similar, however, it seems that uniform recombination is preferable, which is used in UNX and MIX. In general the values for D_{DS} and P are higher for SPH-3 than for SPH-2, indicating that approximating

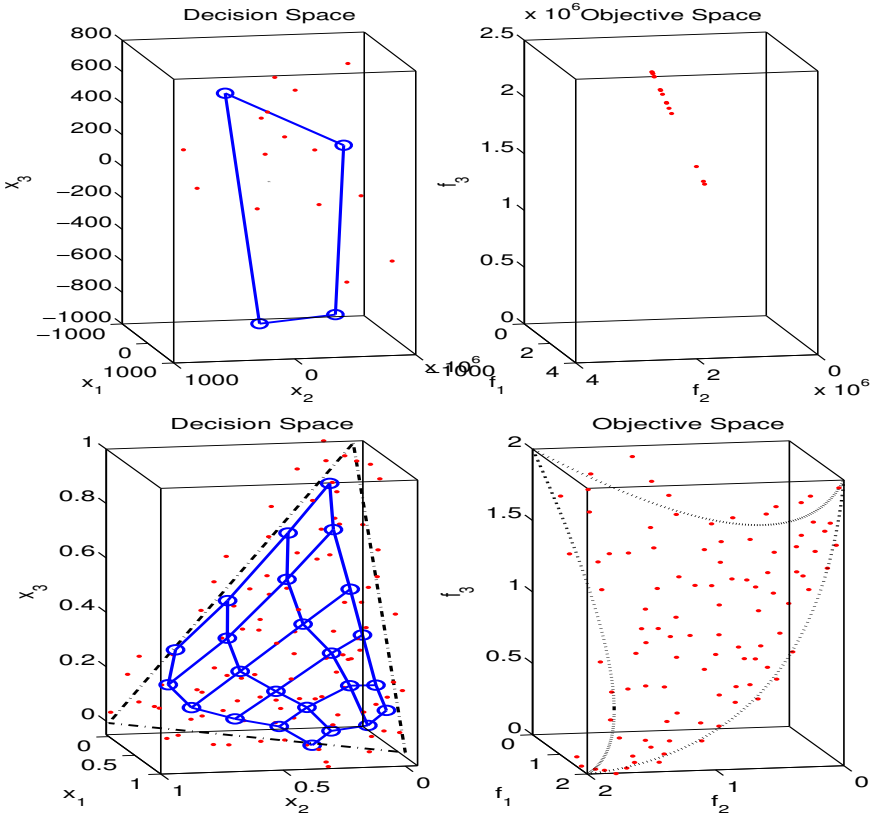

Figure 7.10: Initial (top) and final (bottom) generation of SOMX for the 3-objective sphere problem (SPH-3) with 10 decision variables. The figures contain the SOM [connected circles], the Pareto front [dash-dotted line] and the parent population [dots] in a 3-dim subspace of the decision space and in the objective space. In the initial generation, the neurons of the SOM are randomly placed and in the final generation, the SOM covers the entire Pareto front in the decision space.

the 3-objective Pareto front is more difficult than the 2-objective one. Since the Pareto front of SPH-3 is a surface, the number of nondominated solutions is larger than for the SPH-2 problem.

Test Problem DEB, ZDT1, ZDT2, and ZDT3

For test problems DEB, ZDT1, ZDT2, and ZDT3, the Pareto optimal set is obtained by varying x_1, while setting all other variables to zero. The main difference among the 4 test problems is the shape of the Pareto front, which is convex (DEB, ZDT1), concave (ZDT2), or the density of the Pareto optimal solution is non-uniform (ZDT6).
Figure 7.12 and 7.13 summarize the results on the test problems. On average, UNX and MIX converge closest to the Pareto front as these operators reach on average the smallest value of D_{DS}. Here GIX is not beneficial and converges even worse than applying no recombination (NOX).
The distribution of the solutions along the Pareto optimal set is measured by P. Here, a clearly best operator cannot be found. On average, the SOMX, BIX, and MIX show the best distribution. However, on average NOX, GIX and SBX perform worst. Analysis of the resulting nondominated fronts show problems of UNX. In UNX, the values of the decision variables are just randomly chosen from the parents and thus their value is equal to the values of their parents. This recombination tends to cluster individuals at certain positions along the Pareto optimal set, leading to a very non-uniform approximation. As BIX interpolates the parent decision variables, it avoids this clustering and performs on average better.

Test Problem FF

The optimization difficulty of FF is to approximate the entire Pareto front. The convergence measure P that indicates the spread of the population is more different to minimize than D_{DS}, which indicates the distance of the population to the Pareto optimal set. Furthermore, the difference between the different operators is higher in P than in D_{DS}.
The convergence of all algorithms is plotted in Fig. 7.13. Noticeable are the poor spread of the UNX and the good performance of BIX. This issue has been discussed in Section 7.4. In contrast to all previous test problems, the Pareto optimal set of this test problem is a diagonal line. Walking along this line requires a modification of *all* decision variables. Here, an intermediate recombination (BIX) of two Pareto optimal solutions results again in a Pareto optimal solution, while the uniform recombination (UNX) is very likely to generate a solution far from the Pareto front. Good convergence can also be found for SOMX and MIX.

7.6 Conclusions

Multi-objective optimization algorithms were constructed using self-adaptive mutation, the NSGA-II selection operator and seven different recombination operators. The aim was to analyze the effect of the different recombination operators on the quality and the spread of the resulting set of nondominated solutions. This analysis was performed theoretically and on test problems.
From single objective optimization, four standard recombination operators were analyzed in particular one global (global intermediate recombination) and three binary (simulated binary, uniform, and intermediate recombination) recombination operators. In order to analyze the effect of combining different recombination operators, an operator was constructed that randomly assigns uniform, intermediate or no recombination. Furthermore, the SOM recombination operator that that was especially developed for multi-objective optimization was introduced. All these operators were compared to the case of using no recombination.
Two clear statements about designing of multi-objective recombination operators can be made. First, recombination in general is beneficial as it improved on average the quality and the spread of the resulting nondominated solutions on the test problems. Recombination spreads information (like decision variable values and strategy parameters) among the offspring and thus improves the algorithm. Second, recombination must maintain or even promote diversity in the population as global intermediate recombination, which destroys all diversity, lead to the poorest results.
Among the tree binary recombination operators, simulated binary recombination (SBX) performed clearly worst. Uniform (UNX) and binary intermediate recombination (BIX) performed differently on the test problems. For test problem FF, BIX is preferable as UNX is problematic, if the Pareto optimal set is located on a diagonal line in the decision space as discussed in the theoretical analysis. On test problems DEB, ZDT1, ZDT2, and ZDT6, the nondominated front of UNX was closer to the Pareto front than for BIX. However, the nondominated solutions of BIX showed a wider spread when compared to UNX. We introduced an additional recombination operator (MIX) that selects for each offspring randomly either BIX, UNX and no recombination. This mixed operator seems preferable to using a single operator as it showed less test problem dependence and performed always at least better than BIX or UNX.
Growing Self-Organizing Maps (SOMs) were developed as a recombination operator designed for Pareto optimization. In each generation of the evolutionary optimization, the SOM is trained such as to map the parent population on a lower

dimensional lattice of neurons. The lattice provides an interpolation of the parent population that can be used as recombination operator.
The SOM recombination operator (SOMX) is a first step in designing recombination operators for multi-objective optimization. The key differences to the all other considered operators have been shown: First, the SOM recombination bases on an interpolation of more than two parents. This increases the possible amount of information to recombine. Second, the recombination adapts automatically from unrestricted to restricted mating. Third, the lattice orients along the Pareto optimal set and interpolating neighboring neurons shares information along the Pareto optimal set.
The convergence of an evolutionary algorithm on a multi-objective problem was considered as a two step problem. In the first step, the Pareto front has to be located. Here, unrestricted mating is preferable in order to spread information in the population. In the second step, the algorithm starts to approximate the Pareto front by spreading the population along the Pareto front. The number of nondominated solutions is rising. The number of neurons in the lattice is set proportional to the number of nondominated solutions, in order to store the increasing amount of information in the network. This enables the lattice to adapt to the local properties of parent solutions, as each solution converges to different areas along the Pareto front. Recombination is performed between neighboring neurons in the lattice and for the large network size, recombination can now be considered restricted.
This adaptive strategy shows good performance especially for the SPH-m, DEB, ZDT6, and FF test problems in both spread and quality of the resulting nondominated solutions. When comparing the mean performance over all test problems SOMX, BIX and MIX show the best quality and spread of the resulting nondominated solutions among all 7 considered recombination operators.

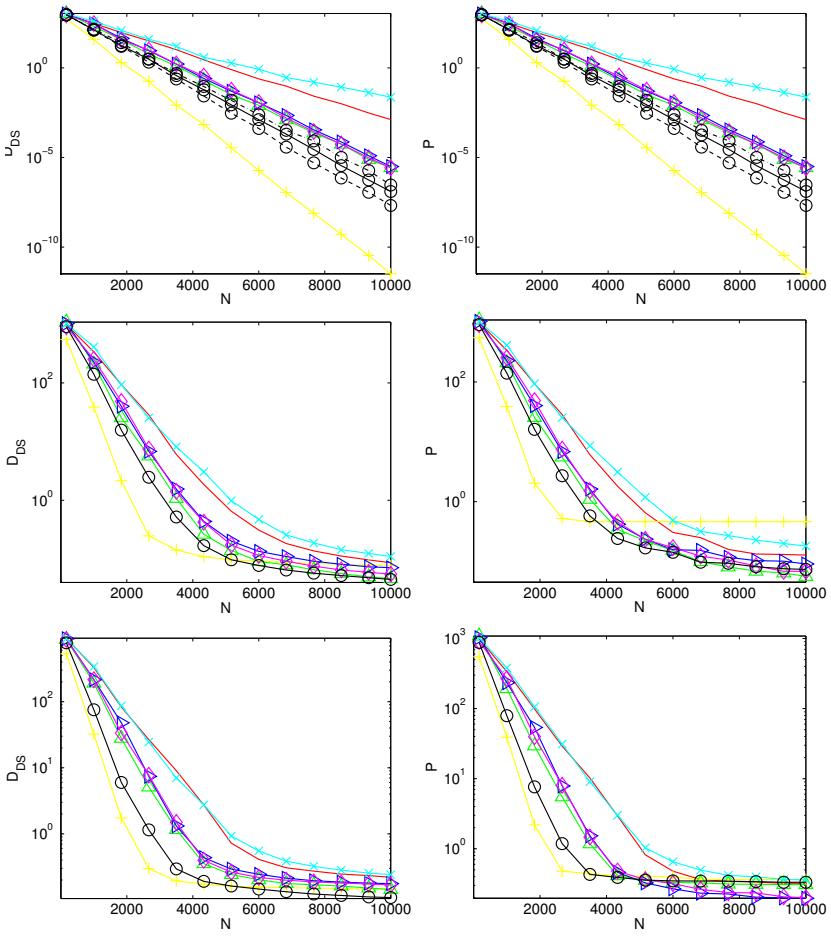

Figure 7.11: Plot of the convergence measure D_{DS} and P over the number of evaluated solutions N for test problem SPH-m with $m = 1$ (top), $m = 2$ (middle) and $m = 3$ (bottom). The different recombination operators are NOX [no symbol], GIX [+], BIX [Δ], UNX [$\triangleright$], MIX [$\Diamond$], SBX [$\times$], and SOMX [o]. To analyze the effect of the SOM lattice dimension, SOMX with a 1- [circle, solid line], 2- [circle, dash-dotted], and 3-dimensional lattice [circle, dashed line] are added.

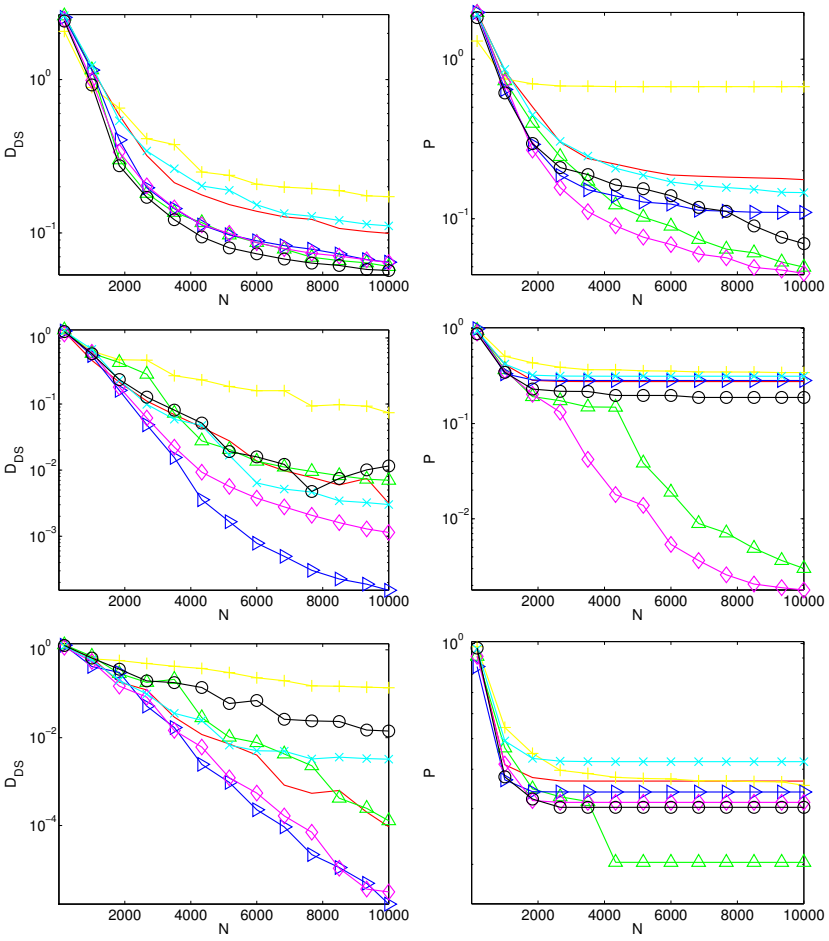

Figure 7.12: Plot of the convergence measure D_{DS} and P over the number of evaluated solutions N for test problem DEB (first row) and ZDT1 (second row), and ZDT2 (third row). The different recombination operators are NOX [no symbol], GIX [+], BIX [Δ], UNX [$\triangleright$], MIX [$\Diamond$], SBX [$\times$], and SOMX [o].

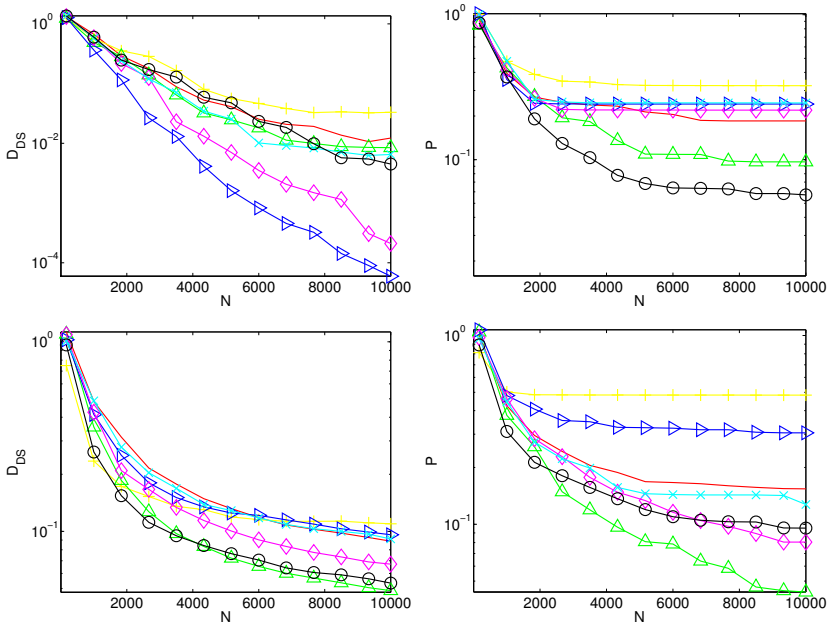

Figure 7.13: Plot of the convergence measure D_{DS} and P over the number of evaluated solutions N for test problem ZDT2 (first row), FF (second row). The different recombination operators are NOX [no symbol], GIX [+], BIX [Δ], UNX [$\triangleright$], MIX [$\Diamond$], SBX [$\times$], and SOMX [o].

Chapter 8

Accelerating Evolutionary Algorithms Using Fitness Function Models

---

A new optimization procedure using empirical models of expensive fitness function is proposed. The procedure bases on the surrogate approach as introduced in Subsection 4.2.2 and uses a Gaussian process as fitness function model. It is denoted as the Gaussian Process Optimization Procedure (GPOP). Implementation issues such as efficient and numerically stable model computation, exploration *vs.* exploitation, local modeling, multiple objectives and constraints, and failed evaluations are addressed. GPOP is compared with CMA-ES on three unimodal and one multimodal test function.

8.1 Selected Approach

In Chapter 4 several optimization procedures with fitness function models are discussed and distinguished into evolution control (Subsection 4.2.1) and surrogate approach (Subsection 4.2.2). In all of these approaches, the goal is to reduce the number of fitness function evaluations of an evolutionary optimization.
In evolution control, the number of fitness function evaluations is reduced by evaluating either a fraction of the population or some generations of the evolutionary algorithm on the model. The fraction of controlled individuals, i.e., individuals that are evaluated on the fitness function is highly dependent on the optimization problem and the efficiency of the optimization algorithm.
We hypothesize:

> The more information from the population is exploited by the evolutionary algorithm (e.g., to adapt the mutation distribution), the higher the fraction of controlled individuals has to be in order to provide sufficient information for the adaptation process.

In other words, evolution control is based on the assumption that the evolutionary algorithm needs very little information, which can be provided by evaluating (controlling) just a fraction of the population. However, this argumentation holds only for those inefficient algorithms that indeed use little information — for "smarter" algorithms such as CMA [52, 51] that pull much more information out of a population, we expect that virtually all individuals must be controlled.
The success of evolution control is thus highly dependent on the fraction of controlled individuals, which is difficult to determine as it depends on both the fitness function complexity and the optimization algorithm. It is always a compromise between avoiding the computational cost of fitness function evaluation and the danger of a poor model misleading the selection operator of the evolutionary algorithm and thus the optimization [59].
In the surrogate approach, a fitness function model is constructed for an initial training set of evaluated points. An optimization algorithm then searches for the optimum of the model's fitness prediction. The predicted optimum constitutes an ideal candidate for an improved point to the problem, and is therefore evaluated on the fitness function. The result of the evaluation is added to the model's training data, facilitating an improved approximation of the fitness function by the model. The procedure then iterates by searching for the optimum of the improved model.

The potential reduction in computational cost is higher for the surrogate approach than for the evolution control, especially once enough data is available to allow for construction of a model that is accurate near the true optimum.
Among the empirical models introduced in Section 4.2.3, Gaussian processes (GPs) appear the most promising to us for fitness function modeling, as they are the only approach to combine the following properties: a GP

- does not require a predefined structure,
- can approximate arbitrary function landscapes including discontinuities and multimodality,
- has meaningful hyperparameters, and includes a theoretical framework for optimizing these hyperparameters,
- provides an uncertainty measure in the form of a standard deviation for the predicted function values.

8.2 Gaussian Process Model

We define a GP using the notation of MacKay [73]: let $f(\mathbf{x})$ be an unknown scalar function and $\mathbf{x} \in \mathbb{R}^n$ a point in an n-dimensional decision space. Eval-

uating f at N data points $\mathbf{X}_N = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ yields the function values $\mathbf{t}_N = \{t_1, t_2, \ldots, t_N\}$, where $(\forall i)\ t_i = f(\mathbf{x}_i)$. Note that subscripts are used here to indicate vector and matrix sizes (for $\mathbf{t}_N$ and $\mathbf{X}_N$) as well as to index elements of those vectors and matrices (e.g., t_i, $\mathbf{x}_i$). The modeling task is to predict the function value t_{N+1} at a new point $\mathbf{x}_{N+1}$. In the following we present the main equations for GPs; for additional details refer to MacKay [73].

The GP imposes a probabilistic model on the given data, namely that the vector of known function values $\mathbf{t}_N$ is one sample of a multivariate Gaussian distribution with joint probability density $p(\mathbf{t}_N|\mathbf{X}_N)$. Similarly, when adding a new point $\mathbf{x}_{N+1}$, the resulting vector of function values $\mathbf{t}_{N+1}$ is assumed to be a sample of the Gaussian joint probability density $p(\mathbf{t}_{N+1}|\mathbf{X}_{N+1}) \equiv p(\mathbf{t}_N, t_{N+1}|\mathbf{X}_N, \mathbf{x}_{N+1})$. Note that the dimensionality of each probability density here equals the number of data points, and is independent of the dimensionality of the decision space n.

Using the rule of conditional probabilities, $p(A|B) = p(A, B)/p(B)$, we can write the probability density for t_{N+1} given the known data points as

$$p(t_{N+1}|\mathbf{X}_{N+1}, \mathbf{t}_N) = \frac{p(\mathbf{t}_{N+1}|\mathbf{X}_{N+1})}{p(\mathbf{t}_N|\mathbf{X}_N)} \tag{8.1}$$

This gives the probability density for the function value t_{N+1} at a new data point $\mathbf{x}_{N+1}$ as a univariate Gaussian, given the N known data points, their associated function values, and the location of the new data point. In the following we transform Eqn. 8.1 so as to express the distribution of t_{N+1} in terms of its mean $\hat{t}_{N+1}$ and standard deviation $\sigma_{t_{N+1}}$. The multivariate Gaussian in the denominator on the right-hand side of Eqn. 8.1 is

$$p(\mathbf{t}_N|\mathbf{X}_N) = \frac{\exp\left(-\frac{1}{2}\mathbf{t}_N^T\mathbf{C}_N^{-1}\mathbf{t}_N\right)}{\sqrt{(2\pi)^N \det(\mathbf{C}_N)}} \tag{8.2}$$

where $\mathbf{C}_N$ is the covariance matrix of the Gaussian distribution, and its mean has been set to zero. Similarly we obtain for $N+1$ data points

$$p(\mathbf{t}_{N+1}|\mathbf{X}_{N+1}) = \frac{\exp\left(-\frac{1}{2}\mathbf{t}_{N+1}^T\mathbf{C}_{N+1}^{-1}\mathbf{t}_{N+1}\right)}{\sqrt{(2\pi)^{N+1} \det(\mathbf{C}_{N+1})}} \tag{8.3}$$

The covariance matrix for the $N+1$ points can be written as

$$\mathbf{C}_{N+1} = \begin{pmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^T & \kappa \end{pmatrix}, \tag{8.4}$$

where $\mathbf{k}$ is a vector containing the covariances between the N known points and the new point, and κ is the variance of the new point. We will determine $\mathbf{k}$ and κ later. Also $\mathbf{C}_{N+1}^{-1}$ can be expressed in terms of $\mathbf{C}_N^{-1}$ [11]:

$$\mathbf{C}_{N+1}^{-1} = \begin{pmatrix} \mathbf{M} & \mathbf{m} \\ \mathbf{m}^T & \mu \end{pmatrix}, \tag{8.5}$$

$$\begin{aligned} \text{where} \quad \mathbf{M} &= \mathbf{C}_N^{-1} + \mu^{-1}\mathbf{m}\mathbf{m}^T, \\ \mathbf{m} &= -\mu\, \mathbf{C}_N^{-1}\mathbf{k}\,, \quad \text{and} \\ \mu &= \left(\kappa - \mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{k}\right)^{-1}. \end{aligned}$$

Inserting Eqns. 8.2 and 8.3 into Eqn. 8.1 gives

$$p(t_{N+1}|\mathbf{X}_{N+1}, \mathbf{t}_N) \propto \exp\left(\frac{1}{2}\left(\mathbf{t}_N^T\mathbf{C}_N^{-1}\mathbf{t}_N - \mathbf{t}_{N+1}^T\mathbf{C}_{N+1}^{-1}\mathbf{t}_{N+1}\right)\right), \tag{8.6}$$

which by the use of Eqn. 8.5 simplifies to

$$p(t_{N+1}|\mathbf{X}_{N+1}, \mathbf{t}_N) \propto \exp\left(-\frac{1}{2}\frac{(t_{N+1} - \hat{t}_{N+1})^2}{\sigma_{t_{N+1}}^2}\right), \tag{8.7}$$

a univariate Gaussian with mean and variance given by

$$\hat{t}_{N+1} = \mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{t}_N\,, \tag{8.8}$$

$$\sigma_{t_{N+1}}^2 = \kappa - \mathbf{k}^T\mathbf{C}_N^{-1}\mathbf{k}\,. \tag{8.9}$$

The covariance matrices $\mathbf{C}_N$ and $\mathbf{C}_{N+1}$ are defined by way of a covariance function C which embodies our prior assumptions about the function to be modeled. Specifically, we define the covariance between function values at two data points $\mathbf{x}_p$ and $\mathbf{x}_q$ to be given by the smooth covariance function [73]

$$C(\mathbf{x}_p, \mathbf{x}_q) = \theta_1 \exp\left(-\frac{1}{2}\sum_{i=1}^{n}\frac{(x_{p,i} - x_{q,i})^2}{r_i^2}\right) + \theta_2 + \delta_{pq}\theta_3\,. \tag{8.10}$$

Here, the first term reflects a distance-dependent correlation between two data points: if their distance is small compared to the length scales r_i, the exponential term is close to one; with increasing distance it exponentially decays to zero.

The hyperparameter θ_1 scales this correlation. In the second term, θ_2 specifies a certain offset of the function values from zero. Finally, the third term adds white noise to the model, scaled by θ_3 and applied only to the diagonal elements of the covariance matrix. The covariance vector $\mathbf{k}$ and variance κ from Eqn. 8.4 can be expressed in terms of the covariance function as

$$k_i = C(\mathbf{x}_i, \mathbf{x}_{N+1}), \quad i = 1, \ldots, N, \tag{8.11}$$

$$\kappa = C(\mathbf{x}_{N+1}, \mathbf{x}_{N+1}) = \theta_1 + \theta_2 + \theta_3 \,. \tag{8.12}$$

8.2.1 Optimizing the Hyperparameters

The GP employs a set of hyperparameters $\theta = \{\theta_1, \theta_2, \theta_3, r_1, r_2, \ldots r_n\}$ which can be set by the user or optimized such that the log-likelihood of the given function values $\mathbf{t}_N$ under the multivariate Gaussian with zero mean and covariance $\mathbf{C}_N = C(\mathbf{X}_N, \theta)$ is maximal. This log-likelihood and its derivative with respect to θ can be expressed as

$$\begin{aligned} \mathcal{L} &= \log\left(p(\mathbf{t}_N|\mathbf{X}_N, \theta)\right) \\ &= -\frac{1}{2}\left(\log\det \mathbf{C}_N + \mathbf{t}_N^T \mathbf{C}_N^{-1} \mathbf{t}_N + N \log 2\pi\right) \end{aligned} \tag{8.13}$$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta} &= \frac{1}{2}\left(\mathbf{t}_N^T \, \Gamma_N \, \mathbf{C}_N^{-1} \mathbf{t}_N \; - \; \text{trace}(\Gamma_N)\right), \\ &\text{where} \quad \Gamma_N \equiv \mathbf{C}_N^{-1} \frac{\partial \mathbf{C}_N}{\partial \theta} \end{aligned} \tag{8.14}$$

Gradient-based or evolutionary algorithms can be used to optimize the hyperparameters, each with their own advantages. Gradient methods are fast local optimizers of smooth functions for which the analytic gradient is available, as is the case here. However, MacKay [73] shows that the hyperparameter optimization landscape is multimodal. This suggests that a slower but more robust global optimizer, such as an evolutionary algorithm, may yield better results.
We therefore use both an evolutionary algorithm (CMA [52]) and a quasi-Newton gradient method (BFGS [28]) as introduced in Section 2.4 and Subsection 1.2.2, respectively, in combination. CMA is always used for the first optimization of the likelihood in order to identify the global minimum. In our experience it suffices to limit CMA to 3 000 likelihood evaluations. To track adjustments to the optimal hyperparameters, e.g., after additional data points have been added, we employ BFGS as a fast optimizer, coupled with a line search by golden section (see, e.g., [101]). The initial step size for each line search is set to 10% of the search space

in order to escape local minima. For BFGS, we found that computing 20 gradients and evaluating 20 points in the line search is sufficient. After optimizing the likelihood 10 times with BFGS, CMA is used again to avoid getting trapped in a local minimum.
To simplify the setting, we normalize the training data such that function values lie within $[0, 1]$ and decision variables within $[-1, 1]$. We then enforce the following bounds on the hyperparameters:

$$\begin{aligned} \theta_1 &\in [10^{-3}, 1] \\ \theta_2 &\in [10^{-3}, 1] \\ \theta_3 &\in [10^{-9}, 10^{-2}] \\ r_i &\in [10^{-2}, 10], \quad i = 1, \ldots, n \end{aligned}$$

Since the ratios of upper to lower bounds are very large, we operate with the log of the hyperparameter values, as proposed by Williams [118]. If the computation of the inverse of $\mathbf{C}_N$ fails too often during optimization, we can increase numerical stability by raising the lower bound on θ_3.

8.2.2 Computational Cost

The key equations of the GP are Eqns. 8.8 and 8.9 for predicting the mean and standard deviation of a new data point, and Eqns. 8.13 and 8.14 for optimizing the hyperparameters. Although all four equations contain the inverse of the covariance matrix $\mathbf{C}_N^{-1}$, the explicit inverse is only needed to compute the gradient of the log-likelihood in Eqn. 8.14, required only for gradient-based optimization algorithms. The other equations contain the product of the inverse with a vector, which amounts to solving a linear system of equations. We can avoid computing the explicit inverse by performing an LU decomposition of $\mathbf{C}_N$:

$$\mathbf{C}_N = \mathbf{L}\,\mathbf{U}, \tag{8.15}$$

where $\mathbf{L}$ and $\mathbf{U}$ are a lower and upper triangular matrix, respectively. The LU decomposition can be computed in order $\mathcal{O}(N^3)$, and is numerically more robust than operating with the explicit inverse. Then, after calculating $\mathbf{C}_N^{-1}\mathbf{t}_N$ via the LU decomposition in $\mathcal{O}(N^2)$, predicting mean and standard deviation for a new point $\mathbf{x}_{N+1}$ are of order $\mathcal{O}(N)$ and $\mathcal{O}(N^2)$, respectively. The log-likelihood for a given LU decomposition and $\mathbf{C}_N^{-1}\mathbf{t}_N$ can also be obtained in $\mathcal{O}(N)$, provided the determinant is computed as proposed in Eqn. 8.16 below.
The gradient of the log-likelihood requires explicit computation of $\mathbf{C}_N^{-1}$, which

is about 3 times as expensive as the LU decomposition. Evolutionary algorithms typically require more evaluations but do not need gradient information, so each evaluation is cheaper. For large N several methods for computing sparse covariance matrices have been proposed (e.g., [105]) which we do not pursue here.

8.2.3 Numerical Difficulties

We encountered numerical problems in computing the log-likelihood according to Eqn. 8.13. Specifically, for $N > 30$ the determinant of the covariance matrix can underflow IEEE 754 double precision floating-point arithmetic, especially if the covariance matrix is ill-conditioned.
Gibbs and MacKay [46] avoid computing the log-likelihood by using only gradient information in their conjugate gradient implementation, including their line search. By contrast, we prefer the likelihood over its gradient since it is computationally cheaper, and allows us to use direct search methods such as evolutionary algorithms. We rewrite the computation of the determinant in the following numerically more stable form:

$$\log\det \mathbf{C}_N \;=\; \sum_{i=1}^{N} (\log \mathbf{L}_{ii} \;+\; \log \mathbf{U}_{ii}) \tag{8.16}$$

In words, we compute the log-determinant as a sum of logs of the elements in the trace of the $\mathbf{L}$ and $\mathbf{U}$ matrices. All quantities in this computation remain within machine precision even where the determinant itself would underflow.

8.3 Gaussian Process Optimization Procedure

We propose an optimization procedure based on the surrogate approach (see Subsection 4.2.2) that uses a Gaussian process as an inexpensive model of the objective function. This Gaussian Process Optimization Procedure (GPOP) starts from an initial set of points, obtained, e.g., from previous optimization runs, by random sampling, or a short run of a conventional optimization algorithm. Our optimization loop then proceeds as follows:
A GP is constructed for the given data set, and the hyperparameters are optimized. An Evolutionary Algorithm (EA) is then used to search for minima of the GP prediction, using several *merit functions* as fitness functions for the EA. Finally, the resulting minima are evaluated on the objective function and added to the data set.

These three steps constitute one iteration of GPOP, and are repeated until a termination criterion is reached.
In the following, we discuss merit functions, describe techniques to limit the training effort of a GP to the local neighborhood of the best solution found so far, and show how real world problems with more than one objective and constraints can be handled.

8.3.1 Merit Functions

Searching the GP for the minimal predicted value $\hat{t}$ *exploits* the knowledge of the GP to find the most promising candidate for reducing the objective function value. However, this carries the risk of premature convergence to non-optimal solutions [113, 34]. To improve the prediction quality of the GP and promote a more thorough global search, there is also a need to *explore* new regions of decision space.
To balance exploration with exploitation, Torczon and Trosset [113] propose a merit function f_{M} which adds a density measure to the predicted function value $\hat{t}$ so as to promote unexplored regions of the decision space. One possible density measure is the maximin distance [62], i.e., the distance to the closest evaluated point. Another possibility is the predicted standard deviation σ_t of the GP, as proposed in [47]. Both measures are minimal at known data points and increase with distance to evaluated solutions. In contrast to the standard deviation, however, the maximin distance is not bounded for unbounded search spaces, and its derivative is discontinuous at all positions equidistant from the two closest evaluated points. We therefore prefer the standard deviation, and define the merit function f_{M} as

$$f_{\mathrm{M}}(\mathbf{x}) \;=\; \hat{t}(\mathbf{x}) \;-\; \alpha\,\sigma_t(\mathbf{x}), \tag{8.17}$$

where $\alpha \geq 0$ balances the two terms by scaling the density measure. (For maximization problems α must be negative.)

We optimize 4 merit functions, using $\alpha = 0, 1, 2, 4$, respectively. Setting $\alpha = 0$ exploits the information of the model by searching for the predicted minimum. By contrast, $\alpha = 4$ strongly pushes the optimization into unexplored regions. The resulting minima can be evaluated in parallel, and updating the GP with several evaluations at a time also serves to reduce the number of GPOP iterations. In our experience, using more than 4 merit functions is not beneficial for the convergence of the optimization.

8.3.2 Local Modeling and Local Search

Both global and local fitness function models are considered in the literature. While global models use all evaluated points, local models only take into account points from a certain region of decision space. Although they thus throw away information, local models have a number of advantages: the precision of any model is limited, and is mainly determined by the difference in the objective function values among the points being modeled. For complicated (e.g., multimodal, discontinuous) function landscapes it may not be feasible to build an accurate global model at all. Finally, since they use less training data, local models typically carry a far smaller computational cost.
Since we want to converge towards the optimum with arbitrary precision, we restrict our model to the local neighborhood of the current best solution $\mathbf{x}^{\mathrm{best}}$. We use as training data for the GP the union of the N_C points closest to $\mathbf{x}^{\mathrm{best}}$ in the decision space, and the N_{R} most recently evaluated points. The closest points serve to model the neighborhood of $\mathbf{x}^{\mathrm{best}}$, while the most recent points are included to allow the data set (and hence the model) to evolve even when the N_C closest points remain the same.
To avoid searching areas that may be poorly modeled, we also restrict the search to a hypercube around the current best solution:

$$\mathbf{x}^{\mathrm{best}} - \mathbf{d}/2 \;\leq\; \mathbf{x} \;\leq\; \mathbf{x}^{\mathrm{best}} + \mathbf{d}/2\,, \tag{8.18}$$

with the hypercube's diagonal $\mathbf{d}$ set to reflect the spread of the N_C closest points:

$$d_i \;=\; \max_c(x_{c,i}) \;-\; \min_c(x_{c,i}), \tag{8.19}$$

where i indexes the dimensions of the search space, and c the N_C closest points. Thus the search space moves around with the current best solution.

8.3.3 Enhancements for Real-World Applications

GPOP was designed to model a single fitness function. For real-world applications this fitness function will typically be composed of several objectives as well as penalties for constraint violations. To model such a complex aggregate of functions with a single GP may prove difficult. We therefore model each objective and penalty by its individual GP, and construct the overall fitness function model as an aggregate of all GPs. While this is computationally somewhat more expensive, it leads to a more accurate model of composite fitness functions.

In real-world applications, the objective or constraint evaluation might fail for some points, due to an infeasible design or inadequacies of the evaluation code. Such failed evaluations should not be used for training the GP [15]. We initially require $N_C/2$ points for training the GP, which we generate with CMA. If some of the evaluations fail, CMA is used to produce an additional $N_C/2$ points at a time, until at least $N_C/2$ points have been evaluated successfully. After this initialization, the iterative optimization procedure is started. In each iteration, the local GP models are constructed for the current data set. Then the minimum for each merit function is located, evaluated on the expensive fitness function, and added to the training data.
If in one iteration the evaluation of all points fails, the GP models remain unchanged. To avoid stagnation in this situation, we add a small Gaussian perturbation $\mathbf{x}^R$ of the current best solution $\mathbf{x}^{\text{best}}$ to the training data:

$$x_i^R \;=\; x_i^{\text{best}} + \; z_i \, d_i/100\,, \quad z_i \sim \mathcal{N}(0,1) \tag{8.20}$$

With these enhancements, our GPOP can be summarized as:

1. **begin**
2. **while** less than $N_C/2$ points evaluated successfully **do**
3. use (2,10)-CMA to generate $N_C/2$ points
4. evaluate them on expensive fitness function
5. **end while**
6. **while** termination criterion not reached **do**
7. find $\mathbf{x}^{\text{best}}$, the best of all points evaluated so far
8. training set = N_C points closest to $\mathbf{x}^{\text{best}}$
9. + N_{R} most recent successful evaluations
10. **for** each objective and constraint:
11. optimize GP hyperparameters (Eqns. 8.13–8.16)
12. **for** each merit function (Eqn. 8.17):
13. find predicted optimum (Eqns. 8.8–8.12)
14. remove optima that have already been evaluated
15. evaluate new optima on expensive fitness function
16. **while** no evaluation successful **do**
17. generate and evaluate perturbation
18. **end while**

19. **end while**
20. **end**

(Eqn. 8.20)

8.4 Performance Analysis

We analyze the performance of GPOP on three unimodal test functions [52] with different properties (a simple quadratic function, Schwefel's function, and Rosenbrock's function) and on a multimodal function (Rastrigin's function):

$$f_{\text{sphere}}(\mathbf{x}) = \sum_{i=1}^{n} x_i^2, \tag{8.21}$$

$$f_{\text{Schwefel}}(\mathbf{x}) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2, \tag{8.22}$$

$$f_{\text{Rosen}}(\mathbf{x}) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right) \tag{8.23}$$

$$f_{\text{Rastrigin}}(\mathbf{x}) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) \right) \tag{8.24}$$

For all functions, the minimal function value is $f = 0$ and the search space is restricted to $x_i \in [-10, 10]$. For the three unimodal functions, we measure the number of function evaluations required to reach a function value smaller than $f = 10^{-10}$. For the multimodal function, we analyze the capability of the algorithm to converge to the global optimum. Thus, we measure the final function value as soon as the algorithm is caught in a local minima. GPOP is assumed to be caught, if the size of the local search area is smaller than $f = 10^{-6}$. Four different training set sizes with: $N_{\text{R}} = N_{\text{C}} = 15, 30, 60, 120$ are analyzed. The results are compared to CMA, an evolutionary algorithm known to perform well on all three unimodal functions [52]. The results represent always the mean of 5 independent optimization runs. Different problem dimensions were analyzed. If for a certain dimension and training set size, no result is given, then this is due to a too poor convergence compared to CMA.

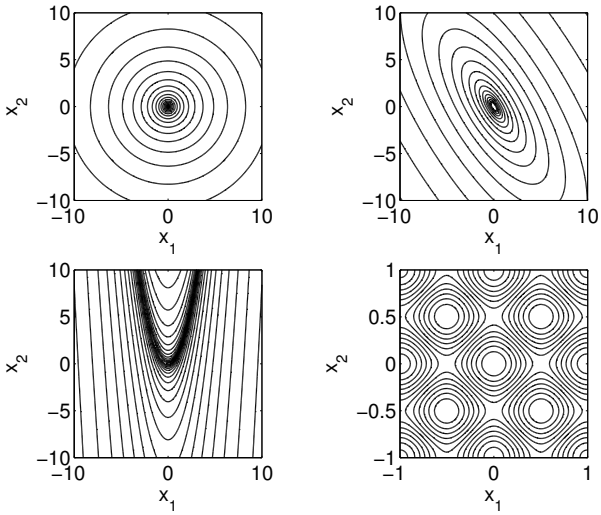

Figure 8.1: Contour plots of the sphere function (upper left), Schwefel's function (upper right), Rosenbrock's function (lower left), and Rastrigin's function (lower right); all in two dimensions.

8.4.1 Sphere function

The minimum of the sphere function is at $\mathbf{x} = 0$. Its contours (Fig. 8.1, left) are hyperspheres in decision space, making this function trivial to optimize. Results for GPOP and CMA are shown in Fig. 8.2, plotted against the problem dimensionality n on a log-log scale.

The number N of function evaluations required increases with n for both algorithms. The increase gets steeper for GPOP past a certain threshold dimensionality, beyond which the training data no longer suffices to model the function well, and the algorithm gets inefficient. As a rule of thumb, this threshold is located at about $n = N_{\mathrm{C}}/2$. In other words, for a sphere function in n dimensions we should have a training set of $N_{\mathrm{C}} \geq 2n$ points, with $N_{\mathrm{R}} = N_{\mathrm{C}}$. As long as this rule is obeyed, GPOP requires about 4 to 5 times fewer function evaluations than CMA.

8.4.2 Schwefel's function

The minimum of Schwefel's function is also at $\mathbf{x} = 0$; its contours are hyperellipsoids (Fig. 8.1, center). Note that the principal axes are not parallel but rotated relative to the coordinate axes of the decision space: Schwefel's function is non-separable, i.e., the decision variables are highly correlated. In contrast to CMA, which was designed to be invariant to such rotations [52], our GP scales the covariance function by a separate hyperparameter along each dimension, and is thus sensitive to them.
Nevertheless, our results for Schwefel's function (Fig. 8.3) indicate that our GP is able to model non-separable functions, and GPOP can optimize them efficiently. The number N of function evaluations required to converge is about the same as for the sphere function, again outperforming CMA by a large factor. For efficient optimization, however, more training data should be used, due to the more complex function topology and the correlation of the decision variables. While the precise threshold for the training set size is less clear here, as a rule of thumb we might require $N_\mathrm{C} \geq 8n$ points, with $N_\mathrm{R} = N_\mathrm{C}$.

8.4.3 Rosenbrock's function

Rosenbrock's function is also non-separable, with highly correlated decision variables. As an added difficulty the minimum is located (at $\mathbf{x} = 1$) in a long, flat-bottomed, curved, and narrow valley (Fig. 8.1, right). Both algorithms consequently require more function evaluations, as shown in Fig. 8.4. The performance gap between GPOP and CMA has narrowed but is still evident, provided that GPOP is given $N_\mathrm{C} \geq 5n$ training data points, with $N_\mathrm{R} = N_\mathrm{C}$.

8.4.4 Rastrigin's function

Rastrigin's function is a superposition of the sphere function and a cosine function with a high oscillation frequency and amplitude. The global minimum is located at (at $\mathbf{x} = 0$). In Fig. 8.5 and 8.6, CMA is compared with GPOP. For Rastrigin's function, we are interested in the global convergence and not the convergence speed. Thus, each optimization runs until the algorithm converged to a local or the global minimum. An algorithm is considered converged as soon as the step size (CMA) or the local search area (GPOP) is 10^{-4} times smaller than the size of the surrounding valley, which is ≈ 1. In the figures, the final function value f_f of the optimization is plotted. The square root of f_f is equal to the distance of the final

point to the optimum.
In Fig. 8.5, the same settings for GPOP are used as for the three unimodal functions. The figure shows that GPOP and CMA are not able to determine the global minima for $n > 2$. This is due to the large number of local minima and the high oscillation amplitude of the function between the minima. For the standard settings, CMA performs better then GPOP. However, with increasing training set for GPOP and problem dimension the difference between CMA and GPOP decreases. There are several possibilities to improve the performance of GPOP on multimodal functions:

- larger values of α in the merit function would promote better exploration of the search space.
- multiple local search spaces (e.g., around the best N_{best} solutions) would facilitate recovering multiple minima.
- increasing the noise level θ_3 of the GP model leads to a smoother approximation of the fitness function; this may remove the local minima from the GP model.

In Fig. 8.6, the third possibility was chosen and the noise level was fixed at $\theta_3 = 0.01$. With this setting and training set size of 60 to 120, GPOP performs similar to CMA.

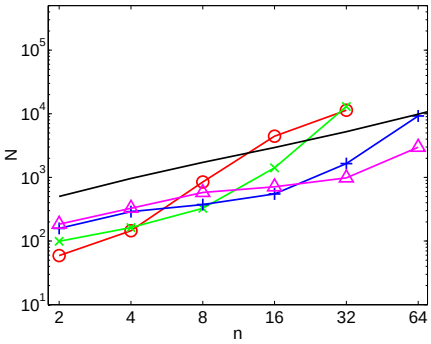

Figure 8.2: Mean number N of function evaluations against problem dimensionality n to reach $f_{\mathrm{sphere}} = 10^{-10}$ for GPOP with training set size $N_{\mathrm{R}} = N_{\mathrm{C}}$ = 15 (o), 30 (x), 60 (+), and 120 (Δ), compared to CMA (unmarked).

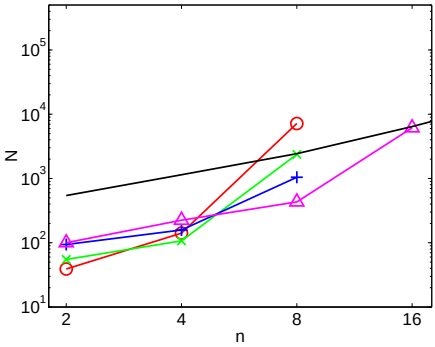

Figure 8.3: Mean number N of function evaluations against problem dimensionality n to reach $f_{\mathrm{Schwefel}} = 10^{-10}$ for GPOP with training set size $N_{\mathrm{R}} = N_{\mathrm{C}}$ = 15 (o), 30 (x), 60 (+), and 120 (Δ), compared to CMA (unmarked).

8.4.5 Summary of the Performance Analysis

GPOP has been shown to be capable of optimizing difficult test problems. Furthermore GPOP has proven to be an efficient optimization procedure as it requires less function evaluations than CMA to find the minimum of three unimodal functions within given precision.

The performance of GPOP is dependent on the training set size as sufficient training data is required for adequately modeling the fitness function. For all considered test problems, a rule of thumb for setting the training set size is given as a function of the problem dimensionality. Following this rule, GPOP requires on average about 3 to 6 times less function evaluations than CMA.

Compared to CMA, the main drawback of GPOP is the higher computational cost of the optimization procedure. The cost scales $\mathcal{O}(N^3)$ with the training set size and is only $\mathcal{O}(N)$ with the problem dimension. Thus, GPOP should mainly be applied to expensive optimization problems, requiring at least several seconds of CPU time per function evaluation. The computational expense is also the reason why the training set size was limited to 120. With this training set size GPOP is more efficient then CMA on problems with a problem dimension of up to 16-64 decision variables.

For the multimodal Rastrigin's function, CMA performs better than GPOP when comparing the final function values. However, several ways to improve the perfor-

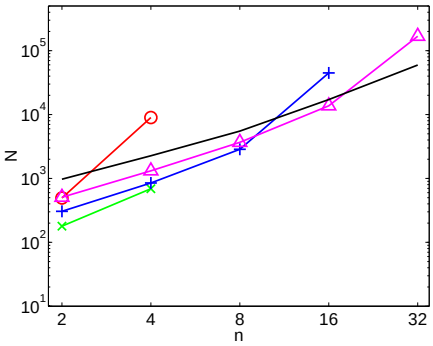

Figure 8.4: Mean number N of function evaluations against problem dimensionality n to reach $f_{\mathrm{Rosen}} = 10^{-10}$ for GPOP with training set size $N_{\mathrm{R}} = N_{\mathrm{C}}$ = 15 (o), 30 (x), 60 (+), and 120 (Δ), compared to CMA (unmarked).

mance of GPOP exist, and by adding white noise with variance $\theta_3 = 0.01$ to the GP model we have obtained performance similar to that of CMA.

8.5 Conclusions

When optimizing expensive fitness functions, it is important to reduce the number of function evaluations required as much as possible. This can be achieved by exploiting knowledge of past evaluated points to train an empirical model that can be used as an inexpensive surrogate of the fitness function.

We have shown how Gaussian processes (GPs) can be used as fitness function surrogates. The resulting Gaussian Process Optimization Procedure (GPOP) is capable of efficiently optimizing even very ill-conditioned problems such as Rosenbrock's function. Here GPOP clearly outperformed CMA, an algorithm known to be efficient in such functions [52].

A scaling analysis over the number of decision variables showed that sufficient training data points must be supplied for GPOP to converge efficiently. The computational expense of the optimization algorithm itself is consequently much higher for GPOP than for CMA. While CMA can be efficiently applied to problems with more than 300 decision variables [52], we encountered computational limits for our test functions as early as 16 to 64 variables. These could be ad-

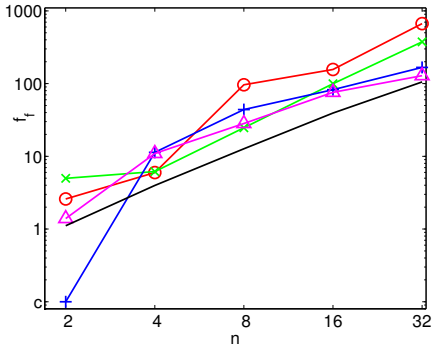

Figure 8.5: Final function value f_f of the converged optimization run on $f_{\mathrm{Rastrigin}}$ against problem dimensionality n for GPOP with training set size $N_{\mathrm{R}} = N_{\mathrm{C}}$ = 15 (o), 30 (x), 60 (+), and 120 (Δ), compared to CMA (unmarked).

dressed by a sparse GP implementation [105].
GPOP also showed global search capabilities: adding some white noise to the GP model allowed GPOP to perform similar to CMA. GPOP's 4 merit functions constitute different compromises between exploration and exploitation, thus balancing the inherent conflict between converging to and escaping from local minima.

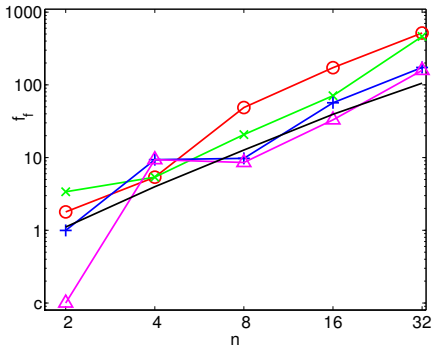

Figure 8.6: Final function value f_f of the converged optimization run on $f_{\mathrm{Rastrigin}}$ against problem dimensionality n for GPOP with training set size $N_{\mathrm{R}} = N_{\mathrm{C}}$ = 15 (o), 30 (x), 60 (+), and 120 (Δ), and a fixed hyperparameter value $\theta_3 = 0.01$, compared to CMA (unmarked).

Part III

Turbomachinery Design Optimization

Chapter 9

Gas Turbines for Energy Generation

Gas turbines are successfully implemented for power generation and for vehicle propulsion. In aircraft propulsion or drives for vehicles, gas turbines are chosen due to their large power-to-weight and power-to-volume ratio. Furthermore, for certain operating conditions the cycle efficiency of gas turbines is high compared to piston engines.

In the field of energy generation, gas turbines have often been chosen in the past when fast start and shut down on demand is required. This is especially needed for compensating peak loads over the daytime. In contrast, steam cycles as used for coal and oil firing or nuclear power are base-load machines since the start and shut down is tremendously longer due to the large heat capacity in the cycle.

In recent years the share of gas turbines among new power plants has significantly increased due to a number of market changes. Since the price of natural gas has dropped compared to oil and the efficiency of gas turbines increased, gas turbines are used for base load as well. In a combined cycle, the exhaust gas of gas turbines can be used in recovery boilers to raise steam for a steam turbine, increasing the overall efficiency. The efficiency of today's base load gas turbines ranges between 34% to 38%. Integrating the machines into a combined cycle increases the efficiency to about 49% to 58%. In addition, moderate installation cost and spatial dimensions of gas turbines compared to steam cycles reduces the overall energy production costs.

Figure 9.1 shows a state-of-the-art mid-size gas turbine. Supporting systems like the secondary air systems are not show. In the figure, the gas flow through the machine is from left to right. Ambient air enters the compressor first. Then the compressed air is burned in the combustion chamber and finally expanded in the turbine. The difference in power between the turbine output and the compressor input is the net power to generate electricity.

In today's machines, axial compressors generate pressure ratios between 1:15 and 1:30 and consist of about 20 to 30 stages. Each stage comprises a stator and rotor

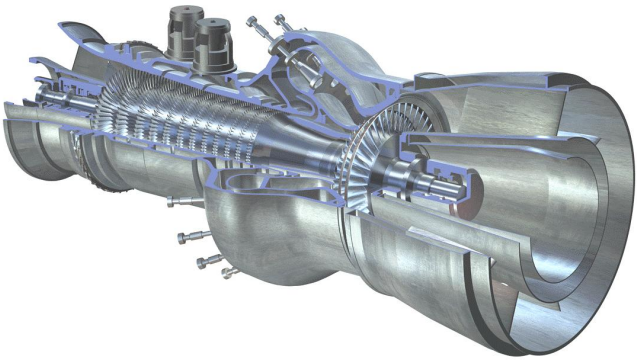

Figure 9.1: Cut-away view of a gas turbine (the picture was downloaded in 2000 from the picture gallery on the homepage of ABB-Alstom Power).

row. For the gas turbine in Fig. 9.1, the combustion chamber is annular around the turbine axis with a set of burners aligned in the annulus. The combustion products leave the combustor at temperatures of about $1200 - 1400°$ C. The design of the turbine differs from the compressor. Since gaseous flow can be more easily expanded than compressed, only about 4 turbine stages are needed to expand the hot gas. Due to the high temperatures of the gas, the combustion chamber and the turbine rows need to be cooled. Most turbines are air-cooled, where the cooling air is bleed air taken at different positions of the compressor. The cooling design is of core importance for the turbine, since for highly efficient engines, the amount of cooling air needs to be reduced at minimum.

Chapter 10

Multi-objective Optimization of Noisy Combustion Processes

We consider the multi-objective optimization of the combustion processes for a single burner of a stationary gas turbine. The burner combusts fuel by a vortex-stabilized lean premixed flame. The burner is analyzed by an atmospheric test-rig. In the test-rig, the combustion can be passively controlled by a set of valves that control the fuel flow rates though different fuel injection holes along the burner axis. Two different setups are considered using either 8 proportional valves to adjust the fuel flow at each injection hole or 16 digital valves which just open or close certain injection holes.
NT-SPEA of Subsection 6.2.5 is applied to the Pareto optimization of the combustion process. The optimization results in an approximation of the Pareto front for minimizing NO_x emissions and reducing the pressure fluctuations (pulsation) of the flame. Both objectives are conflicting and affect the environment and the lifetime of the gas turbine, respectively. The physical implications of different solutions are discussed.

10.1 Introduction

A central component in the design of a gas turbine is the design of the burners in the combustion chamber as the burners are mainly responsible for the emissions of the machine and have a major impact on the thermodynamic inlet conditions of the turbine. The burners mix air and fuel and combust them continuously. This is different to piston engines, which combust in a cyclic manner.
The design of a burner follows various objectives. The burner determines the position of the flame in the combustion chamber. The flame position should be well controlled, avoiding direct contact and thus damage on walls of the combustion chamber. Also, a uniform mixing of air and fuel is desired. Mixing is respon-

sible for the emissions and the pressure pulsations of the combustion flame. For example, the presence of areas of rich combustion results in locally increased temperatures and NO_x emissions. Local temperature peaks in the exhaust gas of the burner may damage the proximate turbine blades. Furthermore, the burner should produce a stable combustion flame, avoiding undesired pressure pulsations. Pulsations are thermo-acoustic waves, which occur in particular for very lean combustion when operating under part load condition. They can reduce the lifetime of the turbine, e.g., by fatigue and by local overheating the blades surface.
Reinke *et al.* [91] distinguish the currently available combustion types into swirl-stabilized burners, diffusion burners, lean premixed burners, and catalytic burners. The considered burner is the Alstom EV burner [96]. The burner mixes air and fuel by swirl and stabilizes the flame position by a vortex breakdown. For part load, a diffusion pilot flame can be used to stabilize the combustion.
Various investigations have been made in order to reduce pulsations and emissions of the burner by active and passive control mechanisms. For example, Paschereit *et al.* [85] reduced the pulsations in their experimental test-rig by an acoustic actuation in a closed control loop. More often, passive control mechanisms are used since the implementation of a feedback loop for active control is a complex and expensive modification.
For complex problems such as combustion processes, numerical simulations are not widely used as a predictive tool due to the complexity of the physical phenomena under investigation. Although intensive research efforts are underway on this front, experimental setups are widely used for the study and optimization of combustion processes. Optimization of combustion processes is especially resource intensive since a large number of different designs has to be evaluated. At the current state, evaluating the designs in an experimental setup is still less time intensive than performing three dimensional numerical simulations.
In this chapter, we optimize the combustion processes of a single burner in an atmospheric test-rig. We optimize a passive control mechanism, choosing the fuel flow rates through the injection holes of the burner as decision variables of the setup. The objectives are to reduce the NO_x emissions and pressure pulsation of the burner. Since the two objectives are conflicting, the optimization results in a Pareto front corresponding to reduced emissions and pulsation of the burner. Experimental setups present a number of challenges to any optimization technique including: availability of point wise information only, experimental noise in the measurements, uncontrolled changing of environmental conditions and measurement failure. The various challenges require a robust multi-objective optimization technique, thus NT-SPEA (Subsection 6.2.5) is chosen. In Chapter 6, the algorithm

was tested on problems with similar noise levels as occur in the experimental test-rig and showed the best convergence of all considered algorithms.

10.2 Atmospheric Test-rig for Gas Turbine Burners

The test-rig for a single burner under atmospheric pressure condition is illustrated in Fig. 10.1. Preheated air enters the test-rig from a plenum chamber. The air flows into the conical burner through two inlets as illustrated in Fig. 10.2. Along the inlets, fuel is injected and mixes with the air due to the difference in velocity. The mixing is enhanced by the swirl in the burner that occurs due to its conical shape. A controlled vortex breakdown is caused by the difference in the cross-section between burner and combustion chamber. The flow recirculates around the combustion zone. Recirculation stabilizes the combustion flame in a predefined combustion area. The fuel is natural gas or oil and is injected through injection holes, which are uniformly distributed along the burner. A more detailed description is given by Sattelmayer *et al.* [96].
Gas analysis equipment and a microphone are used to measure all emissions and the pressure pulsations of the burner. Constant operating conditions are obtained by monitoring the airflow from the plenum chamber, the total fuel flow and the exhaust gas of the burner. The NO_x emissions and the pulsation of the burner are the two objectives to be minimized in a Pareto optimization setup. Pulsations are thermo-acoustic combustion instabilities, involving feedback cycles between pressure, velocity, and heat release fluctuations. The microphone measurements of the pulsation need to be time-averaged over several seconds. NO_x emissions are exponentially dependent on the combustion temperature and occur especially in centers of rich combustion resulting from inhomogeneous mixing of fuel and air.
Figs. 10.1 and 10.2 show the valves that allow to control the fuel flow distribution along the burner axis. The fuel injection is controlled with two different setups by either 8 proportional valves to individually adjust the fuel flow for the different fuel injectors or by 16 digital valves, which include or exclude fuel injectors along the distribution holes. Measuring one valve setting with either digital or proportional valves requires about 1 minute. This includes the time required for changing the valves, for the combustion process to adjust to the changed settings, and for the measuring.

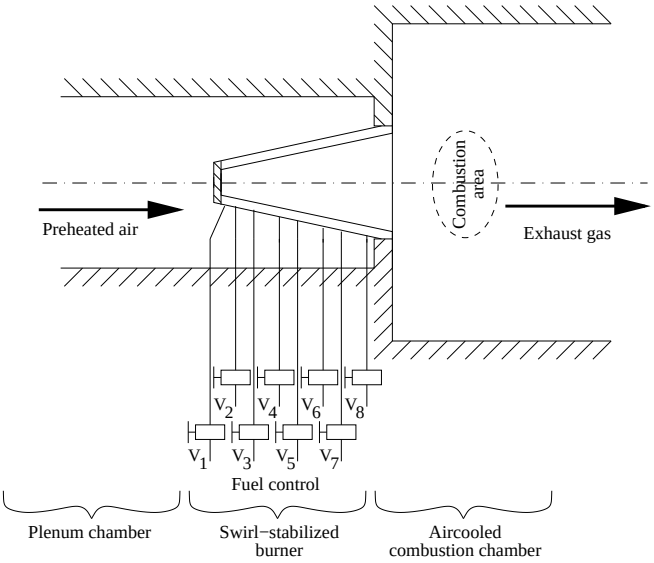

Figure 10.1: Sketch of the atmospheric combustion test-rig with a low-emission swirl stabilized burner. The fuel flows through the injection holes are to be controlled. The NO_x emissions and the pulsation of the burner are the objectives to be minimized.

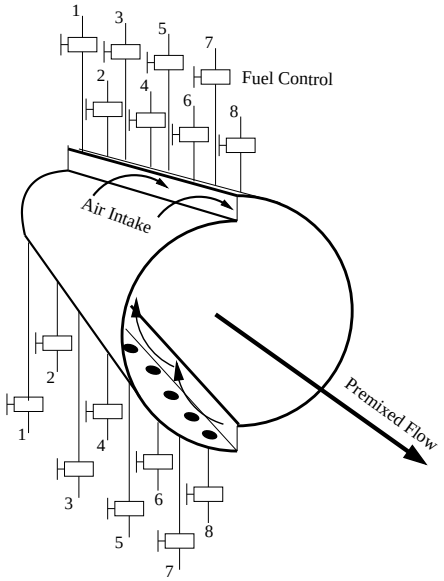

Figure 10.2: Sketch of the EV burner. The burner consists of two half-cones with a certain offset, such that between the two half-cones two intakes emerge. Air enters the burner through the 2 intakes and gets swirled by the shape of the cone. The fuel is injected along the two slots and mixed with the air by the difference in velocity and swirl. The fuel flow rate through the injection holes are to be controlled.

10.3 Burner Optimization with Proportional Valves

10.3.1 Valve Encoding and Optimization Algorithm

Eight proportional valves $V_{i,i=1,\ldots,8}$ are used to control the fuel flow rates. Each valve V_i controls the mass flow $\dot{m}_i$ through a set of adjacent injection holes along the burner axis.
In order to keep the operating conditions constant, the total fuel mass flow $\dot{m}_t = \sum_{i=1}^{8} \dot{m}_i$ is fixed, reducing the number of free decision variables for the optimization from 8 to 7. Fig. 10.3 shows the implemented encoding for the 8 valves V_i by 7 virtual valves $V'_{j,\ j=1,\ldots,7}$. The total mass flow is split by a first virtual valve V'_1 into two flows, with each of the flows feeding either the first or second half of the real valves. The next layer consists of two virtual valves V'_2 and V'_3 and splits the two flows into four. Finally, the virtual valves V'_4, V'_5, V'_6, and V'_7 feed the real valves V_i and determine the fuel flows $\dot{m}_i$. While the evolutionary algorithm operates with the seven virtual valves, the real valves are used in the test-rig. A detailed description about the experimental setup and the fuel control can be found in [30].
The optimization algorithm is NT-SPEA with a population and archive size of 15. The standard recombination and mutation operators of Section 3.3 are used, which were also applied to the test problems in Chapter 6. For the mutation operator, the standard deviation σ is set to of 10% of the interval size in which the variable is defined, and a mutation probability p_{M} of 20%. The standard deviation is set relatively large compared to the interval size, since smaller mutations might not be observable in the presence of the experimental noise. In the following, we refer to the real valves V_i and the real fuel flows $\dot{m}_i$.

10.3.2 Optimization Results

An optimization run is started evaluating a total of 326 different burner settings within one working day. All solutions are plotted in Fig. 10.4 in order to show the possible decrease in NO_x emissions and pulsations by the optimization compared to the given standard burner configuration and between the best and worst designs. The given standard burner configuration is marked in the figure and represents a setting with equal mass flow through all valves. Some solutions found by the optimization process dominate the standard configuration, i.e., are superior in both objectives. Thus the optimization run is successful, delivering improved solutions

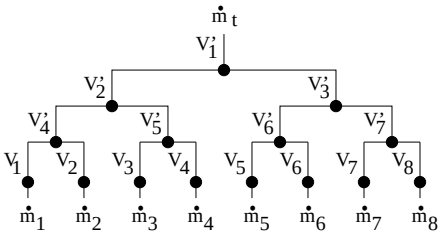

Figure 10.3: Encoding of the fuel flow $\dot{m}_i$ through the 8 valves V_i of the test-rig. Since the total mass flow $\dot{m}_t$ is fixed, the 8 fuel flows can be encoded by 7 virtual valves V'_j.

for both objectives. The occurrence of a wide nondominated front underlines the conflict in minimizing both objectives and just (Pareto) compromise solutions can be found.

In the figure, the objectives are noisy. Thus, drawing just the nondominated front and picking one solution from the front is risky from the point of view, that an inferior solution is picked, which is nondominated due to the noise in its objective values. Picking an area close to the nondominated front increases the confidence in the front, especially if the valve settings are quite similar for the solutions in the area. A second reason for not drawing just the nondominated front is the possible shift of the front towards smaller objective values. The objectives contain noise and the selected nondominated solutions may improve due to noise leading to smaller objective values. In addition we are more interested in the valve settings than in the exact objective values, since the valve settings indicate the physics of the problem.

Five areas along the nondominated front are picked and marked by boxes. For the solutions within the boxes, the valve settings are printed in Fig. 10.5. Fig. 10.1 shows the arrangement of the valves in the combustor. For better illustration, the settings are connected with a line and the dash-dotted line shows the standard burner configuration with equal mass flow through all valves. Within each box, the settings of the different solutions are indeed quite similar.

Boxes 1 and 5 are at the extreme ends of the Pareto front. Box 1 represents Pareto solutions with high NO_x emissions, but low pulsation. The corresponding valve settings show an increased fuel mass flow at valves 1, 2 and 4, while the flow at valves 5 and 6 is reduced. The fundamental mechanism corresponding to these settings is the fact that the increased mass flow through valves 1 and 2 leads to rich

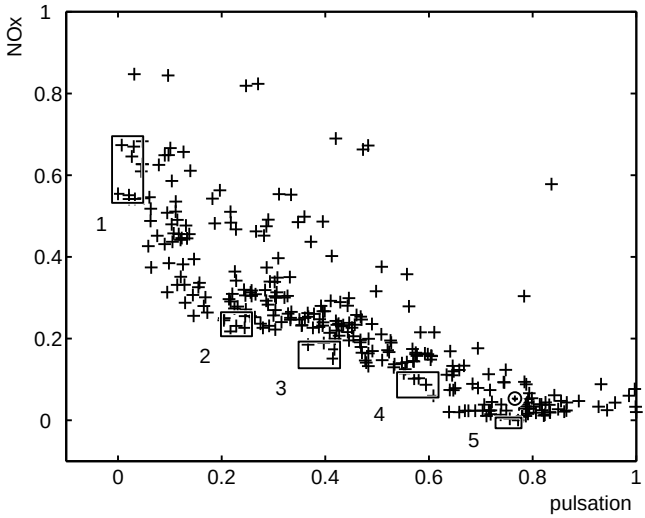

Figure 10.4: Proportional valves: All measured solutions of the burner optimization run [plus symbol] and given standard burner configuration [circular symbol]. 5 boxes mark different areas along the nondominated front.

combustion in the center of the burner. The rich combustion zone stabilizes the combustion like a pilot flame, but increases the NO_x emissions. The lean zones are close to the middle of the burner at valves 5 and 6.
Box 5 contains solutions with minimal NO_x emissions, but high pulsation. The mass flow through each valve is about equal, generating no rich combustion zones. Compared to the standard burner configuration, the small mass flow increases at valves 5 and 8 and decreases at 3 and 4 leads to lower NO_x emissions, while the pulsation is unchanged.

10.3.3 Statistical Analysis

One of the interesting features of the resulting nondominated front is the almost linear change in valve settings along the front. At Box 1, five valves have either strongly increased or decreased mass flow and their amplitude is constantly de-

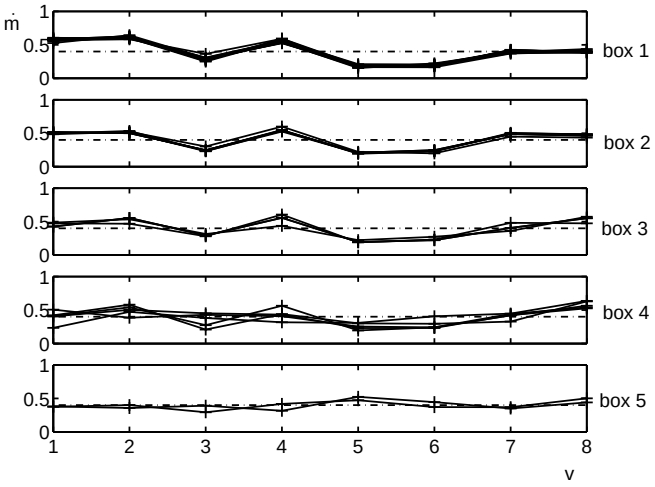

Figure 10.5: Proportional valves: Mass flow $\dot{m}$ through the valves $V_{i,i=1,\ldots,8}$ for solutions along the nondominated front, marked by the 5 boxes of Fig. 10.4.

creasing from Box 1 to 5 until it reaches an almost equal mass flow for all valves in Box 5. This indicates simple dependencies of the valves with the objective functions. Fig. 10.6 contains a scatterplot for the valve settings and objective functions of all measured solutions. A scatterplot contains all possible 2D subspace plots for all decision variables and objectives. The plot in column 9 and row 10 contains the objective space with the nondominated front. Most interesting are the two last rows, containing the correlation of the valves with the objective functions. For example, the horizontal and vertical axis of the plot in row 9, column 1 represent valve 1 and the NO_x emission, respectively. Strong correlation is expressed by narrow stripes under $\pm 45^\circ$ to the axis. An axially symmetrical area of solutions implies no correlation. Strong correlation can be observed between valves 1, 2, 5, 6 and the two-objective functions.
The correlation coefficients $r_{V_i,\,\mathrm{NO}_x}$ and $r_{V_i,\,pulsation}$ for the decision variables and objectives are given in Fig. 10.7. They complement the results from the scatterplot. For all valves, the correlation coefficients have opposite signs for the two objectives. Therefore, changing the fuel injection in any of the valves improves one objective while the other is worsened. Large coefficients indicate a strong correlation and occur between valves 1, 2, 5, 6 and the two objective functions. For increasing the mass flow through valves 1 and 2, the emissions increase while the pulsation decreases. For valves 5 and 6, this is vice versa.
It has to be considered that these observations hold for the solutions obtained through an optimization process. The distribution of the solutions in the scatterplot in Fig. 10.6 illustrates that they do not cover the whole decision space as some areas are not covered with solutions. Hence, these solutions are not uniformly distributed in the decision space and may not be representative.

10.3.4 Noise Analysis

The NT-SPEA algorithm that is used for the burner optimization contains the special feature of re-evaluating solutions after their lifetime expires. Among the 326 evaluated solutions, 40 were re-evaluated at least once by the optimizer. Comparing the difference in NO_x between a solution and the re-evaluated one, the maximal difference is about 8% of the objective range and the mean difference is 2%. For the pulsation, the maximal and mean difference is 13% and 4%, respectively. Thus, the noise in the pulsation is more critical to the optimization. The large ratio between the maximal and mean difference indicate the rare occurrence of outliers and the presence of noise in the objective measurement of all solutions.

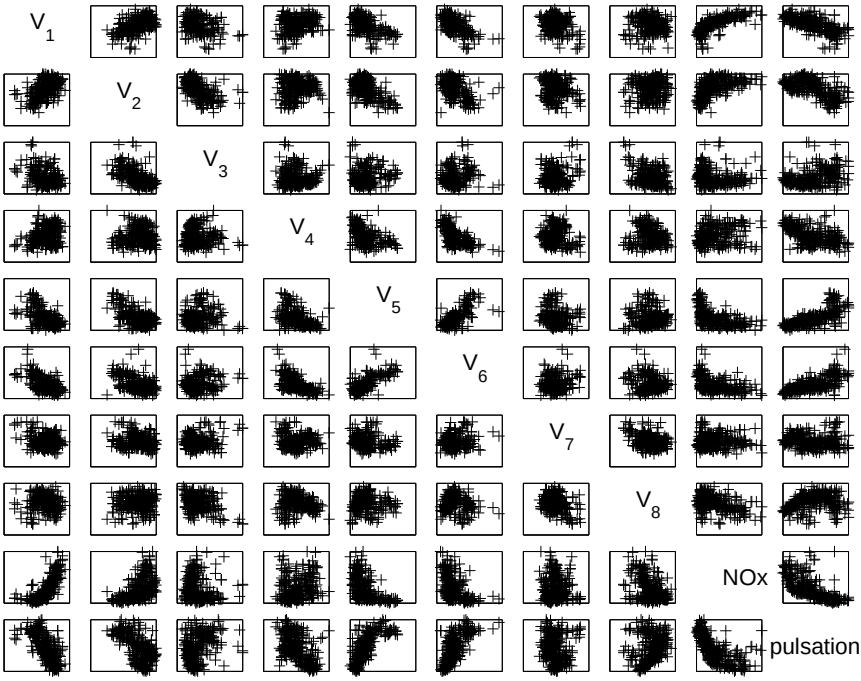

Figure 10.6: Proportional valves: Scatterplot showing the correlation of the valves $V_{i,i=1,\ldots,8}$ and the objectives NO_x and pulsation.

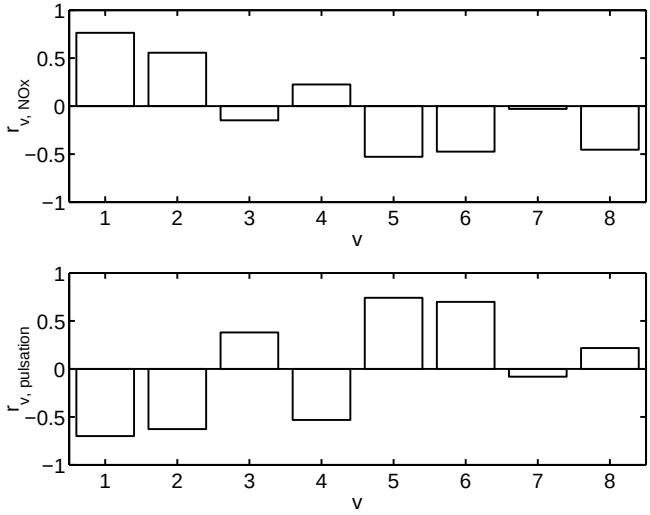

Figure 10.7: Proportional valves: Correlation coefficient r between the mass flow through the valves $V_{i,i=1,\ldots,8}$ and the objectives NO_x and pulsation.

10.4 Burner Optimization with Digital Valves

10.4.1 Valve Encoding and Optimization Algorithm

In the second case, 16 digital valves are used. Digital valves are binary switches, allowing only the two discrete states closed and open. These switches are less expensive to implement into a real machine and can be achieved by either closing an injection hole by welding or reopening it by drilling. As an operating constraint, at most 3 of the 16 valves are allowed to be closed, since for a constant total fuel flow, the open valves have to take over the additional flow.
The encoding of the digital valves in the evolutionary algorithm can be done with different approaches. We use 3 discrete variables with integer values between 1 and 16 to encode the position of the closed valves. This allows encoding all solutions that fulfill the constraint. All settings with 1, 2, or 3 closed valves can be obtained if all three, two or none of the variables are of equal value, respectively. The setting with all valves open is the standard machine design, which was evaluated for reference. Since permutating the variables does not lead to different solutions (e.g., setting the variables to $\{1, 4, 7\}$ is equivalent to $\{7, 4, 1\}$), the variables are always sorted in ascending order. Detecting permutations is important since permutations of the same solution do not contain new information and recombining different permutations should be avoided. The same recombination and mutation operators as for the proportional valves are chosen, except that here we have only 3 variables and the obtained values have to be rounded to integer values. Here, the small mutation strength changes to the solutions slightly by shifting mostly the closest valves to the adjacent positions. The optimization algorithm is NT-SPEA with a population and archive size of 15.

10.4.2 Optimization Results

One optimization run is conducted with NT-SPEA, evaluating in total 165 different valve settings. The results of the optimization run are given in Fig. 10.8 and show the possible reduction in NO_x and pressure pulsations, when compared to the given standard burner configuration. The given standard burner configuration is marked in the figure and represents a setting with equal mass flow through all valves. Some solutions of the optimization process dominate the standard configuration, i.e., are superior in both objectives. Thus, the optimization run is successful, delivering improved solutions for both objectives.
The valve settings corresponding to the different nondominated solutions of the

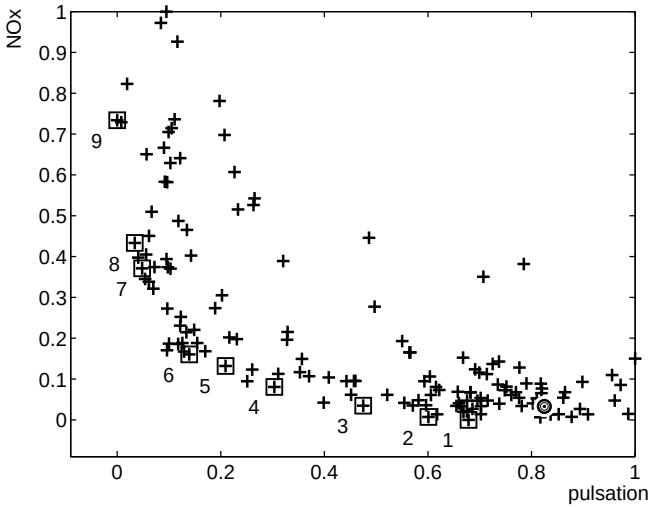

Figure 10.8: Digital valves: All measured solutions of the burner optimization run [plus symbol] and given standard burner configuration [circular symbol]. 9 boxes mark different solutions along the nondominated front.

optimization run are displayed in Fig. 10.9. Similar to the optimization with the proportional valves, the results indicate that for low NO_x, the center of the burner has to be leaner (closing valves 1 and 2 has the most significant effect). For low pulsations the center and the middle of the burner has to be enriched (more valves are closed in the area of the largest diameter of the conical burner). This is also underlined by the computed correlation coefficients in Fig. 10.10.
Consider that the objectives are normalized such that all measured solutions are within unit space. Thus, comparing relative and absolute values with the optimization run for proportional valves is not possible. However, the comparison of the resulting nondominated front against the given standard design shows similar improvements. In numbers, a design with a 20% improvement in NO_x emissions and concurrently 30% reduced pulsation was found.

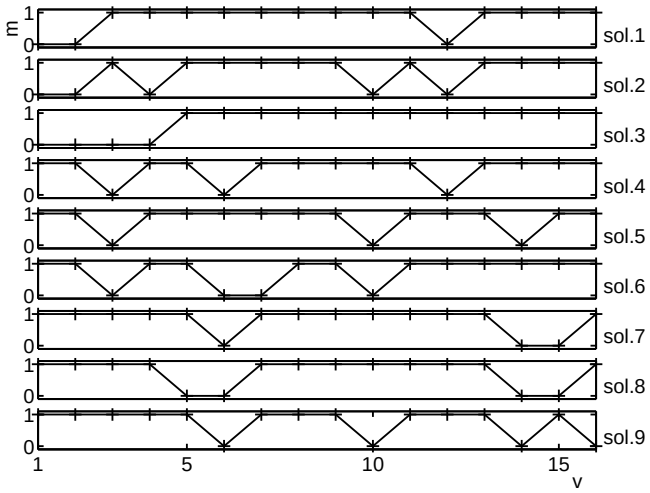

Figure 10.9: Digital valves: Mass flow $\dot{m}$ through the valves $V_{i,i=1,\ldots,8}$ for solutions along the nondominated front, marked by the 9 boxes of Fig. 10.4.

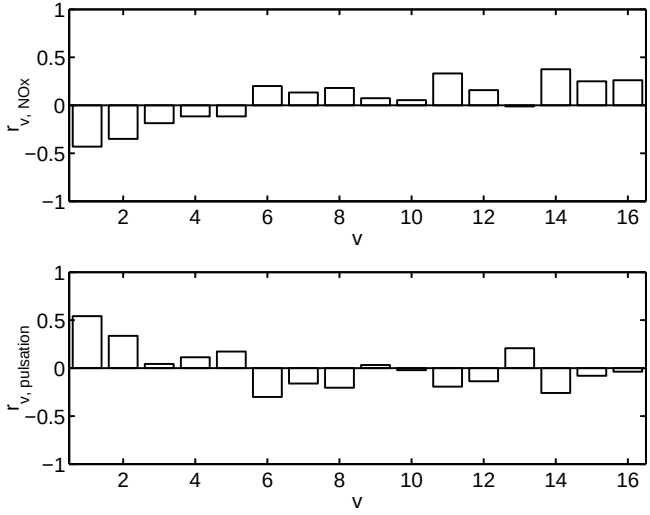

Figure 10.10: Digital valves: Negative correlation coefficient r between the mass flow through the valves $V_{i,i=1,\ldots,8}$ and the objectives NO_x and pulsation. Negative coefficients are given, in order to plot the effect for closing *a valve. Now, a positive value shows that the objective function increases, if the valve is closed.*

10.5 Conclusions

The evolutionary algorithm NT-SPEA is successfully applied to an automated optimization of a gas turbine burner. The optimization leads in an automated fashion to an experimental nondominated front for minimizing pulsation and emissions of the burner. Automated optimization can be considered a supporting tool in the design process, complementing physical understanding as well as trial-and-error design.
While the tests with the proportional valves are more valuable for the understanding of the physical problem by showing the correlation between valves and objective functions, the digital valves allow an easier practical use of the results. In an existing machine, the results for the digital values are easily realized by closing or reopening some of the fuel injection holes. This does not need any additional parts in the machine, but can change the characteristic of the burner significantly. These modifications might be necessary to adjust machines to different environmental conditions at different locations including ambient temperature and pressure or to different fuel.

Chapter 11

Optimization of Compressor Profiles and Blades

An automated optimization procedure for subsonic gas turbine compressor blades is presented. All relevant objectives and constraints of a realistic compressor design process are addressed. The procedure bases on the design of blade sections (profiles) by Q3D CFD. The common problem of stacking the profiles to a blade that fulfills smoothness and mechanical integrity aspects is avoided by encoding the profiles in a 3D parameterization and optimizing all profiles concurrently in one optimization procedure. Furthermore, we propose an inexpensive method to integrate the off-design behavior into the optimization procedure.
As a first step, the procedure is applied to the optimization of single profiles. Various optimization algorithms are compared and the resulting profile shapes are discussed.
In a second step, the procedure is applied to blade optimization. Four adjacent midstages of a compressor are optimized, starting from randomly initialized blades. All optimized blades show similar profile shapes and loading distributions. One blade is analyzed by 3D RANS simulations to validate the optimization results.

11.1 Compressor Design

The design process of multi-stage axial compressors consists of a sequence of design steps with models of increasing complexity. The first step is usually a thermodynamic model with loss correlations for the whole compressor. For each blade row, the model computes the pressure rise and the main geometrical dimensions based on the geometry of the mid-span section of the compressor blades. In the second step, a meridional through-flow code (e.g., a streamline curvature method) is used to compute a radial variation of the main aerodynamic properties like turning angles and Mach numbers. The meridional plane is denoted as the S2 plane and is illustrated in Fig. 11.1. These properties are computed on a number of distinct streamlines. Fig. 11.1 shows streamline 1, 5, and 9 for a compressor row.

The position of the streamlines results from an iterative procedure until the radial equilibrium equation is fulfilled.
The first steps depend mainly on the experience of the design engineers and the loss and derivation correlations in the models. Most dimensions and properties of the compressor are specified in these steps. Especially the aero- and thermodynamic boundary conditions of adjacent blade rows are defined. However, only a rough model of the blade shape exists, i.e., metal angles, stagger angles, and blade thickness are estimated.
The actual blade design is performed on the basis of two-dimensional (2D) cuts (profiles), which are stacked to a three-dimensional (3D) blade. These profiles are designed and analyzed on the streamlines from the preceding S2 calculation. For redesigning an existing compressor, streamlines can also be obtained by 3D Computational Fluid Dynamics (CFD) simulation of the given design [18]. In the considered design approach, the streamline shape is simplified to conical S1 surfaces (Fig. 11.1) that are symmetric to the compressor axis. The aerodynamic properties of a profile are analyzed by CFD on a 2D grid, which takes into account the streamline thickness and radius variation along the compressor axis. This analysis is referred to as quasi-3D or Q3D.
The objective of the profile design is to find shapes that fulfill all aerodynamic boundary conditions of the S2 calculation, especially in terms of the flow turning. In addition, low aerodynamic losses at the design point and at off-design conditions are desired. In the final step, the set of profiles for the different streamlines is stacked to a 3D blade. The blade must sustain aerodynamic and centrifugal forces, avoid critical resonance frequencies, and respect manufactural constraints such as smoothness. Thus, the profile design for the different streamlines is a coupled process and must be addressed simultaneously.
Q3D flow analyses do not capture 3D flow effects like secondary air flows and tip gap flows. 3D effects are often analyzed after the design process by 3D CFD simulations, since they are, even on today's computer clusters, computationally expensive [18]. In addition, the large number of rows in a compressor multiplies the overall design costs. This is a major difference to the design of aircraft wings. Wings are more often optimized by 3D CFD tools as, e.g., described in [75].
Q3D design is limited to subsonic flow due to the three-dimensional nature of shocks, especially since the compressor axis and the casing affect the shocks. A common difficulty of optimizing transonic flow for certain operating points is that the optimization may easily just improve the shock losses for considered operating conditions while worsening the performance elsewhere [58].
Concluding the described design process, a major part of the compressor design

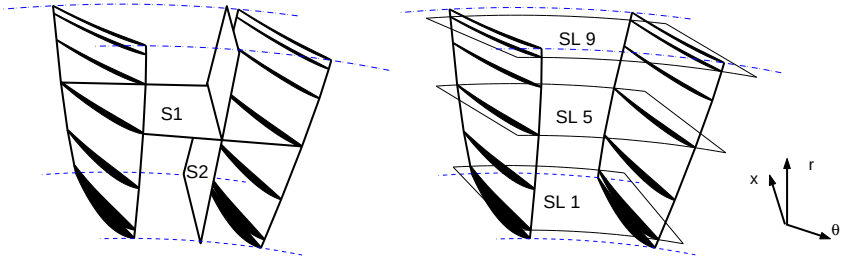

Figure 11.1: (left figure) The S1 plane is defined as a conical cut symmetric to the compressor axis ($r = f(x)$). The S2 plane is a cut at a certain meridional position ($\theta = const.$). (right figure) Between hub and casing [dashed lines], the compressor blade is constructed by stacking a set of profiles [bold lines and filled areas]. The profiles are positioned on different streamlines (SL1, SL5, and SL9), which are S1 planes.

is concerned with the design of 2D profile sections and the stacking to mechanically admissible 3D blades. This task comprises several (partly conflicting) objectives and constraints and is solved in a time-consuming iterative process. Here, an automated optimization procedure would lead to the most significant labor time reduction for the designers.

11.2 Survey on Automated Compressor Optimization

Automatic optimization of compressor profiles is mainly distinguished between inverse and direct design. In inverse design, a pressure distribution is predefined and the according profile shape is searched for by an iterative modification of the profile shape. Depending on the considered approach, the computational cost can be proportional to a single flow analysis and thus comparably cheap [92, 82]. However, the pressure distribution is usually iterated, since it may lead to an unacceptable profile shape. This approach relies significantly on the experience of the designer, who needs to specify a pressure distribution that meets various aerodynamic design aspects in terms of flow turning, boundary layer properties and losses, and which performs also well for off-design condition. According to Reuther *et al.* [92], varying the pressure distribution cannot guarantee to improve a design. For some pressure distribution, no profile shape exists. This occurs especially when prescribing 3D pressure distributions. A shortcoming of inverse

design is also the open question of how to integrate geometrical and mechanical constraints [106].
Direct design optimization considers the shape optimization for secondary aerodynamic properties like aerodynamic losses. The optimization costs are a multiple of a single flow calculation, since a large set of different designs needs to be evaluated. About 20 years ago, Sanger (1983) presented first optimization results for compressor profiles using numerical flow solvers. His optimization of high-subsonic compressor profiles contained already a 2D potential flow solver with an integral formulation of a compressible boundary layer in conjunction with a gradient-based constraint optimization approach (CONMIN).
In direct design, off-design conditions can be included by performing CFD calculations for various operation conditions [32, 67, 58]. Typically, off-design conditions are obtained by varying the inlet flow angle or the Mach number. Geometrical constraints can be included in the parameterization of the blade and mechanical aspects can be considered by adding penalties if certain limits are violated. Blades are usually designed by stacking a set of profiles such that the centers of gravity of the profiles are in a line. This reduces bending moments by centrifugal forces. While for classical NACA profile families, the center of gravity is designed at a similar chord position, optimized blades may have different positions such that the stacking results in a warped shape [107]. Solving this problem may require a simultaneous optimization of all profiles while considering stacking constraints.
Optimization algorithms for direct design are mainly gradient-based [18, 32, 72, 92] methods and stochastic algorithms [82, 114, 119]. Also hybrid approaches combining gradient-based methods and stochastic algorithms are employed [67]. Gradient-based methods rely on derivative information of all objectives and all constraints for determining the search direction of the optimization as discussed in Subsection 1.2.2. Since not all computer programs of the considered optimization loop are given in source code, the application of adjoint formulations[92], automatic differentiation [12], and complex-step method [76] is not possible and the only way to obtain derivatives is finite-differencing. Finite-differencing results in inexact gradients [76] and thus gradient-based methods will not be considered.

11.3 Optimization Approach

We consider a direct optimization procedure for compressor blades. The approach comprises an optimization algorithm, a parameterization for 2D profiles and 3D blades, and numerical tools for aerodynamic and mechanical integrity analysis.

The aerodynamic performance is analyzed by the quasi-3D CFD code MISES [31] on several radial blade sections. The mechanical integrity is analyzed concerning maximal stresses and eigenfrequencies of the 3D blade by a finite element beam model.
The focus of this optimization procedure is on obtaining realistic compressor blades. Thus, the optimization procedure has to take into account all relevant design objectives and constraints and should result in similar or improved blade shapes compared to the manual design process. In most optimization approaches, the compressor blade is stacked out of a set of profiles, which are optimized independently. Stacking these profiles is difficult as the various eigenmodes of the resulting blade (e.g., modes for bending, torsion, and mixed modes) are difficult to predict given only the profile shapes and may conflict with critical eigenfrequencies. Furthermore, the resulting blade might have a warped shape [107]. We avoid this stacking problem by optimizing the different profiles concurrently while addressing smoothness and mechanical integrity aspects. Smoothness constraints are obeyed by encoding the profiles by a 3D blade parameterization, which is restricted to smooth designs. The mechanical integrity properties are directly computed from the 3D blade and are transferred into constraints. A minor focus of the procedure is on generating new blade shapes or design philosophies. Design philosophies can be included in the optimization by restricting the engineering parameters as, e.g., the curvature or the wedge angle at the leading edge.
This optimization procedure provides answer to the following three questions:

- *3D parameterization:* How can a profile parameterization be extended to a 3D blade parameterization for smooth blades?
- *Fitness function:* Which objectives and constraints are relevant in the compressor design and how can they be formulated in mathematical terms? How can off-design performance be included into the fitness function?
- *Optimization algorithms:* What are efficient optimization algorithms that require a small number of design evaluations?

In the following, we will describe a 3D parameterization and define a fitness function. Efficient optimization algorithms result from a performance comparison. The comparison is performed for a single profile reducing the overall computational cost. This allows performing more optimization runs given the limited computing resources.

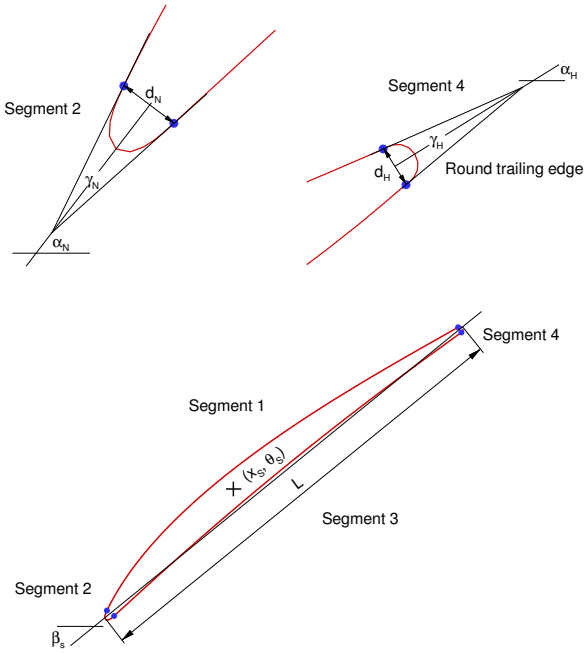

Figure 11.2: Some engineering parameters of a compressor profile.

11.3.1 Parameterization of Compressor Profiles

The compressor profile is parameterized by four Bezier-spline segments (suction surface, leading edge, pressure surface, and trailing edge). Each segment is defined by 6 control points with two coordinates, leading to 48 parameters in total. 16 parameters are determined by enforcing $\mathcal{C}^2$-continuity between the segments. Introducing simplifications for the trailing edge (circular) and leading edge (elliptical), the remaining parameters are translated into 19 engineering parameters, such as metal angles, wedge angles, profile length, etc., as shown in Fig. 11.2. Compared to the control points, engineering parameters simplify the comparison and illustration of different profile parameter sets.

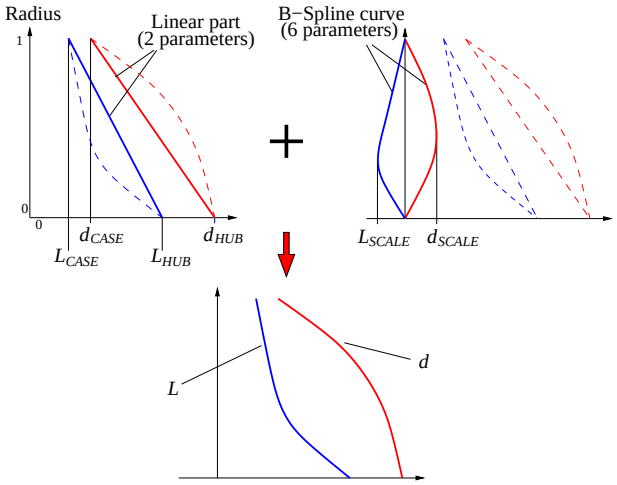

Figure 11.3: Exemplary construction of radial shape functions by superimposing a linear part (upper left) and a B-spline (upper right) for two engineering parameters L and d.

11.3.2 Parameterization of Compressor Blades

Compressor blades are usually stacked from a set of compressor profiles with each profile being described by a set of engineering parameters as introduced in Subsection 11.3.1. Each profile section is located on a different streamline of the S2 calculation. The radial variation of the engineering parameters and some derived quantities such as the relative profile thickness are subject to design rules. A blade is considered acceptable if these quantities are continuous along the blade span without turning points.

These rules can be obeyed by using radial shape functions for the variation of the engineering parameters between the hub and the casing streamlines. These parameters of the shape functions are then used for the blade optimization. The radial shape function consists of two parts, namely a linear variation and an additional B-spline curve that allows for nonlinear variations as illustrated in Fig. 11.3. The linear part implies two free decision variables, i.e., the value of the profile parameter (e.g., profile length L) at hub and casing, L_{HUB} and L_{CASE}, respectively. As shown in Fig. 11.4, the shape of the B-spline curve is defined by 5 parameters L_{SF0} to L_{SF4}. These parameters define the location of three B-spline points,

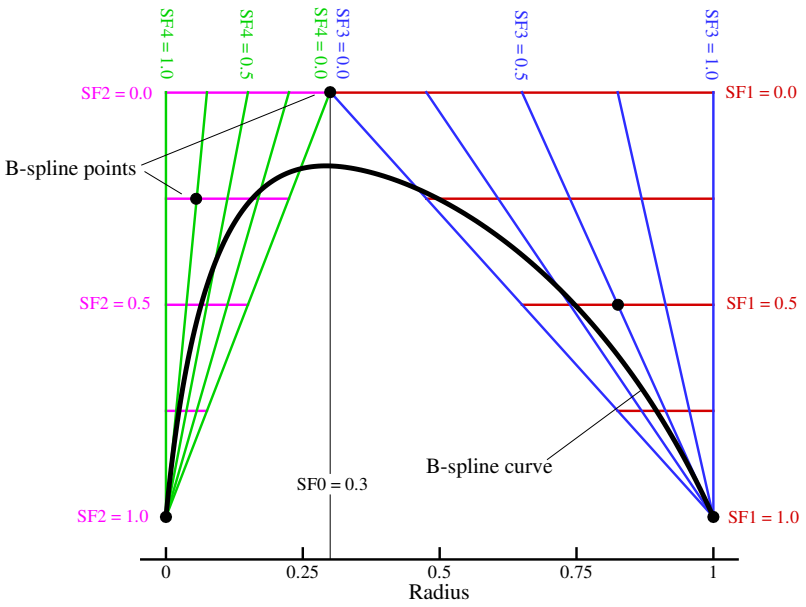

Figure 11.4: Definition of the chosen B-spline curve

which form the control polygon together with a start and end point. The convex hull property of B-splines implies that a convex control polygon leads to a convex curve. In order to avoid turning points in the curve, it is therefore necessary to ensure a convex polygon. This is achieved by a triangular coordinate system. The admissible range of L_{SF0}-L_{SF4} is between 0 and 1. An additional parameter L_{SCALE} determines, how strongly the radial shape function deviates from a straight line. Note that L_{SCALE} = 0.0 leads to a linear variation along the span.

11.3.3 Mechanical Integrity Analysis

For the compressor blades, the mechanical integrity (MI) analysis is solved by a finite element beam model that checks, whether a given blade shape can sustain the aerodynamic and centrifugal forces and if the eigenfrequencies differ from critical excitation frequencies. The results are translated into a number of safety factors.

The factors are positive or zero, if the constraint is violated or not, respectively. The sum of all non-zero factors (violated MI criteria) is called MI indicator.
For a profile design, we simplify the MI analysis by defining a lower limit for the blade thickness.

11.3.4 Q3D Flow Analysis

The aerodynamic properties of compressor profiles and blades are analyzed with the Q3D solver MISES [31]. MISES solves an Euler equation for a single blade row and a single flow path, assuming periodic boundary conditions between two adjacent blades in a row. Viscous effects are modeled by a fully coupled integral boundary layer formulation. The stream tube contraction as well as radius variations along the compressor axis are considered (Q3D).
A performance of a profile or blade is assessed by performing several analysis for different inlet flow angles, while keeping the inlet Mach number constant. Fig. 11.5 shows the aerodynamic losses as a function of the inlet angle. The typical loss polar has a flat region around the design inlet angle and a sharp increase of losses for larger deviations from this design point. The operating range is defined by the range of incidence angles with losses less than twice the losses at design incidence. Usually the operating range decreases with increasing inlet Mach number. For a blade optimization, the aerodynamic properties are analyzed for each profile section separately.

11.3.5 Objectives and Constraints

For the optimization of compressor profiles or blades, two objectives are formulated as well as a set of constraints. The first objective is to minimize the sum of the aerodynamic losses l for the design inlet flow angle α_D and off-design conditions. Low losses l_{α_D} at design condition are important for a high efficiency of the gas turbine when operating as a base load machine under full load. Low losses at off-design conditions are required when operating also under part load or modified environmental conditions. Since computing the complete loss polar of a compressor profile or blade is very time-consuming, a simplified approach is proposed. As an indicator for the robustness to off-design condition, the flow is calculated at two additional inlet flow angles $\alpha_\mathrm{D} \pm \Delta\alpha$. The resulting losses $l_{\alpha_\mathrm{D}-\Delta\alpha}$ and $l_{\alpha_\mathrm{D}+\Delta\alpha}$ are added to the loss at the design incidence.
The losses at the off-design incidences are dependent on the blade shape and on the incidence variation $\Delta\alpha$. With increasing incidence variation $\Delta\alpha$, the losses

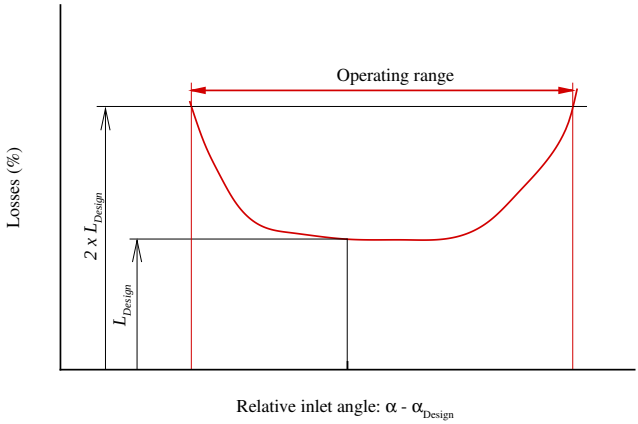

Figure 11.5: Definition of the operating range by loss limits.

increase until the blade stalls. Thus, the inlet angle variation $\Delta\alpha$ has to be set carefully. On one hand, if $\Delta\alpha$ is set too small, the off-design conditions are not very well captured. On the other hand, if $\Delta\alpha$ is set too large, the flow computation may fail due to strong flow separation or stall.
We propose to define the inlet variation $\Delta\alpha$ as the second objective, which is to be maximized. Furthermore, we propose to define the inlet angle variation $\Delta\alpha$ as an additional decision variable, i.e., the inlet angle variation is modified by the optimization algorithm. Then, in the optimization, solutions with large values for $\Delta\alpha$ are preferred.
Aerodynamical constraints are set for the flow properties at design inlet flow angle. The first constraint is set on the deviation of the turning angle $\Delta\beta$ from the target value. Furthermore, Mach number Ma and the non-dimensional shape factor H_{12} at the suction side trailing edge are limited by an upper bound constraint. The latter two quantities are limited in order to avoid transonic effects and flow separation, respectively. For every flow calculation that fails, an additional penalty is added. The value of this penalty is set such that it is higher than the largest possible aerodynamic loss. Mechanical constraints are integrated in the M_{I} indicator, which sums all violated mechanical constraints.
All objectives and constraints are summarized in Table 11.1. For the blade opti-

Table 11.1: Objectives and constraints for the profile and blade optimization

symbol	description	goal
objectives		
l_{α_D}	aerodynamical loss at design and off-design	minimize
$l_{\alpha_D-\Delta\alpha}$, $l_{\alpha_D+\Delta\alpha}$	incidences	
$\Delta\alpha$	inlet angle variation	maximize
constraints		
$\lvert\Delta\beta\rvert$	absolute deviation from the design turning angle	$< -0.1°$
H_{12}	non-dimensional shape factor of the boundary layer	$< H_{12}^{\text{limit}}$
Ma_{max}	maximal Mach number	$<\text{Ma}_{\text{max}}^{\text{limit}}$
M_I	mechanical integrity indicator (considers eigenfrequencies and stresses for the blade optimization and solely the profile thickness for profile optimization)	< 0

mization, the aerodynamical objectives and constraints are computed at 3 streamlines.

11.4 Profile Optimization

We consider the optimization of a mid-span compressor profile as a performance comparison problem for evolutionary algorithms. As decision variables of the optimization, all 19 engineering parameters described in Subsection 11.3.1 and the incidence variation $\Delta\alpha$ from Subsection 11.3.5. The objectives and constraints are taken from Subsection 11.3.5.

We consider two different approaches. In the first approach, the two objectives and constraints are aggregated into one fitness function. The performance of all evolution strategies introduced in Chapter 2 is compared with the Gaussian Process Optimization Procedure (GPOP) of Chapter 8. In the second approach, the two objectives are optimized as a Pareto optimization problem. As these two objectives are conflicting, the optimization results in an wide front of nondominated

solutions. Here, the SOMX-NSGA-II (Chapter 7) and SPEA (Chapter 3) are compared with the single-objective algorithms.
After comparing the different algorithms, the properties of two different nondominated solutions are discussed. The two different designs are chosen in order to illustrate the trade-off between the two objectives.

11.4.1 Objective functions

The objectives and constraints from Subsection 11.3.5 can be written as two objectives f_1 and f_2 and a penalty function p that includes all constraints with:

$$\begin{aligned} f_1 &= l_{\alpha_D} + l_{\alpha_D - \Delta\alpha} + l_{\alpha_D + \Delta\alpha} \\ f_2 &= \Delta\alpha \\ p &= a_1 \max(0, |\Delta\beta| - 0.1^\circ) \\ &\quad + a_2 \max(0, H_{12} - H_{12}^{\text{limit}}) \\ &\quad + a_3 \max(0, \text{Ma}_{\max} - \text{Ma}_{\max}^{\text{limit}}) \\ &\quad + a_4 \max(0, d^{\text{limit}} - d), \end{aligned} \tag{11.1}$$

where $a_{\{1,2,3,4\}} > 0$ are user-defined weights for the different penalties. Objective f_1 sums the losses at the 3 incidences and is to be minimized. Objective f_2 contains the incidence multiplier $\Delta\alpha$ and is to be maximized for maximizing the operating range of the profile.
These two objectives can be either aggregated into a single fitness function or optimized by a Pareto optimization technique, resulting in an approximation of the Pareto front for the two objectives. The penalty function is constructed from four constraints, which are set on the exit flow angle deviation $\Delta\beta$ from the design angle, shape of the boundary layer H_{12}, and Mach number Ma and on a simple mechanical constraint specifying the minimal blade thickness.

11.4.2 Single Objective Optimization

For optimization algorithms that require a single objective function, all objectives and constraints have to be aggregated. We formulate this aggregation for minimization with:

$$f = f_1 - a_0 f_2 + p, \tag{11.2}$$

where $a_0 > 0$ weights the two objectives.

Optimization Algorithms

As representatives for evolution strategies, we choose Rechenberg's Success Rule, the Evolution Strategy with Rotating Angle Mutation (ROT-ES) and the Evolution Strategy with Covariance Matrix Adaptation (CMA-ES) and compare these algorithms with the GPOP algorithm proposed in Chapter 8. The Success Rule is defined for a $(1+1)$ evolution strategy and includes the two parameters c and p_s, which specify a multiplicative factor for adapting the step size and the success rate, respectively. We set $c = 0.817$ as proposed by Schwefel [101]. The success rate is usually set to $p_\mathrm{s} = 1/5$, however we follow the recommendation of Rechenberg [90]) and Schwefel [102] to use smaller success rate for noisy and constrained problems. Thus, setting $p_\mathrm{s} = 1/10$ and $p_\mathrm{s} = 1/20$ is also analyzed.
For ROT, we use the standard settings (Section 2.3) with a $(15, 100)$ selection scheme. A $(2, 10)$ and $(3, 12)$ selection scheme is chosen for CMA-ES.
For GPOP, we choose the first approach in Subsection 8.3.3 and model the losses at the 3 incidences and constraints with a separate Gaussian process (GP). In total, 7 GPs have to be trained. The objective function is then constructed from the GP predictions as given in the equation for the fitness function in Eqn. 11.1.
The GPOP algorithm operates differently than the evolution strategies. From the set of evaluated profiles, GPOP constructs a model of the objective function and then searches the minimum of the model prediction for different merit functions. The resulting minima are then evaluated and finally added to the data set to improve the model.
As training data for the GPs, only converged MISES calculations are chosen. The training data for the GPs results from three MISES computations at different incidences. Since each incidence computation might fail for different solutions, the number of training data for the GPs differs depending on the objective or constraint that the GP models. The GPOP algorithm requires also an uncertainty measure for the predicted value objective function to construct the merit functions. This standard deviation is taken solely from the predicted standard deviations for the losses at the 3 incidences. As training data for the GPOP algorithm, the N_C closest solutions to the currently best solution and N_R most recent evaluated solutions are chosen with $N_\mathrm{C} = N_\mathrm{R} = 15$, 30, 60, or 120, respectively.
For all optimization runs, decision variables are bounded. These bounds are set using knowledge of design engineers and from previous optimization runs, such that the bounds limit the search space to mainly physical designs. For example, generating profiles with zero or negative thickness is avoided. For the optimization, these bounds are scaled such that optimization algorithms operate on a unit

space within $[0, 1]$. Scaling is necessary since the decision variables have different orders of magnitude and units. The initial step size for all evolution strategies is set to 0.1.

Optimization Results

The convergence of the evolution strategies is given in Fig. 11.6 and of the GPOPs in Fig. 11.8. In the beginning of the optimization till about $N = 200$ evaluated solutions, the convergence of all evolution strategies is about similar. This may result from the equal initial step size, which is only minor changed by the different adaptation schemes. However, the population size and other properties differ between the evolution strategies leading to a small difference in convergence.
The convergence of the Success Rule is dependent on the setting of the success rate p_s, which is set to p_s =1/10, and 1/20. A success rate of 1/20 leads to better performance and results in a smaller final objective function value. The step size is plotted in Fig. 11.7. For setting p_s to 1/10, the step size is constantly reduced. In other words, the fraction of successful mutations is always lower than 1/10 and thus, the step size is never increased and becomes inefficiently small. After $N = 700$ evaluations the step size is already smaller than 10^{-3} and convergence stagnates. For setting $p_\mathrm{s} = 1/20$, the step size is both reduced and increased during the optimization, but after about $N = 1250$ evaluations, the step size becomes also inefficiently small. These observations for the success rate agree with Schwefel and Rechenberg's recommendation that for constraint problems, the success rate should be reduced.

The ROT-ES shows a much slower convergence than the Success Rule with $p_\mathrm{s} = 1/20$ success rate and also the final fitness function value is worse. Fig. 11.7 shows that the step sizes do are in average about constant over the $N = 3.000$ evaluations. The optimization is not yet converged and the fitness function value might still improve.
The CMA-ES shows the fastest convergence and also results in the best fitness function value. After 3000 evaluations, the algorithm is converged and the mutation is small.
The number of evaluations and the best function value obtained by the different algorithms is summarized in Table 11.2.

The convergence observations for the different optimization algorithms allow some conclusions about the optimization problems. The Success Rule operates with an isotropic mutation distribution that converges efficiently for functions that are well scaled [9]. In other words, plotting contour lines of the functions are

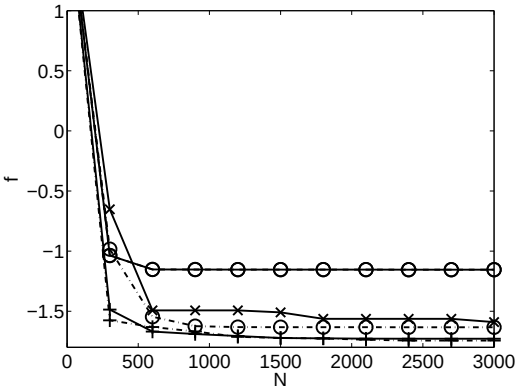

Figure 11.6: Convergence of the Success Rule with 1/10th [solid line with o] and 1/20th [dash-dotted line with o] success, ROT-ES [solid line with x] and (2,10)-CMA-ES [solid line with $+$] and (3,12)-CMA-ES [dashed line with $+$].

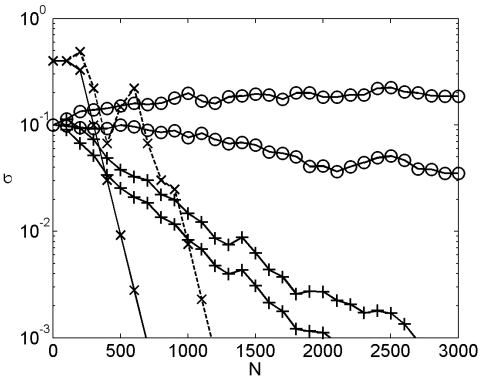

Figure 11.7: Plot of the step size of the Success Rule with 1/10th [solid line with o] and 1/20th [dash-dotted line with o] success over the number of evaluation. For ROT-ES [solid line with x] and (2,10)-CMA-ES [solid line with $+$] the minimal and maximal eigenvalues of the covariance matrix is plotted.

nearly circular. For badly scaled functions (e.g., the contour lines are ellipses with large aspect ratios), the algorithm converges poorly due to the isotropic mutation. Since Success Rule works well on the profile design problem, it can be assumed that the decision variables are nicely scaled. The plot of the eigenvalues of the covariance matrix also for CMA-ES also supports this thesis. The difference between the maximal and minimal is at most 3.5 and thus only slightly scaled. However, one can observe that CMA-ES, which can adapt to scaled and correlated decision variables, requires less function evaluations and results in a better objective value. It seems that the correlated mutation of CMA-ES is beneficial. ROT-ES adapts also correlated mutation. However, the adaptation scheme is less efficient [52] and is problematic for constraint optimization problems [52] as shown in Subsection 5.3.2.

For GPOP, different training set sizes are analyzed with $N_{\mathrm{C}} = N_{\mathrm{R}} = 15$, 30, 60, or 120. The convergence of the GPOP algorithm is shown in Fig. 11.8. For all training set sizes, the algorithm converges to similar final objective function values. Compared to all evolution strategies, the final value for each training set size is superior and is reached within only about 1/7 of the number of evaluations (see Table 11.2).

The good convergence of GPOP for the small training set size $N_{\mathrm{C}} = N_{\mathrm{R}} = 15$ allows similar conclusions about the optimization problem as found for the evolution strategies. We found in the analysis of GPOP in Section 8.4 that for such small training set sizes, GPOP is only able to optimize problems with 20 decision variables, as in the profile optimization, if the problem is well scaled.

11.4.3 Multi-Objective Optimization

In the multi-objective setup, we consider minimizing the two objectives of f_1 and f_2 of Subsection 11.4.1 in a Pareto setup:

$$\begin{aligned} \text{minimize } f_1' &= f_1 + p \\ \text{maximize } f_2' &= f_2 - p \end{aligned} \tag{11.3}$$

The first objective function f_1' comprises the losses at the 3 incidences and the penalty term p (see Eqn. 11.1 and is to be minimized. The second fitness function f_2' contains the incidence variation $\Delta\alpha$ and the penalty and is to be maximized for increasing the operating range of the profile. The penalty term is subtracted from f_2 in order to degrade solutions with violated constraints.

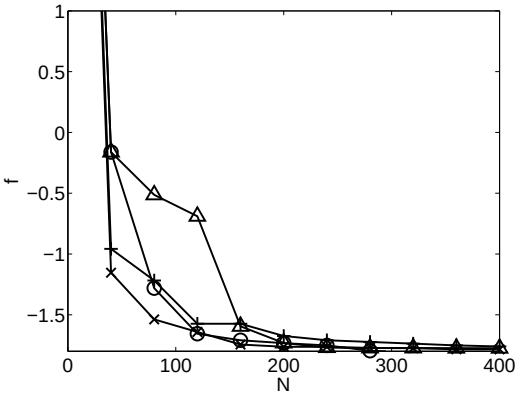

Figure 11.8: Convergence of the GPOP algorithm with a training size of $N_{\mathrm{R}} = N_{\mathrm{C}} = 15$ [x], 30 [+], 60 [o], and 120 [Δ].

Table 11.2: Comparison of different optimization algorithms for the profile optimization based on the best function value and the number of design evaluations

Optimization algorithm	$\min(f)$	N
Evolution Strategy		
1/5th success rule	-1.15	3000
1/10th success rule	-1.15	3000
1/20th success rule	-1.63	3000
$(15, 100)$-ROT-ES	-1.59	3000
$(2, 10)$-CMA-ES	-1.73	3000
$(3, 13)$-CMA-ES	-1.74	3000
Gaussian Process Optimization Procedure (GPOP) with different training data sizes		
$N_{\mathrm{C}} = N_{\mathrm{R}} = 15$	-1.78	400
$N_{\mathrm{C}} = N_{\mathrm{R}} = 30$	-1.76	400
$N_{\mathrm{C}} = N_{\mathrm{R}} = 60$	-1.82	400
$N_{\mathrm{C}} = N_{\mathrm{R}} = 120$	-1.77	400

The Pareto front of the two objective functions consists of solutions, that show minimal losses at the 3 incidences for a certain incidence variation $\Delta\alpha$. While Pareto solutions with small values $\Delta\alpha$ are optimized for low losses near design condition, Pareto solutions with large values of $\Delta\alpha$ are optimized for low losses at off-design conditions.
The incidence multiplier in f_2 is also defined as an additional decision variable. Its value is limited to $\Delta\alpha \in [4°, 7°]$. While the lower bound on $\Delta\alpha$ leads to profiles that have a minimal off-design performance, the upper bound is set for limiting the optimization to incidence values that MISES can handle without failing too often. The values are set for the considered mid-span compressor blade and are dependent on boundary conditions like Mach numbers and profile loading.

Optimization Algorithms

We consider two Pareto optimization algorithms that lead to an approximation of the Pareto front in a single optimization run. We compare the Strength Pareto Evolutionary Algorithm (SPEA) introduced in Chapter 3 with the SOMX-NSGA-II proposed in Chapter 7. For SPEA, the mutation and recombination operator are taken from Section 3.3. The recombination operator always selects two parents applies either intermediate recombination, discrete recombination or randomly selects one parent with a probability of $1/3$ each. The mutation is normally distributed with a constant step size of $\sigma = 0.01$, relative to the intervals of the decision variables. For SOMX-NSGA-II, all settings of Chapter 7 are used.

Optimization Results

Figure 11.9 shows the resulting nondominated front for SOMX-NSGA-II and SPEA after 10.000 evaluations. The nondominated front is constructed only from solutions that do not violate any constraint. The two objectives of the optimization are to minimize the sum of losses at 3 incidences and to maximize the incidence multiplier. The non-dominated front shows that the two objectives are conflicting. For increasing the incidence multiplier, the sum of losses increases.
In order to estimate the convergence of the two Pareto optimization algorithms, the results from the previous single-objective optimization runs are added to the figure. For the single objective optimization the incidence variation $\Delta\alpha$ was bounded either to $\Delta\alpha = 4$, $\Delta\alpha = [4, 5.5]$, or $\Delta\alpha = [4, 7]$, in order to obtain different compromise solutions. All single objective runs result in solutions that are nondominated by the two Pareto optimization runs. In other words these runs dominate a

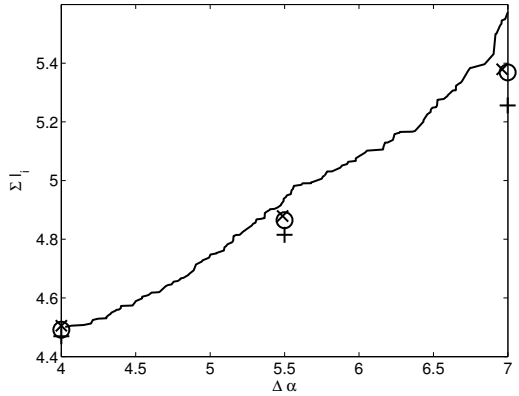

Figure 11.9: Resulting nondominated front for optimizing the two fitness functions with SOMX-NSGA-II [dashed line] and SPEA [solid line]. For comparison, the single objective optimization runs with the 1/20 Success Rule [o], ROT-ES [x], CMA-ES [+] and GPOP[Δ] are added.

fraction of the Pareto optimization results.
Since this is the case for all single objective runs, it can be concluded that the single objective runs lead to better results. While the 1/20 Success Rule and ROT-ES show just minor improvements to the Pareto optimization algorithms, the results of CMA-ES and GPOP are clearly superior. The differences between the Pareto optimization and single objective optimization increase with rising $\Delta\alpha$, since designing profiles for strong off-design is more difficult that designing for the design condition.
Comparing the computational cost, the Pareto optimizations evaluated 10.000 solutions, while the 1/20 Success rule, ROT-ES and CMA-ES evaluated 3.000 solutions and GPOP only 400 solutions. Here, the single objective optimization algorithms are advantageous, if only a small number of nondominated solutions is required. Concluding the performance comparison for the profile optimization, single-objective algorithms are preferable to the Pareto optimization algorithm for both the quality of the resulting designs as well as the lower number of design evaluations, if only a few compromise solutions are required.

11.4.4 Discussion of the Optimized Compressor Profiles

In the following the aerodynamical properties of the optimized profiles as well as the conflict of the two objectives are discussed. We consider the result of the two optimization runs with the GPOP algorithm as shown in Fig. 11.9, denoted as profile A and B, respectively. While profile A is designed for a small incidence multiplier $\Delta\alpha = 4°$, design B is optimized for a large incidence multiplier $\Delta\alpha = 7°$. Thus, design A should be optimized for small incidence variations from the design condition and design B for off-design condition.
Comparing the profile shapes in Fig. 11.10, the profiles are quite similar, especially on the first 30% of the chord for the suction side, where profile A shows a slightly higher curvature. A major difference can be found on the pressure side; the profile A is almost straight over the chord length and profile B shows a double bended shape. The bended shape shifts the maximal profile thickness towards the leading edge to about 30% chord.
The loss polar for the two profiles are given on the right of Fig. 11.10. We defined the operating range as the range of incidence angle variation with losses below twice the losses at design incidence. Profile B shows with $15.6°$ a larger operating range than profile A with $14.0°$. The wider operating range is gained at the expense of the losses at design condition, which are for profile B about 2.5% higher than for profile A. In addition, profile A shows lower losses than profile B within an incidence variation of $-4°$ to $6.5°$ from the design incidence. This coincides with the goal of optimizing the blade at $\pm 4°$ incidence variation. The main increase in operating range of profile B compared to A is for negative incidence variations below $-4°$, while the improvement at positive incidences is just minor and mainly at $7°$. This angle equals the incidence multiplier.
The Mach number distributions of the two profiles are plotted in Fig. 11.11. Both profiles show a strong acceleration of the flow on the suction side with the Mach peek before 10% chord. After the Mach peek, the blades show a strong diffusion, leading to a front-loaded characteristic. On the pressure side, profile A shows an about constant Mach number similar to a controlled-diffusion airfoil (CDA). For profile B, the demand of increased operating range resulted in a strongly bended pressure side geometry with a larger wedge angle and radius at the leading edge. MISES predicts an early transition onset at about 3% chord for both profiles and both suction and pressure side.
The Mach number distribution on the suction side and the transition onset are in contrast to conventional CDAs. CDAs base on the assumption that the flow in the compressor is laminar at least in the area of favorable pressure gradients, i.e., as

long as the flow is accelerated. Since laminar boundary layers show lower losses than turbulent boundary layers, the Mach peek of CDAs is designed as far as possible from the leading edge, typically at 15 to 30% chord. After the peek, the flow is strongly decelerated and transition occurs. For CDAs, the strength of the deceleration decreases monotonously towards the trailing edge.
Schreiber *et al.* [99] analyzed CDAs in an experimental compressor cascade. For moderate Reynolds numbers ($\mathrm{Re} < 0.8 \cdot 10^6$) and turbulence levels ($Tu < 3\%$), their results show a laminar boundary layer on the suction side with a laminar separation and turbulent re-attachment shortly after the Mach peek. However, they show also that for the high Reynolds numbers ($\mathrm{Re} \approx 2 - 6 \cdot 10^6$) and turbulence levels ($Tu \approx 4\%$) as they exist in the considered blade rows, the transition occurs upstream of the Mach peek, shortly after the leading edge. Here, bypass transition is important, caused by impinging wakes from upstream compressor blades on the boundary layer, turbulent spots in the flow or surface roughness [99]. Based on their experimental analysis, Schreiber *et al.* [99] suggested front loaded designs with an early Mach peak on the suction side.
The optimized blades agree with these analysis twofold. First, MISES predicts an early transition for the optimized blades. Second, the turbulent flow is then strongly accelerated with an early Mach peek, leading to a front-loaded design. As transition occurs close to the leading edge, there is no advantage of a moderate acceleration. The thin and turbulent boundary is robust and can be strongly accelerated and decelerated. The benefit of the front loaded design is the resulting moderate deceleration for the less stable and thicker boundary layer at mid-chord and trailing edge.
Köller *et al.* [67] and Sieverding *et al.* [107] performed automated optimizations for single profiles at comparable Mach numbers and turbulence levels. Their optimizations show similar front loaded designs. Front loaded designs have been experimentally investigated on cascades by Küster *et al.* [70] and show an improved operating range and reduced losses compared to CDAs. Furthermore, the investigations showed that MISES predictions for the transition onset agree with the experimental data. However, experimental analysis in a compressor test-rig [107] showed no efficiency improvement. While there is strong evidence that the front loaded designs operate well for mid-span profiles, it is from our point of view still an open question how these designs operate at the blade tip and hub, where secondary airflows and the tip gap strongly influence the flow. The RANS simulation for the blade optimization in the next subsection addresses this question.

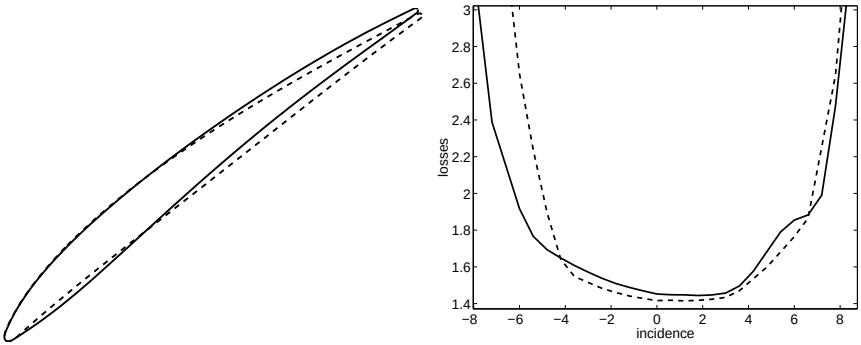

Figure 11.10: Comparison of the profile shape (left) and loss polar (right) for two profiles, which are optimized for an incidence variation of $\Delta\alpha = 4^\circ$ [dashed line] and $\Delta\alpha = 7^\circ$ [solid line], respectively.

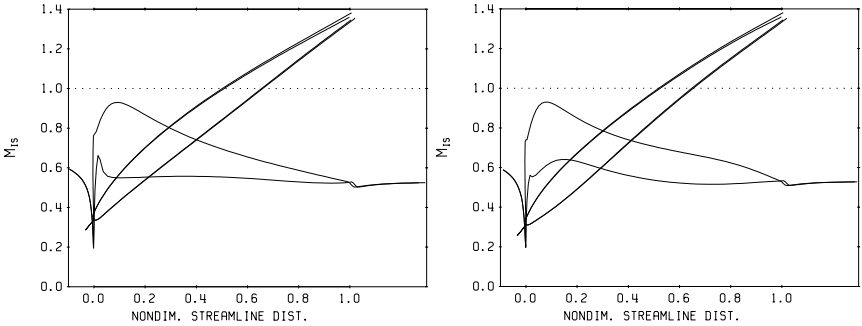

Figure 11.11: Comparison of the Mach number distributions at design condition for two profiles, which are optimized for an incidence variation of $\Delta\alpha = 4^\circ$ (left) and $\Delta\alpha = 7^\circ$ (right), respectively.

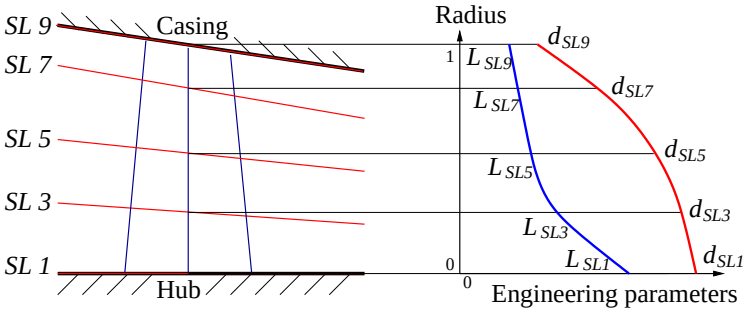

Figure 11.12: Link between radial shape function and engineering parameter

11.5 Blade Optimization

11.5.1 Blade Parameterization

For the blade optimizations, the decision variables are the parameters for the radial shape functions as introduced in Subsection 11.3.2. To limit the computational effort in optimizing compressor blades, the number of computed streamlines is often reduced to 3 [107], with one streamline close to the hub, at mid-span, and close to the blade tip. Describing each radial shape functions with 3 parameters is thus sufficient. From the set of parameters $*_{HUB}$, $*_{CASE}$, $*_{SCALE}$ are chosen and the five shape factors $*_{SF0}$-$*_{SF4}$ that describe the B-spline part of the radial shape function are set to 0.5 With these parameters, the B-spline curve reduces then to a symmetric bump, similar to a parabola.
The compressor profile is defined by 19 engineering parameters. Ten parameters are fixed based on the experience from previous design optimizations. The resulting 9 engineering parameters are encoded by the radial shape functions leading to a total number of 27 decision variables. Three decision variables are added describing the inlet angle variation $\Delta\alpha$ for each profile. Recall, that the flow is computed at the design inlet angleα_{D} and at $\alpha_{\mathrm{D}} \pm \Delta\alpha$. Once the radial shape functions are defined by the set of decision variables, the engineering parameters for each profile are obtained by computing the value of the radial shape functions at the radius corresponding to the streamline of interest. For the profile length L and the leading edge thickness d, this procedure is outlined in Fig. 11.12. The streamlines are labeled SL1 to SL9 from hub to casing.

11.5.2 Objective Function

The objective function for the blade optimization differs from the profile optimization in two aspects. First, the aerodynamical objectives and constraints have to be summed over the 3 streamlines and second, the MI indication replaces the simple constraint on the profile thickness. In mathematical terms:

$$
\begin{aligned}
f = \sum_{i=1}^{3} \Big\{ \quad & (l_{\alpha_\mathrm{D}} + l_{\alpha_\mathrm{D}-\Delta\alpha} + l_{\alpha_\mathrm{D}+\Delta\alpha}) \\
& -a_0 \Delta\alpha \\
& +a_1 \max(0, |\Delta\beta| - 0.1^\circ) \\
& +a_2 \max(0, H_{12} - H_{12}^{\mathrm{limit}}) \\
& +a_3 \max(0, \mathrm{Ma}_{\max} - \mathrm{Ma}_{\max}^{\mathrm{limit}}) \Big\}^{\mathrm{profile}_i} \\
+a_4 M_{\mathrm{I}}, &
\end{aligned}
\tag{11.4}
$$

where $a_{\{0,1,2,3,4\}}$ are user-defined weights for the different addends of the fitness function.

11.5.3 Optimization Loop

The optimization loop comprises an optimization algorithm and the sequence of design tools as shown in Fig. 11.13. For each design, the section parameters are computed from the decision variables, using the radial shape functions. Then, the profile generator computes profile sections for all streamlines. In the next step, the MI indicator is computed. If the latter is below a user-defined limit, the Q3D flow analysis is performed, otherwise the flow analysis is skipped and a fitness function is computed as a sum of the MI indicator plus a penalty, which is higher than the largest possible flow analysis result. The Q3D flow is computed on the hub, blade mid-span, and tip streamline. The flow analysis is the computationally most expensive part, especially since three incidence angles have to be calculated for each streamline. Finally all objectives and penalties are aggregated into the objective function.

11.5.4 Optimization Results

The automated optimization procedure was employed in the design of four rotor blades of a gas turbine compressor, which will be denoted as Cases A to D.

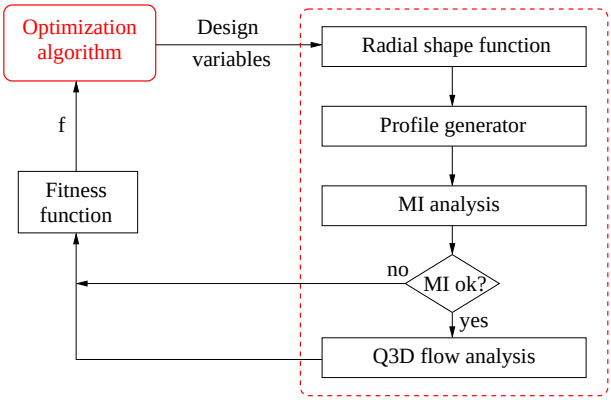

Figure 11.13: Optimization loop for compressor blade optimization

The rotor blades originate from adjacent stages of the subsonic mid-part of the compressor. A typical optimization run requires approximately 4.000 design evaluations. This corresponds to half a day on a four-processor Linux cluster. The resulting Mach number distributions for the hub, mid-span, and tip profile sections are displayed in Figs. 11.14 and 11.15. Compared to the conventional controlled-diffusion airfoils (CDA), the optimized profiles are more front-loaded (see Subsection 11.4.4). It is interesting to note that the general trends for the Mach number distributions are similar for the different rows. This indicates that the optimization procedure does not converge towards completely different optima. Fig. 11.16 compares the loss polar for an existing manually designed mid-span profile and the optimized profile for Case A. The operating range is indeed increased by 15% on both sides of the polar with the loss at design point being about equal. The optimized blades show similar pronounced front loading as the optimized profiles in Subsection 11.4.4. Compared to the profile optimization, the blade optimization considers extended mechanical integrity constraints. Since the resulting profile shapes of both profile and blade optimization are similar, it can be assumed that the mechanical integrity has no significant influence on the profile loading distribution.

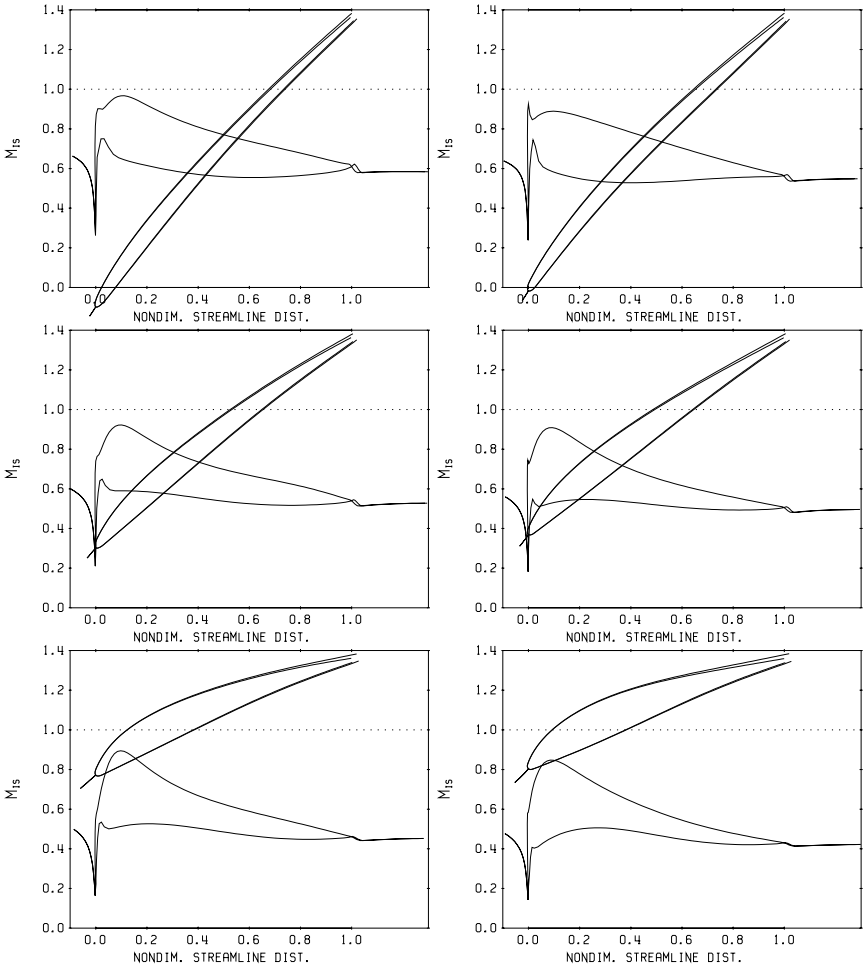

Figure 11.14: Profile shapes and Mach number distributions for the tip (top), mid-span (middle) and hub (bottom) streamline of two rotors from adjacent compressor blades denoted as Case A (left) and Case B (right).

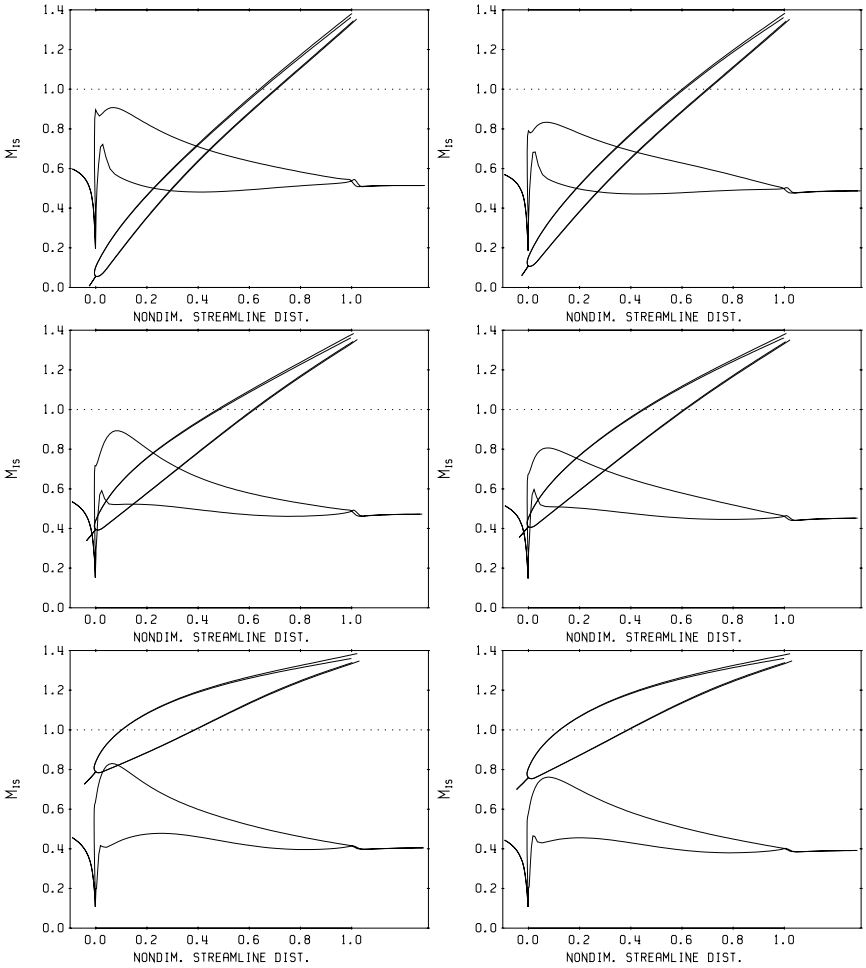

Figure 11.15: Profile shapes and Mach number distributions for the tip (top), mid-span (middle) and hub (bottom) streamline of two rotors from adjacent compressor blades denoted as Case C (left) and Case D (right).

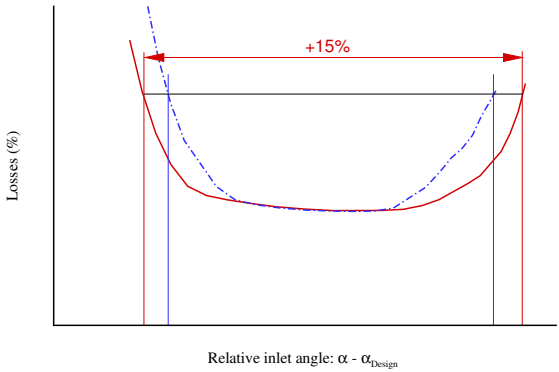

Figure 11.16: Loss polar of the manually designed [dash-dotted line] and the optimized [solid line] mid-span profile (Case A).

11.5.5 Validation of the Blade Optimization by 3D RANS Simulation

The considered optimization approach computes the aerodynamic flow by a quasi-3D simulation on three conical surfaces. Compared to 3D CFD simulations, this simplification of the flow includes two major assumptions: First, the conical surfaces that result from the meridional (S2) through-flow code are assumed to be similar to the streamlines of the real flow in the machine. Second, the impact of 3D flow effects like 3D vortex structures or secondary airflows is assumed to be negligible for the optimization. In order to prove that these assumptions are justified, 3D Reynolds-averaged Navier-Stokes (RANS) simulations are performed and the resulting flow field is compared with the Q3D solution.
The RANS solver is Stage3D, an in-house CFD code of Alstom Power. Stage3D solves the steady-state flow problem on a structured, finite-volume discretization. The turbulence model is the 2-equation k-ω model. An explicit time integration method is used to solve the flow field on a structured grid with a multi-grid formulation. The solver is capable of computing several compressor rows with hub/tip clearances, shrouds and rotating or fixed hubs and casings, if needed. The inlet boundary conditions are the total temperature, total pressure, and inlet flow angle. At the outlet, static pressure is fixed at the casing assuming radial equilibrium. For real compressors, the flow at the interface of two adjacent blade rows are unsteady

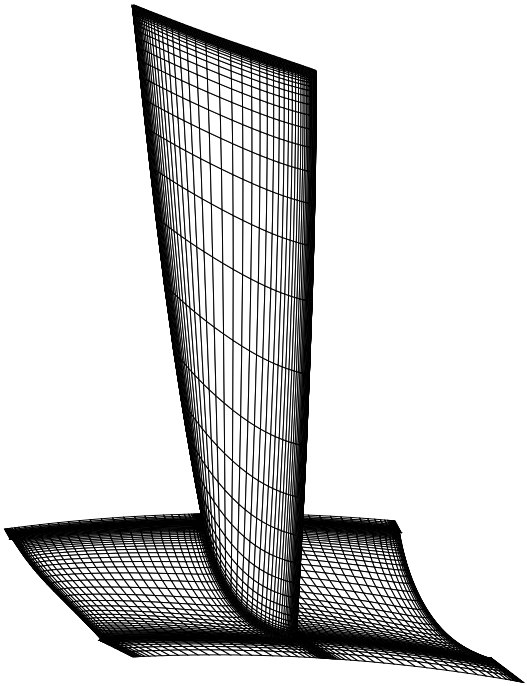

Figure 11.17: Computational grid for the 3D RANS simulation

in circumferential direction due to effects like the rotating wakes. For steady-state simulation, flow conditions are averaged at the interface by the so-called 'mixing-plane'. The mixing-plane transfers circumferentially averaged flow properties to the adjacent blade row. Thus inhomogeneous structures are lost and total energy is not conserved.
The RANS simulations are performed for the optimized rotor blade of case A. Fig. 11.17 shows the blade and the computational domain. In order to be less dependent on exact boundary conditions for the rotor, the upstream and downstream stator are added to the computational domain and linked by mixing-planes. Thus, inserting stator rows damps the effect of the boundary conditions on the rotor.
In Fig. 11.18 (left), the Mach number distribution is plotted for the same three conical streamlines as for the Q3D simulations. The Mach number distributions show

a very good agreement with the Q3D simulations for case A given in Fig. 11.14. Especially the coincident Mach number distribution at the leading edge indicate that the incidence angle is similar and since no Mach peaks occur on either suction or pressure side, the incidence agrees with the design incidence. Furthermore, for Q3D and RANS the peak of the Mach number distribution is of similar value and position. Only for tip streamline, the Mach peak of the RANS is slightly shifted downstream towards the middle of the profile. From our experience, the secondary flow through the tip gap is mainly responsible for this shift.
In the right half of Fig. 11.18, contour plots of the axial velocity are given in order to illustrate the wake of the profiles. The flow field shows no visible recirculation zones. The contour lines indicate also the boundary layer thickness at the aft of the profile. For the Q3D simulation, the thin lines along the profiles in Fig. 11.14 show the displacement thickness. The displacement thickness is always significantly thinner than the visible boundary layer of the real flow. Nevertheless it seems that the RANS shows larger boundary layers.

11.6 Conclusions

This chapter presents the automated optimization procedure of subsonic compressor profiles and blades. Profiles and blades are analyzed by Q3D CFD for design and off-design conditions. The off-design conditions are realized by an incidence variation of $\pm\Delta\alpha$ from the design incidence. The two objectives of the optimization are the minimization of losses at all 3 incidences and the maximization of the incidence multiplier.
For the profile optimization, the problem was formulated as a single objective problem by weighting the objectives and as a Pareto optimization problem. Different evolution strategies and GPOP are analyzed and compared to Pareto optimization algorithms. The single objective algorithms show clearly better performance than the Pareto optimization algorithms. Non of the single objective results was dominated by a Pareto optimization result. Furthermore, among the single objective algorithms, GPOP converged significantly faster than the considered evolution strategies and also led to better results. The optimized profile show a pronounced front loading compared to conventional controlled diffusion airfoils. The front loading is a consequence of the early transition that occurs due to the high Reynolds number and turbulence level in stationary gas turbines.
For the blade optimization, a 3D parameterization is proposed. CFD analyses are performed at three different streamlines. The mechanical integrity is analyzed by

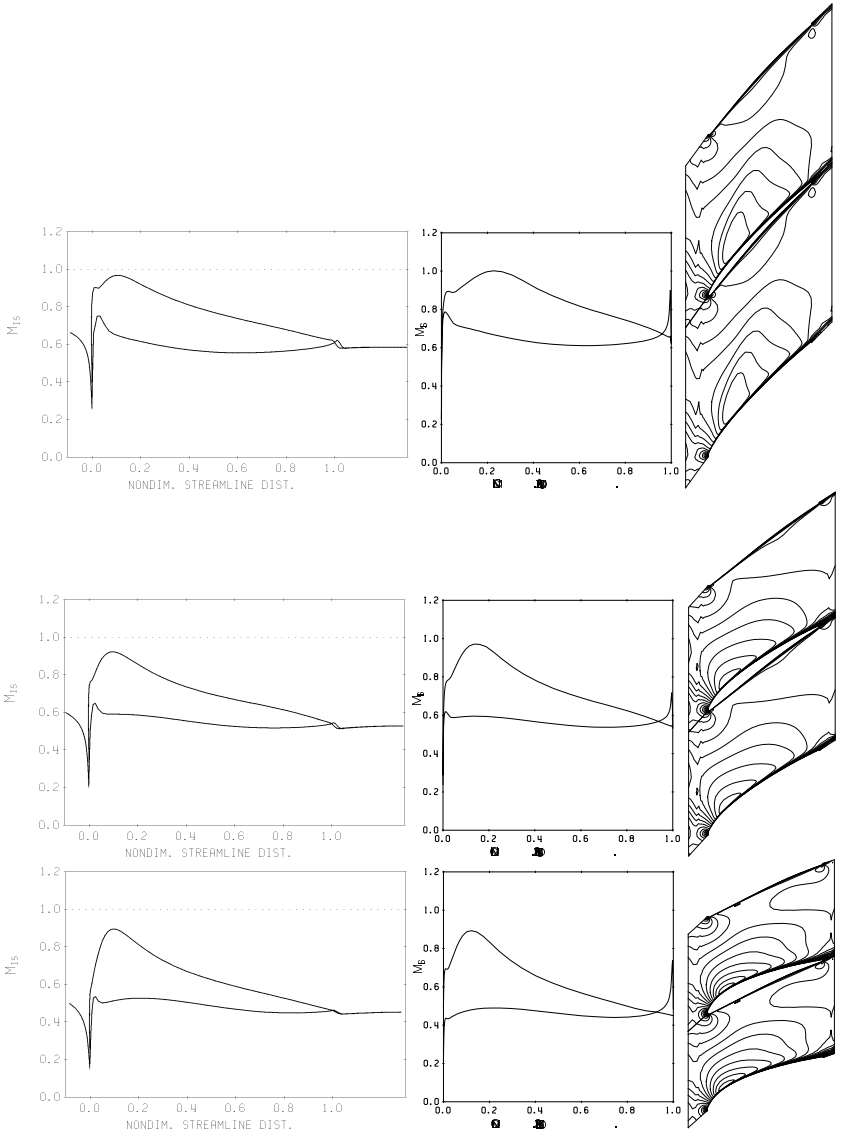

Figure 11.18: Mach number distributions resulting from the Mises (left) and 3D RANS simulation (middle); Contour plot for the axial velocity (right) from 3D RANS simulation of Case A for the tip (top) mid-span (middle) and hub (bottom) streamline.

a beam model for eigenfrequencies and stresses. Rotor blades from four adjacent compressor stages are optimized. The resulting blades show low aerodynamic losses, sustain aerodynamic and centrifugal forces, and avoid critical eigenfrequencies. In addition they show an increased operating range compared to the manual design, while maintaining about equal aerodynamical losses at the design operating point. All optimization runs are started from randomly initialized blades and resulted in similar loading distributions (i.e., profile shapes). Thus, the optimization algorithm converges robustly to similar optima.

Although the flow in a real compressor is three-dimensional, the Q3D simulation is still eligible for the compressor optimization: the Q3D simulations show similar Mach number distributions when compared to 3D RANS simulations. The key advantage of Q3D compared to RANS is the tremendous cost reduction in CPU time, which is especially important since in compressor design a large number of rows needs to be optimized.

In this optimization approach, no gradient information is necessary and the expensive task of formulating an adjoint approach is not necessary. An optimization of a compressor blade requires about 4.000 design evaluation. This corresponds to roughly half a day on a four-processor Linux cluster. The goal of this optimization procedure was to find blade shapes that fulfill all constraints of the real design process and matches with a certain design philosophy. The design philosophy can be obtained by setting constraints to the major flow quantities like, e.g., the peak Mach number or by stressing off-design performance. This goal was reached.

Part IV

Conclusions and Outlook

Chapter 12

Conclusions

Optimization of engineering problems in an automated setup requires blending of domain knowledge with expertise in optimization techniques. Analysis of the problem specific requirements is always the first step in setting up an automated optimization. Then, an optimization algorithm is chosen with respect to the problem requirements. The characteristics and capabilities of efficient optimization algorithms can be identified by comparing different algorithms on test functions. The test functions should be selected such that they reflect the assumed problem features. Improvements of existing algorithms are often necessary, when the performance of the different algorithms is not satisfying for these functions.

Motivated by the need for automated optimization in designing gas turbine components, we proposed a number of algorithmic developments to address these issues. We believe that the proposed methods are also highly suitable for other problems sharing key features with the design of gas turbine components such as noisy and conflicting objectives, expensive function evaluations and only pointwise information about the objective functions. Experimental test-rigs for gas turbine burners represent a noisy multi-objective problem. The goal is to obtain an approximation of the Pareto front for minimizing NO_x emissions and for reducing thermo-acoustic pressure waves (pulsations). Both objectives are time averaged measurements and thus noisy. Furthermore, the optimization is expensive, as only about 300 different solutions can be evaluated within a day. A first optimization run with SPEA showed that the optimization is easily trapped in noisy solutions and outliers. As most well established multi-objective evolutionary algorithms, SPEA is elitist, keeping solutions for infinite time, if no superior solution is found. This is detrimental for noisy problems and lead to the idea of re-evaluating solutions in certain intervals in order to limit the impact of a solution.

The compressor optimization problem has many differences when compared to the burner optimization problem. While the noise level is high for the burner optimization, the considered CFD solver for the compressor blades results in a

low level noise. Furthermore, while in the burner optimization, large mutation steps are needed for a noticeable change in the objective functions, the blade design is very sensitive to small modifications on the blade shape as the problem is highly constraint and the aerodynamics are mainly dependent on the curvature of the blade. The profile optimization problem was formulated both as a single and multi-objective problem. For single objective formulation, there exist various evolution strategies in literature with adaptive mutation distribution, which show efficient optimization. The success rule of Rechenberg and the Evolution Strategy with Covariance Matrix Adaptation (CMA-ES) showed good performance. Most popular in multi-objective optimization is the concept of dominance for the fitness assignment, used in powerful algorithms such as SPEA or NSGA-II. This concept was analyzed theoretically and experimentally. A estimate of the convergence limits was derived as a function of the population size. As a consequence, a sufficiently large population is required for converging close (in objective space) to the Pareto front.

Motivated by the successful step size adaptation techniques in single objective optimization, an adaptive mutation and recombination operator was introduced for multi-objective optimization based on a self-organizing map. These two operators were compared to various non-adaptive operators and showed improved performance on test functions. However, for the profile design problem, all considered multi-objective algorithms showed worse results than the single-objective algorithms. This leads to the conclusions that for problems which require highly converged solutions, the step size adaptation mechanisms of single objective algorithms still outperform multi-objective techniques.

Reducing the number of evaluated solutions in an optimization run is essential, especially if the evaluation is expensive. This thesis showed that evolution strategies are very efficient optimization algorithms. However, the number of evaluation can be further reduced for the considered problems by training empirical models on the set of evaluated solutions as in the proposed algorithm GPOP. While the evolution strategies use only the information of selected solutions for their adaptation process, GPOP uses all evaluated solutions to train the model. This might be one reason for the better performance of GPOP compared to the evolution strategies on the considered test functions and for the profile optimization.

Developing efficient optimization algorithms is one possibility to reduce the number of evaluated solutions in an optimization. Exploiting problem knowledge can also reduce the number of evaluated solutions. For example, bounding the search space to only feasible blades shapes significantly accelerates the compressor blade optimization. Furthermore, setting the penalties for constraint violation to reason-

able (small) values improves the convergence, as too high values may increase the problem difficulty by adding high bumps to the fitness landscape.
This thesis illustrates that automated optimization can find excellent solutions to complex engineering problems. However, a prerequisite is the careful setup of the optimization process with expertise in both optimization algorithms and the problem to optimize. This thesis proposed also some improvements for optimization algorithms and showed that there is still potential for improving these algorithms.

Chapter 13

Outlook and Future Work

This chapter shows possible extension of this thesis for future research.

Convergence Limits of Multi-objective Evolutionary Algorithms

Chapter 5 analyzed the convergence of multi-objective evolutionary algorithms that use the dominance criterion in conjunction with some density measure for the fitness assignment. It was shown that the convergence of these algorithms is limited and depends on the size of the population that stores the nondominated solutions of the optimization. While NSGA-II stores these solutions in the parent population, SPEA and SPEA2 store them in a separate archive.
The theoretical convergence analysis showed that these optimization algorithms stagnate at a certain distance from the Pareto front. For cutting the distance in half, the population size must be increased by a factor of two for a two-objective problem and by a factor of 4 for a 3-objective problem.
An open question is how the population size scales with more than three objectives. Assuming that the population scales with 2^{m-1}, where m is the number of objectives, the population needs to be increased by a factor of $2^3 = 8$ for four objectives. Thus, increasing the number of objectives may lead to an exponential increase in the required population size.
The stagnation occurs, since the selection pressure drops rapidly when converging to the Pareto front. Selection pressure exists only if at least one solution dominates another solution. Since a limited population cannot dominate the entire objective space, the selection pressure decays as soon as the population converges within a distance to the Pareto front.
As a further research topic, it would be interesting to analyze growing population sizes for the fitness assignment. Increasing populations would constantly reduce the nondominated objective space. However, a limited parent population would be preferred for generating offspring, since the parent population should be uniformly

distributed along the current nondominated front, in order to avoid preference of a certain region on the front. The limited parent population could be obtained by using the density measure to select parents.

Multi-objective Evolutionary Algorithms for Noisy Objective Functions

Chapter 6 introduced modifications to the typically elitist multi-objective optimization algorithms for increased robustness to noisy problems. They key concept is to re-evaluate solutions in certain intervals. The fitness of a solution may improve due to the noise in the evaluation. However, in the re-evaluation, the noise may have an opposite effect. This re-evaluation promotes on average solutions of higher fitness and removes noisy solutions or outliers. In this thesis, the concept of re-evaluation was extended by assigning dominance dependent re-evaluation interval. The interval is inversely proportional to the number of solutions that a solution dominates. Thus, solutions that are very dominant are assigned the shortest interval and are re-evaluated first in order to limit their impact on the optimization.
While the performance analysis in Chapter 6 compared these modifications with different approaches from literature for SPEA, it would be interesting to analyze how different selection schemes like NSGA2 compare to SPEA.

Growing Self-Organizing Maps for Multi-Objective Optimization

Chapter 7 proposes Self-Organizing Maps (SOMs) as an adaptive recombination operator for multi-objective evolutionary algorithms. SOMs are a subclass of neural networks that allow a mapping of a high dimensional input space to a regular lattice of neurons. In the context of multi-objective evolutionary algorithms, the parent population is mapped onto the lattice of neurons. The mapping is performed in decision space since recombination operates in the decision space.
Two extensions of the SOM recombination operator could be addressed. The first extension addresses the lattice structure. In the proposed SOM operators, the lattice dimension is set equal to the dimension of the Pareto front. For example, for a two- and three-dimensional objective space, the dimension of the lattice is one and two, respectively. In general, mapping data onto a lattice leads to the best results, if the data is located in a subspace, which is of equal dimension as the SOM. This assumption is surely not valid for all parent populations. Thus, the influence of different lattice dimensions should be analyzed.
As a second extension, the SOM could be replaced by a more flexible structure like, e.g., a growing neural gas [42]. Two advantages can be found for the growing

gas. First, no lattice dimension has to be defined for the growing neural gas, as the connectivity between the gas neurons are evolved. Second, the neural gas allows mapping of also discontinuous data. In the thesis of Michele Milano [77], neural gases have already been applied to single objective optimization.

Accelerating Evolutionary Algorithms Using Fitness Function Models

Knowledge from the history of evaluated solutions can be exploited by various approaches. Evolution strategies exploit this information by adapting the mutation distribution based on successful mutations. In Chapter 8 a different approach was presented, where the information was used to train an empirical model, in particular a Gaussian process. Then, evolutionary algorithms are used to search the minimum of the model prediction for four different merit functions. The merit functions balance the two conflicting objectives of finding better solutions and improving the model by exploring the search space. The resulting minima for the different merit functions are promising new solutions and evaluated on the expensive problem.
The optimization procedure introduces local modeling and local (bounded) search around the currently best solution as new properties. For local modeling, the number of training data was fixed in order to limit the computational effort of training the model as the computational cost scales with $\mathcal{O}(N^3)$, where N is the number of training data. The experimental analysis shows that the required number of training data is problem dependent and also increases with the problem dimension. Setting the size of the training data is thus problematic.
Future research may address the automatic setting of the number of training data. For the analyzed test problems, the likelihood function of the Gaussian process showed a high positive value for converging optimization runs, while a negative value indicated insufficient training data and a low convergence. This observation could be analyzed on a more general set of test functions and/or could be theoretically analyzed in order to find an automatic procedure for setting the size of the training set.

Multi-objective Optimization of Noisy Combustion Processes

Chapter 10 presents an automated optimization of a gas turbine burners for minimizing pulsation and emissions of the burner. The burner is evaluated in an atmospheric test-rig. The optimization is performed by controlling the fuel mass flow with either continuous or digital valves. Both cases lead to an approximation of

the Pareto front for the two objectives.
Compared to the real gas turbine, the test-rig implies two simplifications. First, the test-rig combusts under atmospheric conditions, while the gas turbine operates at elevated pressures. Second, the test-rig comprises a single burner, while in the gas turbine, several burners are aligned in an annular combustion chamber.
Thus, the optimization results have to be validated. The first simplification was addressed by Paschereit *et al.* [86], who analyzed the initial solution as well as two optimized solutions in an elevated pressure test-rig. These analyses confirmed the improvement of the optimized solution compared to the initial solution. Validation in the real gas turbine are still missing.
The optimization results for the burner showed that evolutionary algorithms are able to optimize experimental setups and can lead to a valuable increase in performance. As a future research project, applying evolutionary algorithms to a gas turbine would be an interesting topic. The gas turbine offers more degrees of freedom for the optimization. For example, instead of controlling the mass flow through the different nozzles of a single burner, the mass flow ratio of the different burners in the gas turbines could be varied. Clearly, optimizing real gas turbines is much more expensive than the considered burner test-rig. However, the impact on the gas turbine performance might be significant and the results do not imply uncertainties due to model simplifications. Furthermore, since the pulsations and emissions differ usually between the gas turbines in the field, optimizing a gas turbine for the specific environmental conditions could also be considered.

Automated Design Optimization of Compressor Profiles and Blades

In Chapter 11, an automated optimization procedure for subsonic gas turbine compressor blades is presented. Rotor blades of four adjacent stages of the mid-part of a compressor are optimized for minimizing the aerodynamical losses while several constraints are set on the main aerodynamical properties and the mechanical integrity.
The procedure bases on the design of blade sections (profiles) by Q3D CFD. Compared to 3D RANS simulations, Q3D is very inexpensive allowing blade optimization over night. Furthermore, the procedure has been verified by comparing the results with 3D RANS simulations. However, the procedure also implies several limitations. The procedure is only applicable for the mid-stages of the compressor since the flow analysis bases on a Q3D CFD tool that cannot resolve 3D flow effects. The transonic front stages are difficult to optimize due to the three-dimensional nature of shocks. Hub and casing influence the shock position. In the

rear part of the compressor, the compressor blades have a low aspect ratio (i.e., ratio of height to chord). The ratio of the tip gap to the blade span is larger than for the front and mid-part of the compressor and tip gap leakage influences the flow. Furthermore, the boundary layers at hub and casing increase over the compressor length and complicate the flow in the rear stages.
Future research may address the optimization of front and rear stages by resolving 3D effects with 3D RANS simulations. Setting up an optimization loop with 3D RANS simulation may be simple as only the Q3D solver needs to be replaced by the RANS simulation, while all remaining tools (optimization algorithm, blade parameterization, mechanical integrity analysis) can be kept unchanged. However, 3D RANS optimization may require a further analysis of possible design objectives. While for the Q3D simulations in this thesis, the profile losses were minimized for design and off-design conditions, a blade design with RANS simulations may consider a constant total pressure in radial direction, the choked condition or the shock position as design objectives. In addition, 3D RANS simulations resolve the effect of 3D blading. While in the Q3D approach, all blades are designed by a straight stacking of the profiles, 3D blading may improve the performance by leaning, bowing, or sweeping the blades as described by Denton and Xu [29].

Bibliography

[1] H. A. Abbass. The Self-Adaptive Pareto Differential Evolution Algorithm. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 831–836, Piscataway, New Jersey, May 2002. IEEE Service Center.

[2] D. V. Arnold and H.-G. Beyer. Noisy optimization with evolution strategies. Ci-report 117/01, University of Dortmund, Germany, 2001.

[3] D. V. Arnold and H.-G. Beyer. A comparison of evolution strategies with other direct search methods in the presence of noise. *Computational Optimization and Applications*, 24(1):135–159, 2003.

[4] D. V. Arnold and H.-G. Beyer. Investigation of the (μ, λ) -es in the presence of noise. In *Proceedings of the CEC'01 Conference*, Piscataway, New York.

[5] T. Bäck. Self-adaptation. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation, Sec. C7.1: pp. 1-15*, 1997.

[6] T. Bäck, D. B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation, Sec. C7.1: pp. 1-15*. 1997.

[7] T. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: comments on the history and current state. *IEEE Trans. on Evolutionary Computation*, 1(1):3–17, 1997.

[8] T. Bäck and M. Schütz. Intelligent mutation rate control in canonical genetic algorithms. In *International Syposium on Methodologies for Intelligent Systems*, pages 158–167, 1996.

[9] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evolutionary Computation*, 1(1):1–23, 1993.

[10] C. A. L. Bailer-Jones, H. K. D. H. Bhadeshia, and D. J. C. MacKay. Gaussian process modelling of austenite formation in steel. *Materials Science and Technology*, 15(3):287–294, 1999.

[11] S. Barnett. *Matrix Methods for Engineers and Scientists*. McGraw Hill, Inc., 1979.

[12] C. Bischof and A. Griewank. Tools for the automatic differentiation of computer programs. In G. Alefeld, O. Mahrenholtz, and R. Mennicken, editors, *ICIAM/GAMM 95: Issue 1: Numerical Analysis, Scientific Computing, Computer Science*, pages 267–272, 1996.

[13] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. Pisa - a platform and programming language independent interface for search algorithms. In *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO), Faro, Portugal*, pages 494–508. Springer Lecture Notes in Computer Science, 2003.

[14] A. J. Booker, J. E. Dennis Jr., P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset. A rigorous framework for optimization of expensive functions by surrogates, ICASE Report No. 98-47. Technical report, NASA Langley Research Center Hampton, VA, 1998.

[15] A. J. Booker, J. E. Dennis Jr., P. D. Frank, D. B. Serafini, V. Torczon, and Michael W. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17(1):1–13, 1999.

[16] D. Büche and R. Dornberger. New evolutionary algorithm for multi-objective optimization and the application to engineering design problems. In *Proceedings of the Fourth World Congress of Structural and Multidisciplinary Optimization*, Dalian, China, 2001. International Society of Structural and Multidisciplinary Optimization (ISSMO).

[17] D. Büche, S. Müller, and P. Koumoutsakos. Self-adaptation for multi-objective evolutionary algorithms. In *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO), Faro, Portugal*. Springer Lecture Notes in Computer Science, 2003.

[18] J. Chung, J. Shim, and K. D. Lee. Shape optimization of high-speed axial compressor blades using 3d navier-stokes flow physics. In *Proceedings of the ASME Turbo Expo 2001*, 2001.

[19] C. A. Coello Coello. *List of references on evolutionary multi-objective optimization*. http://www.lania.mx/˜ccoello/EMOO/EMOObib.html, Last accessed July 2002.

[20] C. A. Coello Coello. An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. In *Congress on Evolutionary Computation*, pages 3–13, 1999.

[21] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, 2002. ISBN 0-3064-6762-3.

[22] L. Costa and P. Olivaira. An adaptive elitist evolution strategy for multiobjective optimization. *Evolutionary Computation*, 11(4):417–438, 2003.

[23] K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7(3):205–230, 1999.

[24] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001. ISBN 0-471-87339-X.

[25] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(6):115–148, 1995.

[26] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[27] K. Deb and T. Goel. Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergence. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 67–81. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[28] J. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Series in Computational Mathematics*. Prentice-Hall, 1993.

[29] J. D. Denton and L. Xu. The exploitation of three-dimensional flow in turbomachinery design. *Proceedings of the Institution of Mechanical Engineers, Part C, Journal of Mechanical Engineering Science*, 213:125–137, 1999.

[30] R. Dornberger, P. Stoll, C. O. Paschereit, B. Schuermans, D. Büche, and P. Koumoutsakos. Redundanzfreier Ansteuerungsmechanismus der Ventile zur Beeinflussung des Mischungsprofiles für die Kontrolle verbrennungsgetriebener Schwingungen. *German patent*, pages –, 2001. # 101 04 150.0, ALSTOM Power (Schweiz) AG.

[31] M. Drela and H. Youngren. A user s guide to MISES 2.1. Technical report, Massachusetts Institute of Technology, 1995.

[32] W. Egartner and V. Schulz. Partially reduced SQP methods for optimal turbine and compressor blade design. In Bock et al., editor, *Proceedings of the 2nd European Conference on Numerical Mathematics and Advanced Applications*, 1998.

[33] M. A. El-Beltagy and A. J. Keane. Evolutionary optimization for computationally expensive problems using gaussian processes. In Hamid Arabnia, editor, *Proc. Int. Conf. on Artificial Intelligence IC-AI'2001, Las Vegas*, pages 708–714. CSREA Press, 2001.

[34] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou. Metamodel-assisted evolution strategies. In *Parallel Problem Solving from Nature - PPSN VII*, pages 361–370. Springer-Verlag, 2002.

[35] M. Fleischer. The measure of pareto optima. In *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO), Faro, Portugal*, pages 494–508. Springer Lecture Notes in Computer Science, 2003.

[36] L. J. Fogel. Autonomous automata. *Industrial Research*, 4:14–19, 1962.

[37] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In S. Forrest, editor, *Fifth International Conference on Genetic Algorithms*, pages 416–423, 1993.

[38] C. M. Fonseca and P. J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16, 1995.

[39] C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele. *Evolutionary Multi-Criterion Optimization*. Springer-Verlag, Second International Conference, EMO 2003, Portugal, 2003.

[40] M. C. Fonseca and P. J. Fleming. Multi-objective genetic algorithms made easy: Selection, sharing and mating restrictions. In *Proceedings of the 1st International Conference on Genetic Algorithms in Engineering Systems: Innovations and Application*, pages 45–52, 1995.

[41] B. Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.

[42] B. Fritzke. A growing neural gas network learns topologies. In *NIPS*, 1994.

[43] B. Fritzke. Growing grid - a self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters*, 2(5):9–13, 1995.

[44] K. C. Giannakoglou. Acceleration of genetic algorithms using artificial neural networks - theoretical background. In *von Karman Institute Lectures Series on Genetic Algorithms for Optimization in Aeronautics and Turbomachinery*, 2000.

[45] K. C. Giannakoglou. Optimization and inverse design in aeronautics: how to couple genetic algorithms with radial basis function networks. In J. Periaux, P. Joly, O. Pironneau, and E. Onate, editors, *Innovative Tools for Scientific Computation in Aeronautical Engineering*, 2001.

[46] M. N. Gibbs. *Bayesian Gaussian Processes for Regression and Classification*. Ph.D. thesis, University of Cambridge, England, 1997.

[47] M. N. Gibbs and D. J. C MacKay. Manual for tpros, v2.0, 1997.

[48] A. P. Giotis and K. C. Giannakoglou. Single- and Multi-Objective Airfoil Design Using Genetic Algorithms and Artificial Intelligence. In *Proceedings of EUROGEN'99*, 1999.

[49] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Company, 1989.

[50] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette, editor, *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum Associated, 1987.

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

[52] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[53] S. Haykin. *Neural Networks - A Comprehensive Foundation*. Prentice-Hall, 1994.

[54] J. H. Holland. Outline for a logical theory of adaptive systems. *Journal of the Associated Computer Mathematics*, 9(3), 1962.

[55] J. Horn. Multicriterion decision making. In T. Bäck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation, Sec. F1.9: pp. 1-15*, 1997.

[56] J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Computation*, pages 82–87, 1994.

[57] E. J. Hughes. Evolutionary multi-objective ranking with uncertainty and noise. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[58] L. Huyse and R. M. Lewis. Aerodynamic shape optimization of two-dimensional airfoils under uncertain operating conditions, ICASE No. 2001-1. Technical report, NASA Langley Research Center Hampton, VA, 2001.

[59] Y. Jin, T. Okabe, and B. Sendhoff. Adapting Weighted Aggregation for Multiobjective Evolution Strategies. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 96–110. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[60] Y. Jin, M. Olhofer, and B. Sendhoff. On evolutionary optimisation with approximate fitness functions. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO, Las Vegas, Nevada*, pages 786–793, 2000.

[61] Y. Jin, M. Olhofer, and B. Sendhoff. Managing approximate models in evolutionary aerodynamic design optimization. In *Proceedings of IEEE Congress on Evolutionary Computation, Vol. 1, Seoul, Korea*, pages 592–599, 2001.

[62] M. E. Johnson, L. M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *Journal of Statistical Inference and Planning*, 26:131–148, 1990.

[63] V. Khare, X. Yao, and K. Deb. Performance scaling of multi-objective evolutionary algorithms. In *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO), Faro, Portugal*, pages 376–390. Springer Lecture Notes in Computer Science, 2003.

[64] S. Kirkpatrick, Gelatt C.D., and Vecchi M.P. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[65] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43, 1982.

[66] T. Kohonen. Self-organizing maps: optimization approaches. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Proceedings of ICANN'91, International Conference on Artificial Neural Networks, Vol. II*, pages 981–990. North-Holland, Amsterdam, 1991.

[67] U. Köller, R. Mönig, B. Küsters, and H.-A. Schreiber. Development of advanced compressor airfoils for heavy-duty gas turbines- Part I: Design and optimization. *ASME Journal of Turbomachinery*, 122:397–405, 1999.

[68] F. Kursawe. A Variant of Evolution Strategies for Vector Optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197, Berlin, Germany, 1991. Springer-Verlag.

[69] F. Kursawe. Towards self-adapting evolution strategies. In *1995 IEEE Conf. on Evolutionary Computation*, volume 1, pages 283–288, Piscataway, NJ, 1995. IEEE Service Center.

[70] B. Küsters, H.-A. Schreiber, U. Köller, and R. Mönig. Development of advanced compressor airfoils for heavy-duty gas turbines- Part II: Experimental and theoretical analysis. *ASME Journal of Turbomachinery*, 122:406–415, 1999.

[71] M. Laumanns, G. Rudolph, and H.-P. Schwefel. Mutation Control and Convergence in Evolutionary Multi-Objective Optimization. In *Proceedings of the 7th International Mendel Conference on Soft Computing (MENDEL 2001)*, Brno, Czech Republic, 2001.

[72] Y.-T. Lee, L. Luo, and T. W. Bein. Direct method for optimization of a centrifugal compressor vaneless diffuser. *ASME Journal of Turbomachinery*, 123(1):73–79, 2001.

[73] D. J. C. MacKay. Gaussian processes - a replacement for supervised neural networks? In *Lecture notes for a tutorial at NIPS*, 1997.

[74] J. D. Martin and T. W. Simpson. Use of adaptive metamodeling for design optimization. In *AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 2002.

[75] J. R. R. A. Martins, J. J. Alonso, and J. J. Reuther. Aero-structural wing design optimization using high-fidelity sensitivity analysis. In *CEAS Conference on Multidiciplinary Analysis and Optimization*, 2001.

[76] J. R. R. A. Martins, I. M. Kroo, and J. J. Alonso. A method for sensitivity analysis using complex variables. In *38th AIAA Aerospace Sciences Meeting & Exhibit, AIAA Paper 2000-0689, Reno, NV*, 2000.

[77] M. Milano. *Machine Learning Algorithms for Flow Modeling and Control*. Dissertation ETH Zürich, 2002.

[78] M. Milano, J. Schmidhuber, and P. Koumoutsakos. Active learning with adaptive grids. In *International Conference on Artificial Neural Networks*, 2001.

[79] B. L. Miller and D. E. Goldberg. Genetic algorithms, selection schemes, and the varying effects of noise. Illigal report no. 95005, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithm Laboratory, 1995.

[80] S. D. Müller. *Bio-Inspired Optimization Algorithms for Engineering Applications*. Dissertation ETH Zürich, Shaker Aachen, 2002.

[81] H. Nakayama, M. Arakawa, and R. Sasaki. A comptutational intelligence approach to optimization with unknown objective functions. In *Artificial Neural Networks-ICANN 2001*, pages 73–80, 2001.

[82] B. Naujoks, L. Willmes, W. Haase, T. Bäck, and M. Schütz. Multi-point airfoil optimization using evolution strategies. In *European Congress on Computational Methods in Applied Sciences and Engineering*, 2000.

[83] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[84] V. Pareto. *Manuale die Economia Politica*. Societa Editrice Libraria, Milano, Italy, 1906.

[85] O. Paschereit, E. Gutmark, and W. Weisenstein. Structure and control of thermoacoustic instabilities in a gas-turbine combustor. *Combustion Science and Technology*, 138:213–232, 1998.

[86] O. Paschereit, B. Schuermans, and D. Büche. Combustion process optimization using evolutionary algorithm. In *Proceedings of the ASME/IGTI Turbo Expo 2003*. ASME, 2003.

[87] Robin C. Purshouse and Peter J. Fleming. Conflict, Harmony, and Independence: Relationships in Evolutionary Multi-criterion Optimisation. In *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO), Faro, Portugal*, pages 16–30. Springer Lecture Notes in Computer Science, 2003.

[88] S. R. Ranjithan, S. K. Chetan, and H. K. Dakshima. Constraint Method-Based Evolutionary Algorithm (CMEA) for Multiobjective Optimization. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 299–313. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[89] A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In *Parallel Problem Solving from Nature - PPSN V*, pages 87–96. Springer-Verlag, 1998.

[90] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Fromman-Holzboog, 1973.

[91] M. Reinke, R. Carroni, D. Winkler, and T. Griffin. Experimental investigation of natural gas combustion in oxygen / exhaust gas mixtures for zero emissions power generation. In *6th Int. Conf. on Technologies and Combustion for a Clean Environment, Oporto, Portugal*, pages 1:15–20, 2001.

[92] J. J. Reuther, J. J. Alonso, A. Jameson, M. J. Rimlinger, and D. Saunders. Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers: Part i & ii. *AIAA J. of Aircraft*, 36(1):51–71, 1999.

[93] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel distributed processing: explorations in the microstructures of cognition, Bradford Books/MIT Press, Cambridge, MA*, 1:318–362, 1986.

[94] D. Sadnik and W. Glas. Pareto-stabilisation of evolution strategies with the derandomized covariance matrix adoption. In K. C. Giannakoglou, D. T. Tsahalis, J. Périaux, K. D. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 331–336, Athens, Greece, 2001. International Center for Numerical Methods in Engineering (Cmine).

[95] N. L. Sanger. The use of optimization techniques to design-controlled diffusion compressor blading. *ASME Journal of Engineering for Power*, 105:256–264, 1983.

[96] T. Sattelmayer, M. P. Felchlin, J. Haumann, J. Hellat, and D. Styner. Second-generation low-emission combustors for ABB gas turbines: burner development and test at atmospheric pressure. In *Gas Turbine and Aeroengine Congress and Exposition, 90-GT-162*, 1990.

[97] I. F. Sbalzarini, S. D. Müller, and P. Koumoutsakos. Microchannel optimization using multiobjective evolution strategies. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, pages 516–530. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[98] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In Grefenstette, editor, *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, pages 93–100, 1995.

[99] Heinz-Adolf Schreiber, Wolfgang Steinert, and Berhard Küsters. Effects of reynolds number and free stream turbulence on boundary layer transition in a compressor cascade. *ASME Journal of Turbomachinery*, 124(1):1–9, 2002.

[100] H.-P. Schwefel. *Evolutionsstrategie und numerische Optimierung*. PhD Thesis, 1975.

[101] H.-P. Schwefel. *Numerical Optimization of Computer Models*. Wiley, Chichester, 1981.

[102] H.-P. Schwefel. *Evolution and Optimum Seeking*. John Wiley & Sons, 1995.

[103] H.-P. Schwefel and G. Rudolph. Contemporary evolution strategies. In *Proc. of the Third European Conference on Artificial Life (ECAL'95)*, pages 893–907, Granada, Spain, 1995.

[104] L. Schweizer. A fully neural approach to image compression. In *Artificial Neural Networks*, pages 815–82. T. Kohonen et al., 1991.

[105] M. Seeger, C. Williams, and N. Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *9th International Workshop on Artificial Intelligence and Statistics*. Key West, Florida, 2003.

[106] S. Shahpar. A comparative study of optimization methods for aerodynamic design of turbomachinery blades, 2000-gt-523. In *Proceedings of the ASME Turbo Expo 2000*, 2000.

[107] F. Sieverding, M. Casey, B. Ribi, and M. Meyer. Design of industrial axial compressor blade sections for optimal range and performance. In *Proceedings of the ASME/IGTI Turbo Expo 2003*. ASME, 2003.

[108] T. W. Simpson, T. M. Mauery, J. J. Korte, and F. Mistree. Comparison of response surface and kriging models for multidisciplinary design optimization. In *AIAA paper 98-4758*. 7 th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, USA, 1998.

[109] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1995.

[110] R. Storn and K. Price. Minimizing the real functions of the icec'96 contest by differential evolution. In *IEEE Conference on Evolutionary Computation*, pages 842–844, 1996.

[111] K.C. Tan, T.H. Lee, and E.F. Khor. Evolutionary Algorithms with Dynamic Population Size and Local Exploration for Multiobjective Optimization. *IEEE Trans. on Evolutionary Computation*, 5(6):565–588, 2001.

[112] J. Teich. Pareto-front exploration with uncertain objectives. In E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[113] V. Torczon and M. W. Trosset. Using approximations to accelerate engineering design optimization, ICASE Report No. 98-33. Technical report, NASA Langley Research Center Hampton, VA 23681-2199, 1998.

[114] M. A. Trigg, G. R. Tubby, and A. G. Sheard. Automatic genetic optimization approach to 2d blade profile design for steam turbines. In *ASME paper 97-GT- 392, ASME Turbo Expo97, Orlando, Florida*, 1997.

[115] D. A. Van Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm test suite. In J. et al. Carroll, editor, *Proceedings of the 1999 ACM Symposium an Applied Computing*, pages 351–357, 1999.

[116] D. A. Van Veldhuizen and G. B. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 8(2):125–147, 2000.

[117] D. A. Van Veldhuizen and G. B. Lamont. On measuring multiobjective evolutionary algorithm performance. In *Proceedings of the 2000 Congress on Evolutionary Computation*, pages 204–211, 2000.

[118] C. K. I. Williams. Prediction with gaussian processes: From linear regression to linear prediction and beyond. In M. I. Jordan, editor, *Learning and Inference in Graphical Models*, 1998.

[119] L. Willmes and T. Bäck. Multi-criteria airfoil design with evolution strategies. In *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO), Faro, Portugal*, pages 782–795. Springer Lecture Notes in Computer Science, 2003.

[120] D. J. Willshaw and C. von der Malsburg. How paterned neural connections can be set up by self-organization. In *Proceedings of the Royal Society of London B, 194*, pages 431–445, 1976.

[121] M. H. Wright. Direct search methods: once scorned, now respectable. In D. F. Griffiths and G. A. Watson, editors, *Numerical Analysis 1995 (Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis)*, pages 191–208. Addison Wesley Longman, 1996.

[122] E. Zitzler, K. Deb, L. Thiele, C. A. Coello Coello, and D. W. Corne. *Proceedings of the First Conference on Evolutionary Multi-Criterion Optimization*. Springer-Verlag, 2001.

[123] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou, and T. Fogarty, editors, *EUROGEN 2001. Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, Athens, Greece, 2001.

[124] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.

[125] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[126] E. Zitzler, L. Thiele, and K. Deb. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195, 2000.

Curriculum Vitae

Name:	Dirk Büche
Citizen of:	Germany
Born:	March 8^{th}, 1974, Schaffhausen, Switzerland
May 1993	Abitur (High school leaving certificate) Klettgau-Gymnasium Tiengen, Germany
1993 - 1997	University of Stuttgart, Germany, Major: Aeronautical Engineering
1997 - 1998	Northwestern University, Chicago, USA, Department of Mechanical Engineering, Granted by the German Academic Exchange Service
Oct. 1999	Diploma in Aeronautical Engineering from the University of Stuttgart, Germany
1999 - 2000	Research engineer at the ABB-Alstom Cooperate Research Center, Dättwil, Switzerland
2000 - 2003	Swiss Federal Institute of Technology, ETH Zürich Research and teaching assistant at the Institute of Fluid Dynamics (IFD) and the Institute of Computational Science (ICOS) under the supervision of Prof. P. Koumoutsakos
since 2003	University of Applied Sciences Aargau (Switzerland) Research associate at the Institute of Informatics (ifi)

Publications

Parts of this thesis are published in conference proceedings, journals, and patents:

1. D. Büche, P. Stoll, and P. Koumoutsakos. An evolutionary algorithm for multi-objective optimization of combustion processes. *Annual research briefs, Center for Turbulence Research*, Stanford, 2001.

2. P. Stoll, and P. Koumoutsakos. Multi-objective optimization of combustion processes. In *CISM Course on Modeling, Manipulation and Control of Traverse Jets, Udine, Italy*, 2001.

3. D. Büche, M. Milano, and P. Koumoutsakos. Self-Organizing Maps for Multi-Objective Optimization. *Workshop on Learning and Adaptation in Evolutionary Computation at GECCO 2002*, 2003.

4. D. Büche, G. Guidati, P. Stoll, and P. Koumoutsakos. Self-organizing maps for pareto optimization of airfoils. In *Seventh International Conference on Parallel Problem Solving from Nature (PPSN VII)*, Granada, Spain, September 2002.

5. D. Büche, P. Stoll, R. Dornberger, and P. Koumoutsakos. Multiobjective evolutionary algorithm for the optimization of noisy combustion processes. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, 32(4):460–473, 2002.

6. D. Büche, S. Müller, and P. Koumoutsakos. Self-adaptation for multi-objective evolutionary algorithms. In *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO), Faro, Portugal*. Springer Lecture Notes in Computer Science, 2003.

7. D. Büche, N. N. Schraudolph, and P. Koumoutsakos. Accelerating Evolutionary Aalgorithms Using Fitness Function Models. In *Workshop On Learning, Adaptation, and Approximation in Evolutionary Computation, Genetic and Evolutionary Computation Conference (GECCO 2003)*, 2003.

8. D. Büche, G. Guidati, and P. Stoll. Automated design optimization of compressor blades for stationary, large-scale turbomachinery. In *Proceedings of the ASME/IGTI Turbo Expo 2003*. ASME, 2003.

9. D. Büche, N. Schraudolph, and P. Koumoutsakos. Accelerating evolutionary algorithms with gaussian process fitness function models. *IEEE Transactions on Systems, Man and Cybernetics, Part C, Special Issue on Knowledge Extraction and Incorporation in Evolutionary Computation*, 2004, to appear.

10. R. Dornberger, P. Stoll, C. O. Paschereit, B. Schuermans, D. Büche, and P. Koumoutsakos. Redundanzfreier Ansteuerungsmechanismus der Ventile zur Beeinflussung des Mischungsprofiles für die Kontrolle verbrennungsgetriebener Schwingungen. *European patent*, 2001. # 101 04 150.0, ALSTOM Power (Schweiz) AG.

11. R. Dornberger, P. Stoll, C. O. Paschereit, D. Büche, and P. Koumoutsakos. Numerisch experimentelle Optimierung des Mischungsprofils für die Kontrolle verbrennungsgetriebener Schwingungen und Emissionen. *European patent*, 2001. # 101 04 151.9, ALSTOM Power (Schweiz) AG.

12. S. Kern, S.D. Müller, D. Büche, N. Hansen, and P. Koumoutsakos. Learning probability distributions in continuous evolutionary algorithms. In *Workshop on Fundamentals in Evolutionary Algorithms, Thirteenth International Colloquium on Automata, Languages and Programming, Eindhoven*, 2003.

13. O. Paschereit, B. Schuermans, and D. Büche. Combustion process optimization using evolutionary algorithm. In *Proceedings of the ASME/IGTI Turbo Expo 2003*. ASME, 2003.